CHAOSEATER: FULLY AUTOMATING CHAOS ENGI NEERING WITH LARGE LANGUAGE MODELS

Anonymous authors

004

010 011

012

013

014

015

016

017

018

019

021

025

026

044

045

046

047

048

051

052

Paper under double-blind review

ABSTRACT

Chaos Engineering (CE) is an engineering technique aimed at improving the resiliency of distributed systems. It involves artificially injecting specific failures into a distributed system and observing its behavior in response. Based on the observation, the system can be proactively improved to handle those failures. Recent CE tools realize the automated execution of predefined CE experiments. However, defining these experiments and reconfiguring the system after the experiments still remain manual. To reduce the costs of the manual operations, we propose CHAOSEATER, a system for automating the entire CE operations with Large Language Models (LLMs). It pre-defines the general flow according to the systematic CE cycle and assigns subdivided operations within the flow to LLMs. We assume systems based on Infrastructure as Code (IaC), wherein the system configurations and artificial failures are managed through code. Hence, the LLMs' operations in our system correspond to software engineering tasks, including requirement definition, code generation and debugging, and testing. We validate our system through case studies on both small and large systems. The results demonstrate that our system significantly reduces both time and monetary costs while completing a reasonable CE cycle. Our code is available in the Supplementary Material.

028 1 INTRODUCTION 029

Modern software-based services, such as streaming, e-commerce, and conversational AI platforms, are implemented as distributed systems, where each service is divided into smaller services according to specific functionalities. These small services (i.e., functions), along with the communication network that connects them, constitute the entire service. This design, known as microservice architecture (Bucchiarone et al., 2020), enables scalable and continuous deployment while supporting the integration of heterogeneous technologies. On the other hand, the complex dependencies among small services can lead to unexpected, chaotic behavior in the entire system from even minor failures. However, proactively predicting and addressing such complex behavior is challenging.

To address this and improve the resiliency of distributed systems, numerous organizations, including Netflix, Amazon, and Microsoft, have recently adopted Chaos Engineering (CE) (Basiri et al., 2016; 2019). Its concept is that *rather than predicting the chaotic behavior*, *let's observe it directly by artificially injecting the failures into the system*. Based on the observation, we can proactively rebuild a new system that is resilient to the assumed failures. Systematically, CE cycles through four phases for a system:

- 1. *Hypothesis*: Define steady states (i.e., normal behavior) of the system and injected failures. Then, make a hypothesis that *the steady states are maintained in the system even when the failures occur*.
- 2. (*Chaos*) *Experiment*: Inject the failures into the system while logging the system's response behavior.
- 3. *Analysis*: Analyze the logged data and check if the hypothesis is satisfied. If so, this CE cycle is finished here. If not, move to (4).
- 4. *Improvement*: Reconfigure the system to satisfy the hypothesis. The reconfigured system is tested again in (2) and (3), i.e., repeat (2) to (4) until the hypothesis is satisfied.
- In recent years, several CE tools (Netflix, 2012; Amazon Web Services, 2021; Chaos Mesh, 2021; Microsoft, 2023) have advanced the automation of chaos-experiment execution. Moreover, monitor-

054 ing tools (Prometheus, 2012; Grafana Labs, 2021) enable automating metric collection, aggregation, 055 and threshold-based testing during chaos experiments. Hence, the experiment and analysis phases 056 have been mostly automated. However, defining a hypothesis in the *hypothesis* phase, planning a 057 chaos experiment to test the hypothesis in the *experiment* phase, and reconfiguring the system in the 058 improvement phase still remain manual. These manual operations require a complex set of skills, including domain knowledge in networking and CE, the ability to interpret system configurations, logs, and error messages, as well as creative problem-solving for requirement definition, planning, 060 and system reconfiguration. Consequently, while the costs of these operations remain high, their 061 automation has not been achieved yet with existing algorithmic approaches. 062

063 We believe that Large Language Models (LLMs) are the key to overcoming this challenge. LLMs have recently shown promising capabilities across a wide range of tasks, including natural language 064 processing, coding, and operations for networking (Zhao et al., 2023; Jiang et al., 2024; Ahmed 065 et al., 2024; Piovesan et al., 2024). In more recent years, LLMs have also provided promising 066 performance on software engineering (SE) benchmarks (Yang et al., 2024; Cognition Labs, 2024). 067 In the context of software-based systems configured by the Infrastructure as Code (IaC) paradigm, 068 where the system configurations and artificial failures are managed through code, CE operations can 069 be regarded as SE tasks. The hypothesis phase corresponds to a requirement definition to determine the resilience required for the system. In the *experiment* phase, planning an experiment corresponds 071 to the design of testing, and running the experiment requires coding. The analysis corresponds 072 to the verification of the tests. Lastly, the *improverment* phase corresponds to code debugging. 073 Considering their general capabilities, domain knowledge in networking, and potential in SE, it is 074 also expected that the automation of the entire CE cycle can be achieved with LLMs.

075 Here, we propose CHAOSEATER, a system for automating the entire CE cycle with LLMs. It pre-076 defines the general flow according to the systematic CE cycle and assigns subdivided CE operations 077 within this flow to LLMs. The flow (i.e., CE cycle) then progresses by sequentially performing 078 subdivided operations using the assigned LLMs, while processing data inputs and outputs through 079 rule-based algorithms. Our system assumes CE for IaC-based systems, specifically Kubernetes (K8s) (Kubernetes, 2014) systems. In this paper, we present the flow design, rule-based algorithms, CE 081 operations performed by LLMs, and the integration of the latter two to achieve the designed flow. In evaluation, we validate our system through case studies on both a small system and a large system. 082 The results demonstrate that our system significantly reduces both time and monetary costs while 083 completing a reasonable CE cycle. Lastly, we discuss the broader impacts, limitations, and future 084 directions of our system based on this study. 085

The main contributions of this paper are organized as follows:

- We are the first to propose a *system* for automating the entire CE cycle with LLMs, which reduces time and monetary costs in a CE cycle. This proposal would be a starting point towards the full automation of system resilience improvement.
- We publicly release all code for our *system*. This release provides development practices for constructing complex systems that combine LLMs and rule-based algorithms.
- We validate our *system* through case studies and discuss its broader impacts, limitations, and future directions. The results demonstrate the new potential of LLMs in CE, while our discussion provides insights for advancing the full automation of CE.
- 094 095 096

097

098

087

880

091

092

093

2 PROPOSED SYSTEM: CHAOSEATER

099 In this section, we describe the technical details of CHAOSEATER. Figure 1 shows its simplified 100 system diagram. It takes as input instructions for the CE cycle (optional) and a folder containing 101 K8s manifests (Kubernetes, 2014) and a Skaffold configuration file (Google, 2019). In short, K8s 102 manifests are system configuration files that define the resources (i.e., small services) that constitute 103 a system, while a Skaffold configuration file defines the process to automatically deploy those re-104 sources in a K8s cluster. It then conducts a CE cycle for those inputs through five divided phases: 105 pre-processing, hypothesis, experiment, analysis, improvement, and post-processing phases. Finally, it outputs a summary of the completed CE cycle and a modified folder containing K8s manifests that 106 have been modified to satisfy the hypothesis defined in the hypothesis phase and their Skaffold con-107 figuration file.



Figure 1: A simplified system diagram of CHAOSEATER and its input and output. CHAOSEATER autonomously completes the systematic CE cycle using internal agents and IaC tools. Note that only the representative inputs and outputs of agents are illustrated within the diagram. The two K8s clusters within the diagram refer to the same one.

To ensure that the LLMs perform as intended, our *system* fixes the general flow according to the systematic CE cycle. It then guides LLMs by assigning them subdivided CE operations within this flow. Hereafter, we define each LLM assigned a CE operation as an agent. Our *system* prepares prompt templates for each agent,¹ which include placeholders that can be dynamically filled with text. Therefore, once a user inputs the data, prompts for each agent are dynamically generated according to that data, and the internal agents autonomously complete the flow (i.e., CE cycle). To facilitate data processing within our *system*, all agents output JSON data. This is achieved by instructing agents in their input prompts to output text in JSON format, and then parsing the output text as JSON data. Our *system* uses the JSON output instruction and parser of LangChain (LangChain, 2023). Our *system* has 27 agents, 21 system prompts, 26 user prompts, and two AI prompts.² In the following sections, we describe the details of our *system*'s internal process from input to output, breaking it down into the five phases. See Appendix B for all our *system*'s prompt templates.

2.1 PHASE 0: PRE-PROCESSING

Given user inputs, our *system* first deploys the user's system to the K8s cluster by running the Skaffold configuration file. Then, the agents sequentially process the user inputs as follows:

- 1. Summarize each of the input K8s manifests separately.
- 2. Identify potential issues for resiliency and redundancy in the K8s manifests.
- 3. Assume a possible application of the K8s manifests.
 - 4. Summarize user instructions for the CE cycle if provided. At the same time, filter out suspicious prompts, e.g., jailbreak prompts.

This phase is for deploying the user's system and explicitly filling in the implicit context of the user's input. In the subsequent phases, this added context will also be provided as input.

 ¹Instead of simply appending previous data and agent outputs to the conversation history to create the next
 agent's prompt, we create a new conversation for each agent every time and embed the organized previous data
 and agent outputs within it. However, the verification loop, which will be discussed later, is an exception.

²As our *system* uses chat models, prompts with three different roles—system, human, and AI—are required.

162 2.2 PHASE 1: HYPOTHESIS

167

168

170

171

172

173

174

175

176

177

179

193

The *hypothesis* phase defines the system's resiliency for an assumed failure scenario. Following the principles of CE (Basiri et al., 2016), our *system* first defines steady states and then defines failure injections.

Steady-state definition Steady states are the expected, normal behaviors of a system. Each steady state is defined by a pair of a state value and a threshold, and a steady state is considered satisfied when the state value meets the threshold. Therefore, the state values must be measurable outputs of the system, such as the number of active resources, error rates, and response time. Given the pre-processed user inputs, the agents define steady states as follows:

- 1. Select measurable states critical to maintaining the system's application. If any weak configurations are identified from the K8s manifests, their related states are preferentially selected.
- 2. Select tools to inspect the states. K8s API and k6 (Grafana Labs, 2021) are supported. Then, write the corresponding inspection scripts and inspect the current (normal) values of the states in the system by running the scripts.
- 3. Define the thresholds for each state based on the inspected values (steady states must be satisfied under the current condition).
- 4. Write unit-test scripts to validate whether each steady state is satisfied by adding thresholdbased assertions to the corresponding inspection scripts.



Figure 2: Examples of unit-test scripts to validate steady states.

194 The unit-test scripts are used in the experiment workflow to automatically validate the steady states 195 during chaos experiments. We here call this unit-test-based validation approach Validation as Code 196 (VaC). Validating steady states by an LLM taking log data does not guarantee the consistency of the 197 validation process and may even result in incorrect judgments. On the other hand, with VaC, the validation process becomes fixed once a unit test is written, guaranteeing its consistency. Furthermore, 199 the explicit definition of the process in code enhances its transparency. Figure 2 shows examples of VaC scripts for K8s API (Python) and k6 (Javascript). k6 can collect communication metrics 200 (e.g., response times, error rates, etc) while conducting load tests. In VaC, k6 is used to inspect the communication metrics, while K8s API is used to inspect the other states of K8s resources. Both 202 scripts allow for adjusting test durations through command-line arguments. For k6, the script also 203 sets an appropriate number of virtual users for the load tests. 204

In steps 2 and 4, scripts are repeatedly debugged until they terminate successfully. In this verification loop, as an exception, our *system* simply appends the previous agent's output and the resulting error messages to the initial conversation as conversation history, and uses it as the agent's prompt in the next loop. The verification loops that appear later are similar.

Failure definition For failure injection, our *system* employs Chaos Mesh (Chaos Mesh, 2021), which can manage chaos experiments for K8s through code. Given the pre-processed user inputs and the steady states, the agents define failures that may occur in the system as follows:

213 1. Assume a failure scenario (e.g., a surge in access due to a promotional campaign, cyber attack,
214 etc.) that may occur in the system. Then, define the sequence of failures that simulates the
215 scenario and may affect the defined steady states. The failures are selected from the failure types supported in Chaos Mesh.

2. Define detailed parameters for each failure, such as the scope of the failure injection, the failure sub-type, the failure strength, etc.

In step 1, the agent outputs a 2D list of Chaos Mesh failure type names arranged in the order of insertion. The inner lists involve concurrent failures, and the outer list represents the injection order of each concurrent failure set. For example, [[StressChaos, NetworkChaos], [PodChaos]] represents that PodChaos is injected after simultaneously injecting StressChaos and NetworkChaos.

In step 2, the agent separately defines the detailed parameters of each failure. 224 Each failure type requires a different parameter set. Therefore, given a failure 225 type name, our system dynamically selects the corresponding JSON output 226 instruction. The agent then outputs the corresponding parameter set. Figure 227 3 shows an example of the parameter set of PodChaos. There are seven 228 failure types, the instructions of which are prepared in advance referring to 229 the Chaos Mesh documentation. The parameter sets are verified through 230 a verification loop, which repeatedly debugs them until their Chaos Mesh 231 manifests pass the kubectl apply --dry-run=server command. 232 The failure injection duration and more detailed injection timing are defined 233 in chaos experiment planning (see next section), along with the duration and timing for running the VaC scripts. 234



Figure 3: An example of detailed parameters.

At this point, the hypothesis is reinterpreted as *all VaC scripts pass, even when failure injections are performed.*

237 238

216

217

218

2.3 PHASE 2: (CHAOS) EXPERIMENT

The *experiment* phase plans a chaos experiment to validate the hypothesis and executes it.

240 241 242

250

251

252

253

254

239

Experiment planning To enable systematic planning, we divide a chaos experiment into three phases: pre-validation, failure-injection, and post-validation. In the pre-validation phase, VaC scripts are executed to ensure that the steady states are satisfied under normal conditions. In the failure-injection phase, fault injections are executed. If a steady state, such as response time, needs to be validated during fault injections, the corresponding VaC scripts are executed concurrently. In the post-validation phase, VaC scripts are executed to ensure that steady states have been recovered after the fault injections. Given the pre-processed user inputs and the hypothesis, the agents plan a chaos experiment by dividing it into these three phases as follows:

- 1. Determine the duration of each phase.
- 2. Determine the VaC scripts and failure injections to be executed in each phase. For each of them, specify the duration and grace period within a range that does not exceed the duration of the phase.
- 3. Summarize the timeline of the chaos experiment (i.e., the order of each node) in detail. This summary is referred to when analyzing the experiment results.

In step 2, the agent outputs a list of dictionaries (i.e., schedule list) separately for each phase, with each dictionary containing three keys: name, grace_period, and duration. The name is either a steady state name or a failure type name, and each corresponds one-to-one with the VaC script and failure injection defined in the hypothesis. The grace_period is the waiting time from the start of each phase until the execution of the VaC script or failure injection, allowing flexible adjustment of the execution timing. The duration is the execution period after the grace period.

262 Based on these schedule lists, our system configures the chaos experiment using the Chaos Mesh 263 workflow. This workflow supports three types of nodes: failure node to execute failure injection, 264 task node to execute VaC scripts, and suspend node to wait for a specified duration. By grouping 265 these nodes into a serial or parallel group, the nodes within the group are executed sequentially or in 266 parallel. Our *system* configures the schedule by grouping nodes and groups hierarchically as follows 267 (Figure 4): For each phase, it first creates failure and task nodes according to the schedule list. Next, it serially groups each node that has a grace period greater than zero with a suspend 268 node. This suspend node is placed before the grouped node and waits for the duration of the 269 grace period of that node. It then groups all the serial groups and remaining nodes in parallel in



Figure 4: Hierarchical grouping for implementing Figure 5: Reconfiguration process by the agent a complex chaos experiment plan in Chaos Mesh. and a file management algorithm.

each phase. Finally, it serially groups the parallel groups of each phase. Its Chaos Mesh workflow manifest is generated by formatting this hierarchical group.

286 Experiment replanning (within the improvement loop) Resource types and metadata defined in 287 the K8s manifests may be changed during the *improvement* phase. Therefore, replanning inspection 288 targets in VaC scripts and the scope of failure injections is required between the *improvement* phase and the next experiment execution. Given the original and reconfigured K8s manifests, as well as 289 the previous VaC scripts, the agent adjusts or retains the inspection-target specifications in the VaC 290 scripts. The inspection targets refer to resource specifications (line 11 and 12 in (a)), the request DNS 291 (line 13 in (b)), etc., in Figure 2. Given the original and reconfigured K8s manifests, as well as the 292 previous failure-injection scope, the agent adjusts or retains the scope for the reconfigured manifests. 293 The scope refers to the selector filed in Figure 3. These adjustments are also debugged through 294 the verification loop. After the adjustments, our system regenerates a new ChaosMesh workflow 295 manifest by replacing only the path of VaC scripts of task nodes and the selector field of 296 failure nodes with adjusted ones. Note that this replanning only makes minor adjustments to 297 reflect the changes in the K8s manifests, without altering the chaos experiment's original intent. 298

Experiment execution After a Chaos Mesh workflow manifest is (re-) generated, our *system* applies it to the K8s cluster. Once the workflow is deployed, the workflow (i.e., chaos experiment) will be executed automatically. Therefore, our *system* simply waits for the workflow to finish after that deployment.

303 304

305

280

281 282 283

284

285

2.4 PHASE 3: ANALYSIS

After the chaos experiment is finished, our *system* mechanically checks whether the VaC scripts have passed. If all of them have passed, that means the current system configurations (i.e., K8s manifests) already satisfy the hypothesis. Therefore, our *system* finishes the current CE cycle at this point and moves to the extra phase. If at least one has failed, our *system* moves to the next *improvement* phase after analyzing the experiment results. In this analysis, given the K8s manifests, the timeline of the chaos experiments, and the list of failed scripts with their logs, the agent identifies the cause of the fails and then generates a report containing the causes and recommended countermeasures.

313 314

2.5 PHASE 4: IMPROVEMENT

315 The *improvement* phase reconfigures the K8s manifests to satisfy the hypothesis. Given the K8s 316 manifests, the hypothesis, the experiment plan, and the improvement loop history, the agent recon-317 figures the K8s manifests so that all the VaC scripts pass in the chaos experiment. The improvement 318 loop history stores the history of the experiment results, their analysis reports, and their reconfigu-319 rations, within the improvement loop. The history suppresses the repetition of the same reconfigu-320 ration. There are three reconfiguration modes: create, delete, and replace. The agent first selects the 321 reconfiguration mode while specifying the file name, and then writes the reconfigured K8s manifest only for the create and replace modes. The file manager of our system then edits the folder from 322 the previous improvement loop (in the first improvement, it corresponds to the user's input folder) 323 according to the agent's output. Figure 5 illustrates these reconfiguration processes. The verification 324 loop is also conducted here: the agent's output is debugged repeatedly until all the K8s manifests in 325 the edited folder are correctly applied to the K8s cluster. 326

327 **Improvement loop** After the reconfiguration, our *system* applies the reconfigured K8s manifests 328 to the K8s cluster. Then, they will be validated again through the *experiment* and *analysis* phases. 329 That is, as in the systematic CE cycle, our system also repeats the experiment, analysis, improvement phases until the hypothesis is satisfied. We define this loop as the improvement loop. 330

331 332

333

2.6 EXTRA PHASE: POST-PROCESSING

After the CE cycle is completed, our system finalizes its entire process by summarizing the com-334 pleted CE cycle. The agent summarizes the user's input and each of the four completed phases. 335 Finally, our system provides the user with the summary of the completed CE cycle and the folder 336 containing K8s manifests that have been reconfigured to satisfy the hypothesis defined in the hy-337 *pothesis* phase and their Skaffold configuration file. 338

- 3 CASE STUDY
- 339 340 341

In this section, we validate the entire process of our system through case studies on two different 342 scale systems: NGINX and SOCKSHOP (Weaveworks, 2023). NGINX is a small-scale system that 343 consists of two K8s manifests (i.e., two resources): pod.yaml and service.yaml. The for-344 mer defines a Pod resource including a Nginx server, and the latter defines Service resource 345 routing TCP traffic to the Pod. To verify whether our system can improve the system when there 346 are resiliency issues, we intentionally configure the resource with a non-resilient setting; we set 347 restartPolicy to Never in Pod. yaml. With this configuration, once the Pod goes down, it 348 will never restart, resulting in extended service outages. On the other hand, SOCKSHOP is a prac-349 tical and large-scale e-commerce system that consists of 29 manifests, which define the resources 350 and databases for front-end pages, user information, order, payment, shipping, and so on. The number of replicas of all the Deployment resources is originally set to one. However, this setting 351 could lead to downtime of the single replica when it goes down. To narrow down this original 352 resiliency issue to a single point, we increase the replicas for Deployment resources other than 353 front-end-dep.yaml to two, while keeping a single replica for front-end-dep.yaml. 354 This RELATIVELY reduces the redundancy/resiliency of the front-end resource. In this case study, 355 we validate whether our system correctly identifies and addresses these resiliency issues through a 356 reasonable CE cycle. 357

Long-term experiments are not required for the resiliency issues here. Therefore, to save time, we 358 input the following instruction along with the K8s manifests: "Chaos-Engineering experiment must 359 be completed within 1 minute". For SOCKSHOP, we additionally instruct on how to access its 360 web page as follows: "When using k6 in steady-state definition, always select a request URL from 361 the following options (other requests are invalid): 1. http://front-end.sock-shop.svc.cluster.local/, 362 2. http://front-end.sock-shop.svc.cluster.local/detail.html?id=ID, ...". We use gpt-4o-2024-08-06 as 363 LLMs for our system. To improve the reproducibility of this case study, its temperature is set to 364 zero with the seed fixed at 42. We run a single CE cycle for each system five times under the same 365 settings.³ In the following, we first discuss the aggregated results obtained from multiple runs of 366 single CE cycles: the time and monetary costs, the completion rate, and the reconfiguration rate. 367 Then, we qualitatively validate the operations within the CE cycles conducted by our system. See also Appendix C for more details on the inputs and outputs for each system. 368

369

Time and monetary costs Table 1 shows our *system*'s time and monetary costs of single CE 370 cycles for each system. We first count the input and output tokens using the tokenizer of GPT-40 371 (o200k_base). We then calculate the monetary costs based on the official OpenAI API pricing 372 table in September 2024. For NGINX, our system completes single CE cycles in just \$0.21 and 373 11 minutes (including 2 minutes for the chaos experiment execution). Although we do not have 374 statistical data on the actual working time and labor costs for the same CE cycles performed by 375

³⁷⁶ ³Due to the non-deterministic nature of GPT-40, the outputs are not fully reproduced every time even when 377 the temperature is set to zero and a seed is provided. Therefore, the multiple runs aim to ensure the stability of our system under such randomness.

Table 1: Time and monetary costs of single CE cycles conducted by CHAOSEATER. The values for
each phase are averaged across runs that did not skip that phase, while the values for overall are
averaged across runs that involved system reconfiguration.

				Ngnix			
Metric	Overall	Pre-process	Hypothesis	Experiment	Analysis	Improvement	Post-prosess
Input tokens	59k	2.6k	25k	13k	4.4k	5.5k	8.2k
Output tokens	5.9k	0.5k	2.5k	1.7k	0.6k	0.2k	0.4k
API billing (\$)	0.21	0.01	0.09	0.05	0.02	0.02	0.02
Time	11m	21s	2.6m	4.4m	50s	12s	21s
		SockShop					
Metric	Overall	Pre-process	Hypothesis	Experiment	Analysis	Improvement	Post-prosess
Input tokens	284k	30k	150k	57k	14k	15k	18k
-							
Output tokens	13k	5.7k	3.8k	1.8k	0.7k	0.6k	0.5k
Output tokens API billing (\$)	13k 0.84	5.7k 0.13	3.8k 0.41	1.8k 0.16	0.7k 0.04	0.6k 0.04	0.5k 0.05

human engineers, these total operational time and monetary costs are obviously lower than that. For SOCKSHOP, the monetary cost increases by approximately four times (\$0.84), and the time doubled (25m). However, these values are still intuitively lower than those of human engineers. Even with the number of resources increasing by more than ten times compared to NGINX, the cost increase remains minimal, demonstrating that our *system* can maintain low costs even for large-scale systems.

402 **Completion rate and reconfiguration rate** 403 Table 2 shows the completion rate and recon-404 figuration rate. The former refers to the per-405 centage of runs where our system successfully 406 completes the CE cycle without runtime er-407 rors (e.g., the verification loop or improvement 408 loop reaching the maximum limit of three itera-409 tions), while the latter represents the percentage of runs where our system not only completes the 410

Table 2: Completion rate and reconfiguration rate in five runs of single CE cycles for each system.

System	Completion (%)	Reconfig (%)
NGINX	100	100
SockShop	100	80

CE cycle without runtime errors but also successfully reconfigures the input system. The 100% com-411 pletion rates reported in Table 2 demonstrate the stability of our system across both systems. Our 412 system also achieves the 100% reconfiguration rate for NGINX; it consistently reconfigures the sys-413 tem using a Deployment with multiple replicas, which addresses the resiliency issue of never 414 restarting policy. For SOCKSHOP, in four out of five runs, our system reconfigures the system by 415 setting the number of replicas in the front-end Deployment to two or more. This addresses the 416 downtime issue of the single replica. In other run, our system completes a CE cycle skipping the sys-417 tem reconfiguration. In this case, as the steady states are only checked before and after the pod-kill 418 (i.e., in the pre/post-validation phase), the front-end replica has already recovered by the time of the check, and the downtime issue is not detected. Although this is not the outcome we expected, it is 419 still a valid CE cycle that verifies whether the target Pod can recover properly after a certain period 420 of time. Overall, our system stably completes single CE cycles for both NGINX and SOCKSHOP 421 without any critical issues. 422

423

382

397

398

399

400

401

Qualitative validation on NGINX Here, we pick up one of the five runs for NGINX and qualitatively validate its outputs at each representative phase. The top of Figure 6 shows the highlighted
outputs for NGINX. In the *hypothesis* phase, our *system* first defines two steady states: 1) "The Pod
should be running at least 90% of the time during the check period"; 2) "Service availability should
be at least 99.9% with a response status of 200". The VaC scripts shown in Figure 6 correctly implement these steady states. It then defines a failure sequence that injects NetworkChaos (delay)
into the Nginx Pod following PodChaos (pod-kill) to simulate a cyberattack.

431 In the experiment planning, it plans the following chaos experiment: 1) the two steady states are sequentially validated in the pre-validation phase; 2) in the failure-injection phase, the two steady



Figure 6: The highlighted outputs for NGINX and SOCKSHOP

states are sequentially validated alongside the injection of each failure that may affect them; 3) the two steady states are validated sequentially once again in the post-validation phase.

The first chaos experiment reveals that the two steady states are currently not satisfied in both the failure-injection and post-validation phases. Our *system* successfully identifies the cause as the Nginx Pod's never restarting policy and replaces it with a Deployment with three replicas. It then adjusts the success criteria of the VaC script to ensure that at least one Pod out of three replicas is running and conducts the chaos experiment for the reconfigured system again. Finally, the additional chaos experiment confirms that the hypothesis is satisfied in the reconfigured system.

Intuitively, these operations and outputs follow best practices, demonstrating that our *system* can complete our expected CE cycle for the small-scale system, NGINX, without explicit user instructions.

468

454 455 456

457

458

Qualitative validation on SOCKSHOP Similarly, we qualitatively validate the outputs of a run for SOCKSHOP. The bottom of Figure 6 shows the highlighted outputs for SOCKSHOP. In the hypothesis phase, our system first defines two steady states: 1) "At least 1 ready replica 100% of the time and 2 ready replicas at least 80% of the time during the monitoring period" for the carts Deployment; 2) "At least 1 ready replica must be present 100% of the time during the monitoring period" for the front-end Deployment. The VaC scripts shown in Figure 6 correctly implement these steady states.

- It then defines a failure sequence that injects StressChaos (CPU) into all the carts-db replicas following PodChaos (pod-kill) that targets the single front-end replica to simulate possible problems in a black Friday sale.
- In the experiment planning, it plans the following chaos experiment: 1) the two steady states are simultaneously validated in the pre-validation phase; 2) in the failure-injection phase, the two steady states are sequentially validated alongside the injection of each failure that may affect them; 3) the two steady states are validated simultaneously once again in the post-validation phase.
- The first chaos experiment reveals that the second steady state is currently not satisfied in both the failure-injection and post-validation phases. Our *system* successfully identifies the cause as the single replica setting of the front-end Deployment and increases the number of replicas to two. It then conducts the chaos experiment for the reconfigured system again without experiment adjust-

ments (no adjustment is needed as the resource type remains unchanged). Finally, the additional chaos experiment confirms that the hypothesis is satisfied in the reconfigured system.

This case also broadly follows best practices, demonstrating that our *system* can complete our expected CE cycle even for the comparatively large system, SOCKSHOP, without explicit user instructions.

491 492 493

4 DISCUSSION

494 495

Broader impacts Numerous systems, including the increasing number of LLM applications in 496 recent years, are built in the microservice architecture, and their number is expected to continue to 497 grow in the future. By fully automating CE, it will be possible for anyone to build resilient systems. 498 Moreover, it is expected to combine our system with other LLM systems, such as improving the 499 resiliency of applications created by other LLM systems through our system. Although our system 500 is not yet at a level that can practically realize that, we believe that our system is a good starting 501 point toward such use cases. Even at its current level, our system can be sufficiently used as training 502 materials (including both good and bad practices) for the Chaos Game Day, which is a training 503 exercise for CE engineers.

504 505

Limitations Our system currently has three limitations: 1) Limited deployment environment; al-506 though CE should ideally be conducted in actual production environments, our system is currently 507 only supported in development environments. 2) Limited to GPT-40 only; our system's prompt tem-508 plates are highly tuned only for GPT-40. Therefore, other LLMs can not currently be used for our 509 system. 3) Vulnerability discovery; in the case study, our system improved a system with relatively 510 simple resiliency issues. However, for systems that already possess a certain level of resiliency, our 511 system fails to find new issues through a CE cycle. Given that this is a challenging task even for 512 engineers, our system is currently considered to perform at a level comparable to, or lower than, that 513 of engineers. To find issues in such systems, it is necessary to conduct multiple CE cycles for more 514 complex systems over extended operational periods.

515 516

Future directions Given the current limitations above, we share four future directions for our sys-517 tem: 1) Fully automation of long-term multiple CE cycles; By using the system's output as input for 518 the next CE cycle, we can automate multiple CE cycles even with our current system. However, we 519 additionally need to develop techniques to manage the long-term history of completed CE cycles and 520 continuous learning (if LLMs are fine-tuned). 2) Support for various LLMs; As our system's prompt 521 templates are tuned manually, supporting various LLMs significantly increases their management 522 costs. To address this, automatic prompt tuning is considered an effective solution. 3) Fine-tuning 523 LLMs specifically for CE; We may leverage our system's outputs as the instruction-tuning data. 4) 524 Production deployment and security; if our *system* is deployed on production environments, fur-525 ther research on security will be necessary. This includes controlling the impact range of failures, 526 preventing our system from becoming a proxy for attacking production services, and proposing emergency response measures (e.g., a higher-level monitoring system that monitors our system). 5) 527 Improvement for more complex systems; We need to incorporate the recent advances in LLMs x 528 graph approaches to extract necessary sub-graphs from large system graphs. This sub-graph extrac-529 tion is important to organize the agent's inputs in each phase. 530

- 531 532
- 5 CONCLUSION

533 534

In this paper, we proposed CHAOSEATER, a system for automating the entire CE workflow with
LLMs. We presented the technical details of our *system* and validated it through a case study.
The results demonstrated that our *system* successfully reduces the time and monetary costs while
completing a reasonable CE cycle. In future work, we will improve our *system* following the future
directions discussed above. We are also excited that other researchers and developers will propose
related works in the same or different directions.

540	REFERENCES
541	REI EREITEEB

581

582

583 584

585

586

587 588

589

590

Tasnim Ahmed, Nicola Piovesan, Antonio De Domenico, and Salimur Choudhury. Linguistic intel ligence in large language models for telecommunications. 2024.

- Peter Alvaro, Kolton Andrus, Chris Sanden, Casey Rosenthal, Ali Basiri, and Lorin Hochstein. Automating failure testing research at internet scale. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, SoCC '16, pp. 17–28, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345255. doi: 10.1145/2987550.2987555. URL https://doi.org/10.1145/2987550.2987555.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021.
- Lina Bariah, Hang Zou, Qiyang Zhao, Belkacem Mouhouche, Faouzi Bader, and Merouane Debbah.
 Understanding telecom language through large language models, 2023.
- Ali Basiri, Niosha Behnam, Ruud de Rooij, Lorin Hochstein, Luke Kosewski, Justin Reynolds, and Casey Rosenthal. Chaos engineering. *IEEE Software*, 33(3):35–41, 2016. doi: 10.1109/MS.2016. 60.
- Ali Basiri, Lorin Hochstein, Nora Jones, and Haley Tucker. Automating chaos experiments in pro duction. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software
 Engineering in Practice (ICSE-SEIP), pp. 31–40, 2019. doi: 10.1109/ICSE-SEIP.2019.00012.
- Antonio Bucchiarone, Nicola Dragoni, Schahram Dustdar, Patricia Lago, Manuel Mazzara, Victor Rivera, and Andrey Sadovykh (eds.). *Microservices, Science and Engineering*. Springer, 2020. ISBN 978-3-030-31646-4. doi: 10.1007/978-3-030-31646-4. URL https://doi.org/10. 1007/978-3-030-31646-4.
- Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests, 2022.

569 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared 570 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, 571 Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, 572 Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fo-573 tios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex 574 Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, 575 Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec 576 Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob Mc-577 Grew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large 578 language models trained on code, 2021. 579

- Suman De. A study on chaos engineering for improving cloud software quality and reliability. In 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), volume 1, pp. 289–294, 2021. doi: 10.1109/CENTCON52345.2021. 9688292.
- Firefly. Aiac (github repository), 2022. URL https://github.com/gofireflyio/aiac.
- Google. Skaffold (github repository), 2019. URL https://github.com/ GoogleContainerTools/skaffold.
- Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming the rise of code intelligence, 2024.
- Yudong Huang, Hongyang Du, Xinyuan Zhang, Dusit Niyato, Jiawen Kang, Zehui Xiong, Shuo
 Wang, and Tao Huang. Large language models for networking: Applications, enabling techniques, and challenges. 2023.

- 594 Hiroki Ikeuchi, Akio Watanabe, and Yousuke Takahashi. Coverage based failure injection toward 595 efficient chaos engineering. In ICC 2023 - IEEE International Conference on Communications, 596 pp. 4571-4577, 2023. doi: 10.1109/ICC45041.2023.10279387. 597 Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language 598 models for code generation, 2024. URL https://arxiv.org/abs/2406.00515. 600 Shuyang Jiang, Yuhao Wang, and Yu Wang. Selfevolve: A code evolution framework via large 601 language models, 2023. 602 K8sGPT. K8sgpt (github repository), 2023. URL https://github.com/k8sgpt-ai/ 603 k8sqpt. 604 605 Imtiaz Karim, Kazi Samin Mubasshir, Mirza Masfiqur Rahman, and Elisa Bertino. SPEC5G: 606 A dataset for 5G cellular network protocol analysis. In Jong C. Park, Yuki Arase, Baotian 607 Hu, Wei Lu, Derry Wijaya, Ayu Purwarianti, and Adila Alfa Krisnadhi (eds.), Findings of 608 the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings), pp. 20-38, Nusa Dua, Bali, November 2023. Association for Computational Linguistics. URL https: 609 //aclanthology.org/2023.findings-ijcnlp.3. 610 611 Dominik Kesim, André van Hoorn, Sebastian Frank, and Matthias Haussler. Identifying and prior-612 itizing chaos experiments by using established risk analysis techniques. pp. 229–240, 10 2020. 613 doi: 10.1109/ISSRE5003.2020.00030. 614 Kubernetes (github repository), 2014. URL https://github.com/ Kubernetes. 615 kubernetes/kubernetes. 616 617 LangChain. Langchain (github repository), 2023. URL https://github.com/ 618 langchain-ai/langchain. 619 Ali Maatouk, Fadhel Ayed, Nicola Piovesan, Antonio De Domenico, Merouane Debbah, and Zhi-620 Quan Luo. Telequa: A benchmark dataset to assess large language models telecommunications 621 knowledge, 2023. 622 623 Yukai Miao, Yu Bai, Li Chen, Dan Li, Haifeng Sun, Xizheng Wang, Ziqiu Luo, Yanyu Ren, Dapeng 624 Sun, Xiuting Xu, Qi Zhang, Chao Xiang, and Xinchi Li. An empirical study of netops capability 625 of pre-trained large language models, 2023. 626 Microsoft. Azure chaos studio (product page), 2023. URL https://azure.microsoft. 627 com/products/chaos-studio/. 628 629 Netflix. Chaos monkey (github repository), 2012. URL https://github.com/Netflix/ 630 chaosmonkey. 631 Nicola Piovesan, Antonio De Domenico, and Fadhel Ayed. Telecom language models: Must they 632 be large?, 2024. 633 634 Prometheus (github repository), 2012. URL https://github.com/ Prometheus. 635 prometheus/prometheus. 636 Pulumi. Pulumi ai (github repository), 2023. URL https://github.com/pulumi/ 637 pulumi-ai. 638 639 Robusta. Kubernetes chatgpt bot (github repository), 2023. URL https://github.com/ robusta-dev/kubernetes-chatgpt-bot. 640 641 Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi 642 Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Ev-643 timov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, 644 Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, 645 Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. 646
- 647 Amazon Web Services. Aws fault injection service (product page), 2021. URL https://aws. amazon.com/fis/.

- 648 Chaos Mesh. Chaos mesh (github repository), 2021. URL https://github.com/ 649 chaos-mesh/chaos-mesh/. 650
- Cognition Labs. Introducing devin, the first ai software engineer (official blog), 2024. URL https: 651 //www.cognition.ai/blog/introducing-devin. 652
- 653 Grafana Labs. k6 (github repository), 2021. URL https://github.com/grafana/k6.
- 654 Kennedy A. Torkura, Muhammad I.H. Sukmana, Feng Cheng, and Christoph Meinel. Security chaos 655 engineering for cloud services: Work in progress. In 2019 IEEE 18th International Symposium on 656 *Network Computing and Applications (NCA)*, pp. 1–3, 2019. doi: 10.1109/NCA.2019.8935046. 657
- Weaveworks. Sock shop (github repository (archived in 2023)), 2023. URL https://github. com/microservices-demo/microservices-demo. 659
- 660 Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-661 based software engineering agents, 2024. URL https://arxiv.org/abs/2407.01489.
- 662 John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, 663 and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 664 2024. URL https://arxiv.org/abs/2405.15793. 665
 - Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. 2023.
 - Sertaç Ozercan. Kubectl ai (github repository), 2023. URL https://github.com/ sozercan/kubectl-ai.
- 671 672 673

682

666

667

668

669 670

658

Α RELATED WORK

675 Chaos Engineering Since the paper (Basiri et al., 2016) introduced its name, CE has gained at-676 tention and is now employed in various services (Basiri et al., 2019; De, 2021). Several works have 677 advanced CE from the perspective of optimization of failure injections (Ikeuchi et al., 2023; Alvaro 678 et al., 2016), risk analysis (Kesim et al., 2020), and security (Torkura et al., 2019). In application, 679 various automation tools have been developed from both the open-source (Chaos Mesh, 2021; Netflix, 2012) and commercial sectors(Amazon Web Services, 2021; Microsoft, 2023). While partial 680 automation of CE has progressed, full automation has not yet been explored. 681

LLMs for software engineering LLMs for coding have been explored from various aspects: Pre-683 training models (Chen et al., 2021; Rozière et al., 2024; Guo et al., 2024), prompting (Chen et al., 684 2022; Jiang et al., 2023), and evaluation (Austin et al., 2021; Chen et al., 2021). For more general SE 685 tasks, LLMs that solve issues in a repository have also emerged (Yang et al., 2024; Cognition Labs, 686 2024; Xia et al., 2024). 687

688 LLMs for networking (NW) LLMs for NW have been explored from various aspects in recent 689 years: Datasets (Bariah et al., 2023; Karim et al., 2023), benchmarks (Maatouk et al., 2023; Miao 690 et al., 2023), fine-tuned models (Bariah et al., 2023), an agent framework for NW-related tasks 691 (Huang et al., 2023), and comprehensive evaluation (Ahmed et al., 2024; Piovesan et al., 2024). 692 These works empirically demonstrate the promise of applying LLMs to the NW domain. In par-693 allel with the research side, applications have also been developed, especially for IaC. They range from LLM-based code generation (Firefly, 2022; Özercan, 2023; Pulumi, 2023) to diagnostic tools 694 (K8sGPT, 2023; Robusta, 2023). 695

696 697

698

700

В IMPLEMENTATION DETAILS

699 **B.1** SYSTEM PROMPTS

In this section, we share all prompt templates for LLM agents of CHAOSEATER. Words enclosed 701 in blue curly braces {} denote placeholders that change dynamically based on user input. These enclosed words are variable names, and identical variable names will have the same text embedded.
Examples of the text embedded in each placeholder are also shared under each system prompt. On
the other hand, words enclosed in red curly braces {} denote placeholders that are replaced with
dynamic templates. These pre-defined templates are dynamically retrieved and embedded in the
placeholders according to the situation.

B.1.1 PRE-PROCESSING

707 708

709 710

711 712

713

714

715

716

717

718

719 720

721

722

723

724

725

726 727

728

729

730 731

732

733 734 735

736 737

738

739

740

741

742

743

744

745

0-0: Agent for drafting a steady state System: System: You are a professional Kubernetes (k8s) engineer. Given a K8s manifest, please summarize it according to the following rules: - The summary must be written in bullet points. - Summarize the functions of the K8s manifest in a way that is understandable to even beginners. - The output should be formatted as a JSON instance that conforms to the JSON schema below. As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \" description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \" string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\", \"baz\"]}} is not well-formatted. Here is the output schema: {\"properties\": {\"k8s_summary\": {\"title\": \"K8S Summary\", \"description\": \" Summary of the K8s manifest. Summarize it in bullet points like '- the 1st line\\n- the second line...'\", \"type\": \"string\"}}, \"required\": [\"k8s_summary\"]} Human: # K8s manifest {k8s vaml} Please summarize the above K8s manifest. AI: ''json {\"k8s_summary\":

Example text embedded to **k8s_yam1**

```nginx/pod.yaml
apiVersion: v1
kind: Pod
metadata:
 name: example-pod
 labels:
 app: example
spec:
 restartPolicy: Never
 containers:
 - name: example-container
 image: nginx:1.17.1
 ports:
 - containerPort: 80
```

746 747 748

749

750 751 752

0-1: Agent for finding potential weaknesses

System:

```
You are a professional Kubernetes (K8s) engineer.
Given K8s manifests for a system, you will identify their potential issues for
resiliency and redundancy when failures occur in the system.
Always keep the following rules:
```

756	
757	- List each issue with its name, associated K8s manifest(s), potential issues due to
758	improvements).
759	- If the same issue exists in different manifests, merge them into a single issue,
760	- The output should be formatted as a JSON instance that conforms to the JSON schema
761	below.
762	As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
763	<pre>description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \"</pre>
764	<pre>string\"}}, \"required\": [\"IOO\"]}\nthe object {\"IOO\": [\"bar\", \"baz\"]} is a well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\",</pre>
765	<pre>\"baz\"]}} is not well-formatted.</pre>
766	Here is the output schema:
767	
768	<pre>with its name, potential issues due to fault injection, and manifest configuration</pre>
769	causing the issues (no need to suggest improvements).\", \"type\": \"array\", \"items
770	<pre>\": {\"\$ref\": \"#/definitions/Kosissue("}}}, \"required\": [\"issues\"], \"definitions \": {\"K8sIssue\": {\"title\": \"K8sIssue\", \"type\": \"object\", \"properties\": {\"</pre>
771	<pre>issue_name\": {\"title\": \"Issue Name\", \"description\": \"Issue name\", \"type\": \"</pre>
772	potential issues due to fault injection/", \"type\": \"string\"}, \"manifests\": {\"
773	title\": \"Manifests\", \"description\": \"manifest names having the issues\", \"type
774	<pre>\"Problematic Config\", \"description\": \"problematic configuration causing the issues</pre>
775	<pre>(no need to suggest improvements).\", \"type\": \"string\"}}, \"required\": [\" icsue name\", \"icsue details\", \"manifests\", \"problematic config"]}}}</pre>
776	issue_name(, (issue_uecalis(, (manifests(, (problematic_config())))
777	Human:
778	<pre># here are the kos manifests for my system. {k8s_yamls}</pre>
779	Plazas list issues for each Was manifest
780	Please list issues for each kas manifest.
781	AI:
782	{\"issues\":
783	
784	

Example text embedded to **k8s_yamls**

```nginx/pod.vaml
apiVersion: v1
kind: Pod
metadata:
name: example-pod
labels.
app: example
spec.
restartPolicy: Never
containers.
- name: example-container
- Hame, example-concarner
Image. Ingilix.I.I/.I
ports:
- concarnerPort: 80
```nginx/service.yaml
apiVersion: v1
kind: Service
metadata:
name: example-service
spec:
selector:
app: example
norte:
- protocol, TCP
- prococor: ICP
port: 80
targetPort: 80

810 # 0-2: Agent for assuming an application 811 812 System: 813 You are a professional Kubernetes (k8s) engineer. 814 Given k8s manifests and dependencies between them, please assume a real-world application (service) of the manifests according to the following rules: 815 - If the application is explicitly specified in the instructions, assume it. - You can leverage any given information, including file name, manifests, and 816 dependencies, to guess the purpose of the manifests. 817 - The output should be formatted as a JSON instance that conforms to the JSON schema 818 below. 819 As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \" description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \" string\"}}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a 820 821 well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\", 822 \"baz\"]}} is not well-formatted. 823 Here is the output schema: 824 {\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Before assuming an application, reason logically why you assume it for the given manifests. e. 825 g., from file name, instructions, or other elements?\", \"type\": \"string\"}, \"
k8s_application\": {\"title\": \"K8S Application\", \"description\": \"Specify what the
service (application) offers to users.\", \"type\": \"string\"}, \"required\": [\"
thought\", \"k8s_application\"]} 826 827 828 829 830 Human: {user_input} 831 832 Please assume a real-world application of the manifests. 833 AI: ''json 834 835 836 837 Example text embedded to **user_input** 838 839 # K8s manifest: ```nginx/pod.yaml 840 apiVersion: v1 841 kind: Pod 842 metadata: name: example-pod 843 labels: 844 app: example spec: 845 restartPolicy: Never 846 containers: - name: example-container 847 image: nginx:1.17.1 848 ports: containerPort: 80 849 • • • 850 # Summary of nginx/pod.yaml: - This manifest defines a Kubernetes Pod. 851 - The Pod is named 'example-pod'. 852 - It includes metadata with a label 'app: example'. - The Pod's restart policy is set to 'Never', meaning it won't restart automatically if 853 it fails. 854 - The Pod contains one container named 'example-container'. - The container uses the 'nginx:1.17.1' image. 855 - The container exposes port 80, which is typically used for HTTP traffic. 856 # K8s manifest: 857 ```nginx/service.yaml 858 apiVersion: v1 kind: Service 859 metadata: 860 name: example-service spec: 861 selector: 862 app: example ports: 863

864	
865	- protocol: TCP
866	port: 80 targetPort: 80
867	
868	# Summary of nginx/service.yaml: - This manifest defines a Kubernetes Service.
869	- The Service is named 'example-service'.
870	- It uses the 'v1' API version. - The Service selects pods with the label 'app: example'.
871	- It exposes the Service on port 80 using the TCP protocol.
872	- The target port for the Service is also set to 80, meaning it forwards traffic to port 80 on the selected pods.
873	

#0-3: Agent for summarizing user instructions

878	System:
879	You are a professional Chaos Engineering practitioner.
880	Chaos Engineering is an engineering technique aimed at improving the resiliency of distributed systems. It involves artificially injecting specific failures into a
881	distributed system and observing its behavior in response. Based on the observation,
882	the system can be proactively improved to handle those failures.
883	new insights into the system through Chaos-Engineering experiments.\nSystematically,
884	Chaos Engineering cycles through four phases: hypothesis, experiment, analysis, and
885	1) Hypothesis: Define steady states (i.e., normal behavior) of the system and
886	injected failures (i.e., faults). Then, make a hypothesis that \u201cthe steady
887	2) Experiment: Inject the failures into the system and monitor/log the system's
888	behavior in response.
880	3) Analysis: Analyze the logged data and check if the hypothesis is satisfied. If so, one CE cycle is finished here. If not, move to (4)
890	4) Improvement: Reconfigure the system to satisfy the hypothesis. The reconfigured
801	system is tested again in (2) and (3), i.e., repeat (2) to (4) until the hypothesis is is satisfied.
892	
803	Given user instructions for the Chaos Engineering, please filter out obviously irrelevant instructions according to the following rules:
000	- Organize the instructions in bullet points.
094	- For relevant instructions, just copy it to avoid changing any user intents.\n- Ignore instructions irrelevant obviously to the Chaos-Engineering, such as jailbreaking
090	prompts.
896	- For those that are evident, explain in which phase (our entire cycle) each
897	- If you are unsure whether something is related or not, include it in the output.
898	- The output should be formatted as a JSON instance that conforms to the JSON schema
899	Delow.
900	As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
901 902	<pre>description\": \"a list of strings\", \"type\": \"array\", \"ltems\": {\"type\": \" string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a well-formatted instance of the schema. The object {\"properties\": {\"bar\".</pre>
903	<pre>\"baz\"]}} is not well-formatted.</pre>
904	Hara is the output scheme.
905	inere is the output schema.
906	{\"properties\": {\"ce_instructions\": {\"title\": \"Ce Instructions\", \"description
907	in bullet points like - summary of instruction #1\\n- summary of instructions #2\\n-
908	<pre>\", \"type\": \"string\"}}, \"required\": [\"ce_instructions\"]}</pre>
909	
910	Human:
011	<pre># Instructions {ce instructions}</pre>
912	
013	Please filter out the above instructions for the CE.
01/	AI:
015	```json {\"ce instructions\":
016	
210	

Example text embedded to **ce_instructions**

The Chaos-Engineering experiment must be completed within 1 minute.

B.1.2 Hypothesis

# 1-0: A	Agent for drafting a steady state
System:	
You are	a helpful AI assistant for Chaos Engineering.
steady	states (i.e., normal behaviors) that are related to potential issues of t
stem.	
Always	keep the following rules:
resourc	e that is easiest to encounter issues when certain failures occur.
- Consi	der whether a new steady state needs to be added, and if so, add a steady
If not	, indicate the end of the steady-state addition with 'exits=True'.
verify	through Chaos Engineering whether it's truly a problem later.
- An ad	ded steady state must be a measurable output, such as the number of pods,
through	put, error rates, latency percentiles, etc.
having	potential issues for resiliency and redundancy.
- An ad	ded steady state must be different from the already defined ones.
- The o	utput should be formatted as a JSON instance that conforms to the JSON so
DCTOW.	
As an e	<pre>xample, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"</pre>
descrip	tion\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": "}}}. \"required\": [\"foo\"]}\nthe object {\"foo\". [\"bar\" \"baz\"]}
well-fo	<pre>rmatted instance of the schema. The object {\"properties\": {\"foo\": [\"</pre>
\"baz\"]}} is not well-formatted.
Here is	the output schema:
{\"prop	erties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Descr
manifes	t) that is easiest to encounter the issues. Describe also the details of
steady	<pre>state itself.\", \"type\": \"string\"}, \"exits\": {\"title\": \"Exits\",</pre>
descrip	tion\": \"Whether to stop adding a new steady state or not. If you stop h
boolean	<pre>\"}, \"manifest\": {\"title\": \"Manifest\", \"description\": \"The targe</pre>
-manife	st name. Specify a SINGLE manifest.\", \"type\": \"string\"}, \"name\":
\": \"N	ame\", \"description\": \"Steady state name including the target K8s reso t) name. Please write it using a-z. A-Z. and O-9 \". \"type\": \"string\"
require	<pre>d\": [\"thought\", \"exits\", \"manifest\", \"name\"]}</pre>
Human:	
# Here	is the overview of my system:
{user_i	nput2}
# Pleas	e follow the instructions below regarding Chaos Engineering:
{ce_ins	tructions}
# Stead	v states already defined are the following.
{predef	<pre>ined_steady_states;</pre>
λ £+	anaidaming whathan a new standy state and to be added define the
that is	onsidering whether a new steady state needs to be added, define a steady different from the already defined steady states, if necessary
2.1.4.0 10	arroad, aorrhoa becau, beaces, ir heeebary.
AI:	
{\"thou	ght\":
Exampl	e text embedded to user input 2
Examp	
# Tho c	veter consists of the following K9s manifest(s):

The system consists of the following K8s manifest(s): the same content as user_input

972	
973	
974	# The resiliency issues/weaknesses in the system are as follows:
975	- details: The Pod will not restart automatically if it fails, which can lead to
976	downtime.
977	- problematic config: restartPolicy: Never
978	Torus #1. Circula Ded Deployment
979	- details: Using a single Pod without a controller like Deployment or ReplicaSet can
980	lead to lack of redundancy and no automatic recovery if the Pod fails.
981	<pre>- manifests having the issues: ['nginx/pod.yaml'] - problematic config: kind: Pod</pre>
082	
092	# The expected type of application on the system (i.e., K8s manifests): Web server application using Nginx to serve HTTP content.: The manifests provided
903	define a Pod and a Service in Kubernetes, both related to an application labeled '
904 005	example'. The Pod runs an Nginx container, which is a popular web server and reverse
900	default port for HTTP traffic. Given these details, it is logical to assume that the
986	application is a simple web server or a basic web application, as Nginx is commonly
987	assumption.
988	
989	
990	# 1-1: Agent for defining an inspection strategy
991	
992	
993	System: You are a helpful AI assistant for Chaos Engineering.
994	Given Kubernetes (K8s) manifests for a network system and its state type, you will
995	inspect the current value of the state type.
996	- You can use either K8s API (Python) or k6 (Javascript) to inspect the state.
997	- Use the K8s API for checking the current state of K8s resources
998	time, latency, etc.
999	- If you use K8s API, consider appropriate test duration. If you use k6, consider not
1000	load test.
1001	- Pay attention to namespace specification. If the namespace is specified in the
1002	manifest, it is deployed with the namespace. If not, it is deployed with the 'default' namespace.
1003	- When sending requests to a K8s resources, use their internal DNS names in the format:
1004	'''service-name.namespace.svc.cluster.local:port'''. For the port setting, use the service port, not the targetPort or nodePort. Ensure that the port matches the service
1005	port defined in the manifest.
1006	- The output should be formatted as a JSON instance that conforms to the JSON schema
1007	DETOM.
1008	As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
1009	<pre>string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a</pre>
1010	<pre>well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\", \"bac\"lll is not well-formatted</pre>
1011	(baz())) is not well-ionnalted.
1012	Here is the output schema:
1012	{\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Describe
1017	your thoughts for the tool usage. e.g., the reason why you choose the tool and how to
1014	use.\", \"type\": \"string\"}, \"tool_type\": {\"title\": \"Tool Type\", \"description \": \"Tool to inspect the steady state. Select from ['k8s', 'k6'].\", \"enum\": [\"k8s
1010	\", \"k6\"], \"type\": \"string\"}, \"tool\": {\"title\": \"Tool\", \"description\": \"
1010	If tool_type='k8s', write here K8sAPI. If tool_type='k6', write here K6JS.\", \"anyOf
1017	required\": [\"thought\", \"tool_type\", \"tool\"], \"definitions\": {\"K8sAPI\": {\"
1018	<pre>title\": \"K8sAPI\", \"type\": \"object\", \"properties\": {\"duration\": {\"title\": \"Duration\", \"Duration of the status that a status of the s</pre>
1019	loop. Set appropriate duration to check the current state of the system. The maximum
1020	<pre>duration is 5s.\", \"type\": \"string\"}, \"script\": {\"title\": \"Script\", \"</pre>
1021	of a K8s resource. Write only the content of the code, and for dictionary values.
1022	enclose them within a pair of single double quotes (\\\"). Implement a for loop that
1023	checks the status every second for the duration, and prints a summary of the results at the end.\\n- To support docker env, please configure the client as follows: ```\\n#
1024	Load Kubernetes configuration based on the environment\\n if os.getenv('
1025	KUBERNETES_SERVICE_HOST'):\\n config.load_incluster_config()\\n else:\\n

```
1026
                   config.load_kube_config()\\n```\\n- Please add an entry point at the bottom to
1027
            allow the test to be run from the command line.
\\n- Please add argparse '--duration' (
1028
            type=int) so that users can specify the loop duration.\", "type\": \"string\"}}, \"
            required\": [\"duration\", \"script\"]}, \"KGS\": {\"title\": \"KGS\", \"type\": \"
object\", \"properties\": {\"vus\": {\"title\": \"Vus\", \"description\": \"The number
1029
1030
            of virtual users. You can run a load test with the number of virtual users. \", \"type
            \": \"integer\"}, \"duration\": {\"title\": \"Duration\", \"description\": \"Duration
1031
            of the load test. Set appropriate duration to check the current state of the system.
1032
            The maximum duration is 5s.\", \"type\": \"string\"}, \"script\": {\"title\": \"Script
            \", \"description\": \"k6 javascript to inspect the current state. Write only the
1033
            content of the code, and for dictionary values, enclose them within a pair of single
1034
            double quotes (\\\"). In options in the javascript, set the same 'vus' and 'duration'
            options as the above. The interval of status check must be 1s second(s). Set a
1035
            threshold that triggers an error when a request failure is clearly occurring.\", \"type
            \": \"string\"}, \"required\": [\"vus\", \"duration\", \"script\"]}}
1036
1037
1038
            Human:
            # Here is the overview of my system:
1039
            {user_input2}
1040
            # You will inspect the following steady state in my system:
1041
            {steady_state_name}: {steady_state_thought}
1042
            # Please follow the instructions below regarding Chaos Engineering:
1043
            {ce instructions}
1044
            Please define the way to inspect "{steady_state_name}" in the system defined by the
1045
            above k8s manifest(s).
1046
            AT:
1047
             ''json
1048
            {\"thought\":"
1049
1050
             ----- In the verification loop, the prompts below will be stacked as history -----
1051
1052
            AI:
            {output}
1053
1054
            Human:
            Your current inspection script causes errors when conducted.
1055
            The error message is as follows:
1056
            {error_message}
1057
            Please analyze the reason why the errors occur, then fix the errors.
1058
            Always keep the following rules:
            - NEVER repeat the same fixes that have been made in the past.
1059
            - Fix only the parts related to the errors without changing the original content.
            - If requests failed, double-check if the service port is correct.
            - You can change the tool (k8s -> k6 or k6 -> k8s) if it can keep the original
1061
            intention.
1062
             {format instructions}
1063
            AI:
1064
             ```json
 {\"thought\":"
1065
1066
1067
 Example text embedded to steady_state_name
1068
1069
1070
 example-pod-running-state
1071
1072
 Example text embedded to steady_state_thought
1073
1074
1075
 The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod
 .yaml' manifest. This is a critical issue because if the Pod fails, it will not restart
1076
 automatically, leading to potential downtime. A steady state related to this issue
1077
 would be to ensure that the Pod is running and available. This can be measured by
 checking the number of running Pods. Since there is only one Pod, the steady state is
1078
 that the Pod should always be in a 'Running' state.
```

# 1-2: Agent	for defining a threshold
System: You are a bel	nful AI assistant for Chaos Engineering
Given k8s man	ifests for a network system, its steady state, and the current value o
the steady st	ate, you will define the threshold for the steady state.
- The thresho	ne following rules: ld must be representative value (e.g., ratio, percentage, ect.), pot f
absolute val	ue.
- The thresho	ld must include reasonable tolerance that makes the threshold being mo
- The current	value of the steady state must satisfy the threshold (including tole:
) as the curr	rent value is the normal state and the threshold represents whether the
- The output	s normal. should be formatted as a JSON instance that conforms to the JSON sche
below.	
As an example description\"	, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \" : \"a list of strings\". \"type\": \"arrav\". \"items\": {\"type\": \
<pre>string\"}},</pre>	<pre>\"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is</pre>
well-formatte	d instance of the schema. The object {\"properties\": {\"foo\": [\"ba
\"baz\"]}} is	not well-formatted.
Here is the o	utput schema:
()"propertice	\"• {\"thought\"• {\"tit]e\"• \"Thought\" \"decorintion\"• \"Write y
thought proce	ss to determine the threshold of the steady state.\", \"type\": \"str
$\"}, \"$ thresh	old\": {\"title\": \"Threshold\", \"description\": \"the threshold of
<pre>steady state, \"}}. \"requi</pre>	<pre>which should be satisfied in the current state.\", \"type\": \"strin red\": [\"thought\". \"threshold\"]}</pre>
# Here is the	overview of my system:
{user_input2}	
# You will de	termine a reasonable threshold for the following staady state of my sy
:	comme a reasonable enconora for the forlowing sectary state of my s
{steady_state	_name}: {steady_state_thought}
{inspection_s	ummary}
# Please foll	ow the instructions below regarding Chaos Engineering:
{ce_instructi	ons}
Now, please d	efine a reasonable threshold for the steady state according to the abo
information.	
Example text	embedded to inspection_summary
# The Python	code of k8s client libraries to inspect the current state of the stead
## Script:	reduce are the forrowing.
```python	
import os\nim	port time es import client, config
def check_pod	_status(namespace, pod_name, duration):
# Load Ku	bernetes configuration based on the environment
if os.get	env('KUBERNETES_SERVICE_HOST'): g.load incluster config()
else:	,
confi	g.load_kube_config()
vl = clie	nt.CoreVlApi()
running_c	$\operatorname{punt} = 0$
<i>.</i> .	
IOT _ in trv:	range(duration):
1.	od = v1.read_namespaced_pod(name=pod_name, namespace=namespace)
p	f pod status phase (Dupping).
p i	<pre>rupping count += 1</pre>
p i	<pre>running_count += 1 print(f\"Pod status: {pod.status.phase}\")</pre>
p i excep	<pre>running_count += 1 print(f\"Pod status: {pod.status.phase}\") t client.exceptions.ApiException as e:</pre>
p i excep	<pre>running_count += 1 print(f\"Pod status: {pod.status.phase}\") t client.exceptions.ApiException as e: rint(f\"Exception when calling CoreVIApi->read_namespaced_pod: {e}\")</pre>

```
1134
                 print(f\"Pod was running {running_count} out of {duration} seconds.\")
1135
1136
             if __name__ == '__main__
                                       1:
                 import argparse
1137
                 parser = argparse.ArgumentParser(description='Check the running state of a Pod.')
1138
                 parser.add_argument('--duration', type=int, default=5, help='Duration to check the
                 Pod status in seconds.')
1139
                 args = parser.parse_args()
1140
                 check_pod_status(namespace='default', pod_name='example-pod', duration=args.
                 duration)
1141
             • • •
1142
             ## Result (current state):
1143
             Pod status: Running
1144
             Pod status: Running
             Pod status: Running
1145
             Pod status: Running
1146
             Pod status: Running
             Pod was running 5 out of 5 seconds.
1147
1148
1149
             #1-3-a: Agent for writing VaC script (K8s Python API)
1150
1151
1152
             Svstem:
             You are a helpful AI assistant for writing unit tests in Python.
1153
             Given the steady state, python script to inspect it, and its threshold, please write a
1154
             Python unit test (including for-loop for certain duration) to verify if the steady
             state satisfies the threshold by adding assertion.
1155
             Always keep the following rules:
1156
             - Include as many comments as possible in your code so that humans can easily
1157
             understand what you did later.
             - Use the Kubernetes Python API.
- Add argparse '--duration' (type=int) so that users can specify the loop duration as
1158
             the previous python script.
1159
             - NEVER use "unittest" module to use argparse.
1160
             - Create a unit test by inheriting from the 'K8sAPIBase' class below (available via ``` from unittest_base import K8sAPIBase```):
1161
              '''python
1162
             import os
             from kubernetes import client, config
1163
1164
             class K8sAPIBase:
                              (self):
                 def ___init__
1165
                      # Load Kubernetes configuration based on the environment
1166
                      if os.getenv('KUBERNETES_SERVICE_HOST'):
                          config.load_incluster_config()
1167
                      else:
1168
                          config.load_kube_config()
1169
                      # Create a Kubernetes API client
1170
                      self.v1 = client.CoreV1Api()
             • • •
1171
             - Add an entry point at the bottom to allow the test to be run from the command line,
1172
             as follows:
1173
             if _____name___ == '____main___':
             ___..ame__
main()
1174
1175
             - The output should be formatted as a JSON instance that conforms to the JSON schema
1176
             below.
1177
             As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
1178
             description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \"
string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a
1179
             well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\",
1180
             \"baz\"]}} is not well-formatted.
1181
             Here is the output schema:
1182
             {\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Describe how
1183
             you add the threshold assertion to the inspection Python script.\", \"type\": \"string \"}, \"code\": {\"title\": \"Code\", \"description\": \"Python unit test code.
1184
             Implement a for loop that checks the status every second for the duration, and
1185
             implement assertion for the summary at the end.\\n- Please add a Add a entry point at
1186
             the bottom to allow the test to be run from the command line.\\n- Please add argparse
              --duration' (type=int) so that users can specify the loop duration. Write only the
1187
```

```
1188
            content of the code, and for dictionary values, enclose them within a pair of single
1189
            double quotes (\\\").\", \"type\": \"string\"}}, \"required\": [\"thought\", \"code\"]}
1190
1191
            Human:
1192
            The steady state:
            {steady_state_name}: {steady_state_thought}
1193
1194
            The steady state was inspected with the following python code of k8s client libraries:
            {script (inspection_summary without results)}
1195
1196
            The threshold of the steady state: {steady_state_threshold}; {
            steady_state_threshold_description}
1197
1198
            Given the above steady state, command, and threshold, please write a Python unit test
            to check if the steady state satisfies the threshold.
1199
            The threshold in the unit test must exactly match the threshold defined above.
1200
            Implement it to support variable durations. Use a representative value (e.g.,
            percentage, ratio, etc.) for the threshold. NEVER use any fixed absolute values for the
1201
             threshold.
1202
1203
             ----- In the verification loop, the prompts below will be stacked as history -----
1204
1205
            AI:
1206
            {output}
1207
            User:
1208
            Your current unittest cause errors when coducted.
            The error message is as follows:
1209
            {error_message]
1210
            Please analyze the reason why the errors occur, then fix the errors.
1211
            Always keep the following rules:
1212
            - Ensure that the implementation supports variable durations again.
            - NEVER repeat the same fixes that have been made in the past.
1213
            - Fix only the parts related to the errors without changing the original content.
1214
            - the same format instructions as in the System role
1215
1216
1217
            # 1-3-b: Agent for writing VaC script (K6 Javascript)
1218
1219
            System:
1220
            You are a helpful AI assistant for writing unit tests in k6.
            Given a steady state, k6 javascript to inspect it, and its threshold, please write a k6
1221
             unit test to verify if the steady state satisfies the threshold by adding threshold
1222
            options.
            Always keep the following rules:
1223
            - Include as many comments as possible in your code so that humans can easily
1224
            understand what you did later.
            - Add "thresholds" in "options" section to the given k6 javascript.
1225
            - {format_instructions}
1226
            Human:
1227
            The steady state:
1228
            {steady_state_name}: {steady_state_thought}
1229
            The steady state can be inspected with the following k6 javascript:
1230
            {script (inspection_summary without results) }
1231
            The threshold of the steady state: {steady_state_threshold}; {
1232
            steady_state_threshold_description}
1233
            Given the above steady state, k6 javascript, and threshold, please write a k6 unit test
1234
             to check if the steady state satisfies the threshold by adding threshold options.
            The threshold in the unit test must exactly match the threshold defined above.
1235
1236
             ----- In the verification loop, the prompts below will be stacked as history -----
1237
1238
            AI:
1239
            {output}
1240
            User:
1241
```

1242	
1010	Your current unittest cause errors when adducted
1243	The error message is as follows:
1244	{error_message}
1245	Please analyze the reason why the errors occur, then fix the errors.
1246	Always keep the following rules:
1247	 Ensure that the implementation supports variable durations again. NEVER repeat the same fixes that have been made in the past.
1248	- Fix only the parts related to the errors without changing the original content.
1249	- the same format instructions as in the System role
1250	
1251	
1252	Example text embedded to steady_state_threshold
1253	
1254	The Pod should be in the 'Running' state at least 90\% of the time during the
1255	observation period.
1256	
1257	
1258	Example text embedded to <pre>state_threshold_description</pre>
1250	
1200	The steady state we are considering is the 'example-pod-running-state', which requires
1200	the Pod to be in a 'Running' state. The current state shows that the Pod was running 5
1201	out of 5 seconds, which is 100\% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small
1262	percentage of time where the Pod might not be in the 'Running' state due to transient
1263	issues. A reasonable threshold could be that the Pod should be in the 'Running' state
1264	while still ensuring the Pod is mostly available.
1265	
1266	
1267	# 1-4: Agent for drafting failure injection
1268	
1269	Rush and
1270	System: You are a helpful AI assistant for Chaos Engineering.
1271	
1 4 1	Given k8s manifests for a system, the steady states of the system, and user's
1272	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions.
1272 1273	Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc.
1272 1273 1274	Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: - First assume a real-world event that may be most impactful in the the system, such
1272 1273 1274 1275	 Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc.
1272 1273 1274 1275 1276	 Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of
1272 1273 1274 1275 1276 1277	 Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the
1272 1273 1274 1275 1276 1277 1278	 Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady
1272 1273 1274 1275 1276 1277 1278	 Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of <i>l</i>
1272 1273 1274 1275 1276 1277 1278 1279	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: - First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. - Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. - Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. - The injected faults should be selected from the following fault types of { ce_tool_name}:
1272 1273 1274 1275 1276 1277 1278 1279 1280	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unpaulability.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailablility, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name): PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailablility, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss,
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name): PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. DNSChaos: simulates DNS failures, such as the parsing failure of DNS domain name
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. DNSChaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name): PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned. HTTPChaos: simulates HTTP communication failures, such as HTTP communication
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286	Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailablility, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. DNSChaos: simulates HTP communication failures, such as HTTP communication latency. StressChaos: simulates CPU race or memory race.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287	Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. DNSChaos: simulates HTP communication failures, such as HTP communication latency. StressChaos: simulates CPU race or memory race. IOChaos: simulates the I/O failure of an application file, such as I/O delays, read red write feithweat
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288	 Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. DNSChaos: simulates MTTP communication failures, such as HTTP communication latency. StressChaos: simulates CPU race or memory race. IOChaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. TimeChaos: simulates the time jump exception.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289	 Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. DNSChaos: simulates CPU race or memory race. IOChaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. The output should be formatted as a JSON instance that conforms to the JSON schema
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290	 Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. Networkchaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned. HTTPChaos: simulates HTTP communication failures, such as HTTP communication latency. StressChaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. TimeChaos: simulates the time jump exception.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1287 1288 1289 1290 1291	 Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. DNSChaos: simulates HTTP communication failures, such as HTTP communication latency. StressChaos: simulates CPU race or memory race. IOChaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. TimeChaos: simulates the time jump exception. The output should be formatted as a JSON instance that conforms to the JSON schema below.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1288 1289 1290 1291 1292	 Given K8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. The injected faults should be selected from the following fault types of { ce_tool_name}: PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. NetworkChaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned. HTTPChaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. TimeChaos: simulates the time jump exception. The output should be formatted as a JSON instance that conforms to the JSON schema below.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1287 1288 1289 1290 1291 1292	Given K85 manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: - First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. - Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. - Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. - The injected faults should be selected from the following fault types of { ce_tool_name}: - PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-Kill', 'container-kill'. - NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. - DNSChaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned. - HTTPChaos: simulates CPU race or memory race. - IOCchaos: simulates CPU race or memory race. - IOCchaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. - TimeChaos: simulates the time jump exception. - The output should be formatted as a JSON instance that conforms to the JSON schema below. As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \" description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \" array\", \"items\": {\"type\": \" arr
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1286 1287 1288 1289 1290 1291 1292 1293 1294	Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: - First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. - Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. - Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. - The injected faults should be selected from the following fault types of { ce_tool_name}: - PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. - Networkchaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. - DNSChaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned. - HTTPChaos: simulates HTP communication failures, such as HTP communication latency. - Stresschaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. - TimeChaos: simulates the time jump exception. - The output should be formatted as a JSON instance that conforms to the JSON schema below. As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \" description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \" string\"}} is a well-formatted.
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294	Given k8s manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: - First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. - Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. - Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. - The injected faults should be selected from the following fault types of { ce_tool_name}: - PodChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. - NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. - DNSChaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned. - HTTPChaos: simulates HTP communication failures, such as HTP communication latency. - StressChaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. - TimeChaos: simulates the time jump exception. - The output should be formatted as a JSON instance that conforms to the JSON schema below. As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \" description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"too\"; {\"bar\", \"baz\"]} is a well-formatted. - HTre output schema:
1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295	Given KMS manifests for a system, the steady states of the system, and user's instructions for Chaos Engineering, you will define the most impactful fault injections to reveal potential weaknesses of the system, such as insufficient recovery functions, resource allocation, redundancy, etc. Always keep the following rules: - First, assume a real-world event that may be most impactful in the the system, such as promotion campaign, cyber attacks, disasters, etc. - Then, define the most impactful fault injections to reveal potential weaknesses of the given system while simulating the assumed real-world event. - Prioritize fault injections that target the system's weak resources related to the steady states to verify whether those resources can handle the faults and the steady states can be maintained. - The injected faults should be selected from the following fault types of { ce_tool_name}: - Prointize faults should be selected from the following fault types of { ce_tool_name}: - NetworkChaos: simulates Pod failures, such as Pod node restart, Pod's persistent unavailability, and certain container failures in a specific Pod. The supported subtypes include 'pod-failure', 'pod-kill', 'container-kill'. - NetworkChaos: simulates network failures, such as network latency, packet loss, packet disorder, and network partitions. - DNSChaos: simulates DNS failures, such as the parsing failure of DNS domain name and the wrong IP address returned. - HTPChaos: simulates the I/O failure of an application file, such as I/O delays, read and write failures. - TimeChaos: simulates the time jump exception. - The output should be formatted as a JSON instance that conforms to the JSON schema below. As an example, for the schema {\"properties\": {\"fon\": {\"title\": \"Foo\", \" description\": \" a list of strings\", \"type\": \"array\", \"items\": {\"too\": {\"baz\"]} is a well-formatted instance of the schema. The object {\"properties\": {\"foo\": {\"baz\"]} is a well-formatted. Here is the output schema:

	{\"properties\": {\"event\": {\"title\": \"Event\", \"description\": \"Consider a real-	
	world fault event that may be most impactful of the system, such as promotion campaign,	
	Thought\", \"description\": \"Write down your thought process to define a sequence of	
	fault injections that exploit the system's weaknesses of while simulating the fault	
	event: 1) how the system's weaknesses affect the steady state; 2) how each fault injection evolut the system's weaknesses: 3) how the sequence simulates the phenamena	
	in the fault event (consider carefully the sequence order). Prioritize fault injections	
	that directly attack the weaknessses of the system, such as insufficient recovery	
	<pre>iunctions, resource allocation, redundancy, etc.\", \"type\": \"string\"}, \"faults\": {\"title\": \"Faults\", \"description\": \"Define a sequence of fault injections that</pre>	
	exploit the system's vulnerabilities to the fullest according to the above thoughts. In	
	the inner list, a set of simultaneously injected faults are listed, while in the outer	
	fault_c]] indicates that fault_a is injected, then fault_b and fault_c are injected	
	<pre>simultaneously.\", \"type\": \"array\", \"items\": {\" smaflue, \"#/definitiong/Eault\", \"type\": \"array\", \"items\": {\"</pre>	
	<pre>\": \"#/definitions/Fault\"; {\"required\": [\"event\", \"Lought\", \"laults \"], \"definitions\": {\"Fault\": {\"title\": \"Fault\", \"type\": \"object\", \"</pre>	
	<pre>properties\": {\"name\": {\"title\": \"Name\", \"description\": \"Select a fault type</pre>	
	<pre>trom [\\\"PodChaos\\\", \\\"NetworkChaos\\\", \\\"DNSChaos\\\", \\\"HTTPChaos\\\", \\\" StressChaos\\\", \\\"IOChaos\\\", \\\"TimeChaos\\\"]\", \"enum\", [\"PodChaos\\", \\"</pre>	
	NetworkChaos\", \"DNSChaos\", \"HTTPChaos\", \"StressChaos\", \"IOChaos\", \"TimeChaos	
	<pre>\"], \"type\": \"string\"}, \"name_id\": {\"title\": \"Name Id\", \"description\": \"An identifier to provent name conflicts when the same "suit arrest."</pre>	
	starting from 0 in sequential order to prevent name conflicts.\", \"type\": \"integer	
	<pre>\"}, \"scope\": {\"title\": \"Scope\", \"description\": \"Specify only the fault</pre>	
	<pre>injection scope (i.e., the target resource where the fault is injected) in advance here \". \"type\". \"object\". \"additionalProperties\". {\"type\". \"string\"}}</pre>	
	required\": [\"name\", \"name_id\", \"scope\"]}}	
	Human :	
	Here is the overview of my system:	
	<pre>{user_input2}</pre>	
	Steady states of the network system defined by the manifests are the following:	
	{steady_states}	
	Please follow the instructions below regarding Chaos Engineering as necessary:	
	{ce_instructions}	
	Now, please define fault injections to reveal the system's vulnerabilities.	
	Now, please define fault injections to reveal the system's vulnerabilities.	
	Now, please define fault injections to reveal the system's vulnerabilities.	
	Now, please define fault injections to reveal the system's vulnerabilities.	
-	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined.	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined.	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. 1st steady states: - Name: example-pod-running-state	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. 1st steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state.	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state. - Threshold for the steady state: The Pod should be in the 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. 1st steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state. - Threshold for the steady state: The Pod should be in the 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. 1st steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in the 'Running' state. - Threshold for the steady state: The Pod should be in the 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable are con-	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: Name: example-pod-running-state Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: Name: example-pod-running-state Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state. Threshold for the steady state: The Pod should be in the 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be in the 'Running' state due to transient issues. A reasonable threshold could be that 	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod/yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state Threshold for the steady state: The Pod should be in the 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be in the 'Running' state due to transient issues. A reasonable threshold could be that the Pod should be in the 'Running' tate at least 90% of the time during the observation period. This allows for some tolerance while still ensuring the Pod is	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod/yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state Threshold for the steady state: The Pod should be in the 'Running' state. At least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod maint not be in the 'Running' state due to transient issues. A reasonable threshold could be that the Pod should be in the 'Running' the pod is mostly available.	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state. - Threshold for the steady state: The Pod should be in the 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be in the 'Running' state due to transient issues. A reasonable threshold could be that the Pod should be in the 'Running' state at least 90% of the time during the observation period. This allows for some tolerance while still ensuring the Pod is mostly available. - Whether the steady state meets the threshold is determined by the following Python servict with X8e ABL:	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be in the 'Running' state due to transient issues. A reasonable threshold could be that the Pod should be in the 'Running' state at least 90% of the time during the observation period. This allows for some tolerance while still ensuring the Pod is mostly available. - Whether the steady state meets the threshold is determined by the following Python script with K8s API: 	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state at least 90% of the time during the observation period.; The steady state ware considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be in the 'Running' state due to transient issues. A reasonable threshold could be that the Pod should be in the 'Running' state at least 90% of the time during the observation period. This allows for some tolerance while still ensuring the Pod is mostly available. - Whether the steady state meets the threshold is determined by the following Python script with K8s API: '''' import os	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod.yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state at least 90% of the time during the observation period.; The steady state we are considering is the 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be in the 'Running' state due to transient issues. A reasonable threshold could be that the Pod should be in the 'Running' state a least 90% of the time during the observation period. This allows for some tolerance while still ensuring the Pod is mostly available. - Whether the steady state meets the threshold is determined by the following Python script with K&S API: '''' import os import time import time import time	
	Now, please define fault injections to reveal the system's vulnerabilities. Example text embedded to steady_states Steady states of the network system defined by the manifests are the following: 2 steady states are defined. Ist steady states: - Name: example-pod-running-state - Description: The first issue to address is the Pod's restart policy set to 'Never' in the 'nginx/pod/yaml' manifest. This is a critical issue because if the Pod fails, it will not restart automatically, leading to potential downtime. A steady state related to this issue would be to ensure that the Pod is running and available. This can be measured by checking the number of running Pods. Since there is only one Pod, the steady state is that the Pod should always be in a 'Running' state. - Threshold for the steady state: The Pod should be in the 'Running' state. The 'example-pod-running-state', which requires the Pod to be in a 'Running' state. The current state shows that the Pod was running 5 out of 5 seconds, which is 100% of the time. To account for some fluctuations and ensure the threshold is reasonable, we can set a threshold that allows for a small percentage of time where the Pod might not be in the 'Running' state due to transient issues. A reasonable threshold could be that the Pod should be in the 'Running' state at least 90% of the time during the Pod is mostly available. - Whether the steady state meets the threshold is determined by the following Python script with K88 API: '''' import os import time import time import science import client, config	

1350	
1351	class TestPodRunningState(K8sAPIBase):
1352	
1052	if name == ' main '·
1050	parser = argparse.ArgumentParser(description='Test the running state of a Pod.')
1000	parser.add_argument('duration', type=int, default=5, help='Duration to check the
1300	args = parser.parse_args()
1356	<pre># Create an instance of the test class and run the test test = TestBedBunningState(namenpage/default/_ ned name/example_ped/_ duration=</pre>
1357	args.duration)
1358	<pre>test.test_pod_running_state()</pre>
1359	
1360	2nd steady states:
1361	 Name: example-service-http-response-state Description: The next issue to address is the lack of redundancy due to the single
1362	Pod deployment in the 'nginx/pod.yaml' manifest. This is a significant issue because if
1363	the Pod fails, there is no automatic recovery or redundancy, which can lead to service
1364	Service is able to route traffic to the Pod. This can be measured by checking the
1365	Service's ability to respond to HTTP requests successfully. Since the Service is
1366	respond with a successful HTTP status code (e.g., 200 OK) for a certain percentage of
1367	requests.
1368	- Threshold for the steady state: 95% of HTTP requests should return a 200 OK status.; The steady state for the system is defined as the Service's ability to respond with a
1369	successful HTTP status code (200 OK) for a certain percentage of requests. The current
1370	state shows that 100% of the requests received a 200 OK status, which indicates a
1371	the threshold is reasonable, we should allow for some tolerance. A common practice is
1372	to set a threshold slightly below the current perfect state to accommodate minor, non-
1373	threshold at 95% ensures that the system is considered healthy as long as it maintains
1374	a high level of successful responses, while still allowing for some minor issues.
1275	- Whether the steady state meets the threshold is determined by the following K6 Javascript:
1070	1) 1) 1) 1
1077	<pre>import http from 'k6/http';\nimport { check } from 'k6';</pre>
1070	export const options = {
1378	vus: 5,
1379	thresholds: {
1380	<pre>'http_req_failed': ['rate<0.05'],</pre>
1381	}; };
1382	
1383	<pre>export default function () { const res = http.get('http:\/\/example-service.default.svc.cluster.local:80');</pre>
1384	check (res, {
1385	'is status 200': (r) => r.status === 200,
1386	}
1387	
1388	
1389	
1390	# 1-5: Agent for determining detailed failure parameters
1391	
1392	System:
1393	You are a helpful AI assistant for Chaos Engineering.
1394	Given k8s manifests that define a network system, its steady states, and a fault type
1395	fault.
1306	Always keep the following rules:
1207	- ray attention to namespace specification. If the namespace is specified in the manifest, it is deployed with the namespace. If not, it is deployed with the 'default'
1000	namespace.
1398	- The parameters follow the format of Chaos Mesh.
1399	Human:
1400	Here is the overview of my system:
1401	{user_input2}
1402	Steady states of my system:
1403	{steady_states}

1404 1405 A fault scenario that may occur in my system and may affect the steady states: 1406 {fault scenario} 1407 Please follow the instructions below regarding Chaos Engineering as necessary: 1408 {ce\_instructions} 1409 Now, please detail the parameters of the fault "{refined\_fault\_type}". 1410 {detailed\_param\_instructions} 1411 1412 ----- In the verification loop, the prompts below will be stacked as history -----1413 1414 AI: {output} 1415 1416 Human: Your current fault parameters cause errors when conducted. 1417 The error message is as follows: 1418 {error message} 1419 Please analyze the reason why the errors occur, then fix the errors. 1420 Always keep the following rules: - NEVER repeat the same fixes that have been made in the past. 1421 - Fix only the parts related to the errors without changing the original intent. 1422 1423 1494 Example text embedded to fault\_scenario 1425 1426 An assumed fault scenario is as follows: 1427 - Event: Cyber Attack Simulation 1428 - Used Chaos Engineering tool: Chaos Mesh - Faults to simulate the event: [[{'name': 'PodChaos', 'name\_id': 0, 'scope': {'pod': ' 1429 example-pod'}}], [{'name': 'NetworkChaos', 'name\_id': 0, 'scope': {'service': 'example-1430 service' }}]] - Description: Given the system's weaknesses, a cyber attack targeting the web server 1431 could be highly impactful. The Pod's restart policy set to 'Never' and the single Pod 1432 deployment without redundancy are critical vulnerabilities. If the Pod fails, it will not restart, leading to downtime, and the lack of redundancy means there is no backup 1433 to handle traffic. To simulate a cyber attack, we can inject faults that exploit these 1434 weaknesses. First, we will use PodChaos to simulate a Pod failure, which will test the system's ability to maintain the 'example-pod-running-state'. Since the Pod will not 1435 restart automatically, this will directly impact the steady state. Next, we will use NetworkChaos to simulate network latency, which will test the system's ability to 1436 maintain the 'example-service-http-response-state'. This sequence simulates a cyber 1437 attack where the Pod is targeted first, followed by network disruptions, revealing the 1438 system's vulnerabilities in handling such events. 1439 1440 Example text embedded to **refined\_fault\_type** 1441 1442 1443 NetworkChaos({'service': 'example-service'}) 1444 1445 detailed\_param\_instructions for PodChaos (template embedded dynamically) 1446 1447 1448 The output should be formatted as a JSON instance that conforms to the JSON schema below. 1449 1450 As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]}
the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The 1451 1452 object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted. 1453 Here is the output schema: 1454 {"properties": {"action": {"title": "Action", "description": "Specifies the fault type 1455 from 'pod-failure', 'pod-kill', or 'container-kill'. Note that you may select 'pod-1456 failure' only when the target Pod's container has livenessProbe and readinessProbe defined.", "example": "pod-kill", "enum": ["pod-failure", "pod-kill", "container-kill 1457

1458	
1450	"I "time", "string") "mode", ("title", "Mode" "description", "Specifies the mode of
1459	the experiment. The mode options include 'one' (selecting a random Pod), 'all' (
1460	selecting all eligible Pods), 'fixed' (selecting a specified number of eligible Pods),
1461	'fixed-percent' (selecting a specified percentage of Pods from the eligible Pods), and
1462	'random-max-percent' (selecting the maximum percentage of Pods from the eligible Pods) ". "example": "one". "enum": ["one". "all". "fixed-percent". "random-max-
1463	percent"], "type": "string"}, "value": {"title": "Value", "description": "Provides
1/6/	parameters for the mode configuration, depending on mode. For example, when mode is set
1465	to fixed-percent, value specifies the percentage of Pods.", "example": "1", "type": " string", "selector", ("title", "Selector", "description", "Specifies the target Pod ".
1465	"example": null, "allof": [{"\$ref": "#/definitions/Selectors"}]}, "containerNames": {"
1/67	this configuration is mandatory to specify the target configure faction in ecting
1400	<pre>faults.", "example": ["prometheus"], "type": "array", "items": {"type": "string"}}}, "</pre>
1468	required": ["action", "mode", "selector"], "definitions": {"SetBasedRequirements": {"
1469	<pre>title": "SetBasedRequirements", "type": "object", "properties": {"key": {"title": "Key ", "description": "Label key", "type": "string"}, "operator": {"title": "Operator", "</pre>
1470	description": "Select an operator.", "enum": ["In", "NotIn", "Exists", "DoesNotExist"],
1471	"type": "string"}, "values": {"title": "Values", "description": "Label values. The
1472	values set must be non-empty in the case of In and NotIn.", "type": "array", "items":
1473	": "Selectors", "type": "object", "properties": {"namespaces": {"title": "Namespaces",
1/7/	"description": "Specifies the namespace of the experiment's target Pod. If this
1475	selector is None, Chaos Mesh will set it to the namespace of the current Chaos experiment "type", "array" "items", ("type", "string"), "labelSelectors", ("title
1475	": "Labelselectors", "description": "Specifies the label, hey/value pairs that the
1476	experiment's target Pod must have. If multiple labels are specified, the experiment
1477	target must have all the labels specified by this selector.", "type": "object", "
1478	Expressionselectors", "description": "Specifies a set of expressions that define the
1479	label's rules to specifiy the experiment's target Pod.", "example": [{"key": "tier", "
1480	operator": "In", "values": ["cache"]}, {"key": "environment", "operator": "NotIn", " values": ["dey"]]] "type": "arryu" "items": ["creft". "#'definitions/
1481	SetBasedRequirements"}}, annotationSelectors": {"title": "Annotationselectors", "
1/182	description": "Specifies the annotation-key/value pairs that the experiment's target
1402	Pod must have. If multiple annotations are specified, the experiment target must have all annotations specified by this selector " "type". "object" "additionalProperties".
1483	{"type": "string"}}, "fieldSelectors": {"title": "Fieldselectors", "description": "
1484	Specifies the field-key/value pairs of the experiment's target Pod. If multiple fields
1485	are specified, the experiment target must have all fields set by this selector.", "
1486	", "additionalProperties": {"type". "string"}, metadata.namespace. uaradit, type. Object
1487	Podphaseselectors", "description": "Specifies the phase of the experiment's target Pod.
1488	If this selector is None, the target Pod's phase is not limited.", "type": "array", "
1489	string"}}, "nodeSelectors"; {"title": "Nodeselectors", "description": "Specifies the
1/100	node-label-key/value pairs to which the experiment's target Pod belongs.", "type": "
1404	object", "additionalProperties": {"type": "string"}}, "nodes": {"title": "Nodes", " description": "Specifies the node to which the experiment's target Pod belongs. The
1491	target Pod can only belong to one node in the configured node list. If multiple node
1492	labels are specified, the node to which the experiment's target Pod belongs must have
1493	all labels specified by this selector.", "type": "array", "items": {"type": "string"}}, "pods": {"title": "Pods", "description": "Specifies the namespaces and list of the
1494	experiment's target Pods. If you have specified this selector, Chaos Mesh ignores other
1495	<pre>configured selectors.", "example": {"default": ["pod-0", "pod-2"]}, "type": "object",</pre>
1496	<pre>"additionalProperties": {"type": "array", "items": {"type": "string"}}}}}</pre>
1497	
1498	

detailed\_param\_instructions for **NetworkChaos** (template embedded dynamically)

The output should be formatted as a JSON instance that conforms to the JSON schema below.

As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]} the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted.

Here is the output schema:

1499

1500 1501 1502

1503 1504

1505 1506 1507

1508

1509

1510

1511

{"properties": {"action": {"title": "Action", "description": "Indicates the specific fault type. Available types include: netem, delay (network delay), loss (packet loss), duplicate (packet duplicating), corrupt (packet corrupt), partition (network partition) , and bandwidth (network bandwidth limit). After you specify action field, specify

1512	
1513	action-related fields for other necessary field configuration.", "example": "Partition
1514	", "enum": ["netem", "delay", "loss", "duplicate", "corrupt", "partition", "bandwidth "], "type": "string"}, "direction": {"title": "Direction", "description": "Indicates
1515	the direction of target packets. Available vaules include from (the packets from target
1516), to (the packets to target), and both (the packets from or to target). This parameter
1517	example": "both", "enum": ["from", "to", "both"], "type": "string"}, "target": {"title
1518	": "Target", "description": "Used in combination with direction, making Chaos only
1519	empty in netem action.", "allof": [{"\$ref": "#/definitions/Selector"}]}, "mode": {"
1520	title": "Mode", "description": "Specifies the mode of the experiment. The mode options
1521	Include one (selecting a random Pod), all (selecting all eligible Pods), fixed (selecting a specified number of eligible Pods), fixed-percent (selecting a specified
1500	percentage of Pods from the eligible Pods), and random-max-percent (selecting the
1502	<pre>maximum percentage of Pods from the eligible Pods)", "example": "one", "enum": ["one", "all", "fixed", "fixed-percent", "random-max-percent"], "type": "string"}, "yalue": {"</pre>
1523	title": "Value", "description": "Provides parameters for the mode configuration,
1524	depending on mode. For example, when mode is set to fixed-percent, value specifies the
1525	Selector", "description": "Specifies the target Pod.", "allof": [{"\$ref": "#/
1526	<pre>definitions/Selectors"}]}, "externalTargets": {"title": "Externaltargets", "description</pre>
1527	": "Indicates the network targets except for Kubernetes, which can be IPv4 addresses or domains. This parameter only works with direction: to.", "example": ["1.1.1.1", "www.
1528	<pre>google.com"], "type": "array", "items": {"type": "string"}}, "device": {"title": "</pre>
1529	Device", "description": "Specifies the affected network interface", "example": "eth0", "type": "string"} "delay": {"title": "Delay" "description": "When setting action to
1530	delay means simulating network delay fault, you also need to configure this parameters
1531	.", "allOf": [{"\$ref": "#/definitions/Deplay"}]}, "loss": {"title": "Loss", "
1532	also configure this parameters.", "allof": [{"\$ref": "#/definitions/Loss"}]}, "
1533	<pre>duplicated": {"title": "Duplicated", "description": "When setting action to duplicate,</pre>
1534	<pre>meaning simulating package duplication, you can also set this parameters.", "allOf": [{"\$ref": "#/definitions/Duplicate"}]}, "corrupt": {"title": "Corrupt", "description":</pre>
1535	"When setting action to corrupt means simulating package corruption fault, you can also
1536	<pre>configure the following parameters.", "allOf": [{"\$ref": "#/definitions/Corrupt"}]}, " rate": {"title": "Rate". "description": "When setting action to rate means simulating</pre>
1537	bandwidth rate fault, you also need to configure this parameters. This action is
1538	similar to bandwidth/rate below, however, the key distinction is that this action can
1539	over the bandwidth simulation such as limiting the buffer size, select the bandwidth
1540	<pre>action.", "allof": [{"\$ref": "#/definitions/Rate"}]}, "bandwidth": {"title": "Bandwidth " "decomposition": "When patting (pation(to (bandwidth) means simulating bandwidth)</pre>
1541	limit fault, you also need to configure this parameters. This action is mutually
1542	exclusive with any netem action defined above. If you need to inject bandwidth rate
1543	allof": [{"\$ref": "#/definitions/Bandwidth"}]}}, "required": ["action", "mode", "
1544	<pre>selector"], "definitions": {"SetBasedRequirements": {"title": "SetBasedRequirements", "</pre>
1545	type": "object", "properties": {"title": "Key", "description": "Label Key", " type": "string"}, "operator": {"title": "Operator", "description": "Select an operator
1546	.", "enum": ["In", "NotIn", "Exists", "DoesNotExist"], "type": "string"}, "values": {"
15/17	title": "Values", "description": "Label values. The values set must be non-empty in the case of In and NotIn.", "type": "array", "items": {"type": "string"}}}, "required": ["
1547	key", "operator", "values"]}, "Selectors": {"title": "Selectors", "type": "object", "
1540	properties": {"namespaces": {"title": "Namespaces", "description": "Specifies the
1049	it to the namespace of the current Chaos experiment.", "type": "array", "items": {"
1000	type": "string"}}, "labelSelectors": {"title": "Labelselectors", "description": "
1001	multiple labels are specified, the experiment target must have all the labels specified
1552	<pre>by this selector.", "type": "object", "additionalProperties": {"type": "string"}}, " overcosionSelectors": ("title": "Evercosionselectors", "description": "Selectors", "title": "title": "Evercosionselectors", "description": "Selectors", "Selectors, "Selectors", "Selectors, "Selectors</pre>
1553	of expressions that define the label's rules to specify the experiment's target Pod.",
1554	"example": [{"key": "tier", "operator": "In", "values": ["cache"]}, {"key": "
1555	environment", "operator": "NotIn", "values": ["dev"]}], "type": "array", "items": {" \$ref": "#/definitions/SetBasedBeguirements"}}, "annotationSelectors": {"title": "
1556	Annotationselectors", "description": "Specifies the annotation-key/value pairs that the
1557	experiment's target Pod must have. If multiple annotations are specified, the
1558	object", "additionalProperties": {"type": "string"}}, "fieldSelectors": {"title": "
1559	Fieldselectors", "description": "Specifies the field-key/value pairs of the experiment'
1560	fields set by this selector.", "example": {"metadata.name": "my-pod", "metadata.
1561	<pre>namespace": "dafault"}, "type": "object", "additionalProperties": {"type": "string"}},</pre>
1562	"pogrnaseselectors": {"title": "Podphaseselectors", "description": "Specifies the phase of the experiment's target Pod. If this selector is None, the target Pod's phase is
1563	not limited.", "type": "array", "items": {"enum": ["Pending", "Running", "Succeeded", "
1564	<pre>ralled", "Unknown"], "type": "string"}}, "nodeSelectors": {"title": "Nodeselectors", " description": "Specifies the node-label-kev/value pairs to which the experiment's</pre>
1565	

1566	
1567	target Pod belongs.", "type": "object", "additionalProperties": {"type": "string"}}, "
1568	nodes": {"title": "Nodes", "description": "Specifies the node to which the experiment's
1569	list. If multiple node labels are specified, the node to which the experiment's target
1570	Pod belongs must have all labels specified by this selector.", "type": "array", "items
1571	namespaces and list of the experiment's target Pods. If you have specified this
1572	selector, Chaos Mesh ignores other configured selectors.", "example": {"default": ["pod
1573	type": "string"}}}}, "Selector": {"title": "Selector", "type": "object", "properties":
1574	{"mode": {"title": "Mode", "description": "Specifies the mode of the experiment. The
1575	fixed (selecting a specified number of eligible Pods), fixed-percent (selecting a
1576	specified percentage of Pods from the eligible Pods), and random-max-percent (selecting
1577	the maximum percentage of Pods from the eligible Pods)", "example": "one", "enum": [" one", "all", "fixed", "fixed-percent", "random-max-percent"], "type": "string"}, "
1579	selector": {"title": "Selector", "description": "Specifies the target Pod.", "example":
1570	<pre>null, "allof": [{"\$ref": "#/definitions/Selectors"}]}}, "required": ["mode", "selector "]}, "Reorder": {"title": "Reorder", "type": "object", "properties": {"reorder": {"</pre>
1579	title": "Reorder", "description": "Indicates the probability to reorder", "default":
1000	"0", "example": "0.5", "type": "string"}, "correlation": {"title": "Correlation", "
1500	the previous time's length of delay time. Range of value: [0, 100]", "default": "0", "
1582	example": "50", "type": "string"}, "gap": {"title": "Gap", "description": "Indicates
1583	<pre>integer"}}}, "Deplay": {"title": "Deplay", "type": "object", "properties": {"latency":</pre>
1584	{"title": "Latency", "description": "Indicates the network latency", "example": "2ms",
1585	the correlation between the current latency and the previous one. Range of value: [0,
1586	100]. Specify only the number. NEVER include any units.", "example": "50", "type": "
1587	<pre>string"}, "jitter": {"title": "jitter", "description": "indicates the range of the network latency", "example": "lms", "type": "string"}, "reorder": {"title": "Reorder",</pre>
1588	"description": "Indicates the status of network packet reordering", "allof": [{"\$ref":
1589	<pre>"#/definitions/Reorder"}]}}}, "Loss": {"title": "Loss", "type": "object", "properties": {"loss": {"title": "Loss", "description": "Indicates the probability of packet loss.</pre>
1590	Range of value: [0, 100]. Specify only the number. NEVER include any units.", "default
1591	": "U", "example": "50", "type": "string"}, "correlation": {"title": "Correlation", " description": "Indicates the correlation between the probability of current packet loss
1592	and the previous time's packet loss. Range of value: [0, 100]. Specify only the number
1593	. NEVER include any units.", "default": "0", "example": "50", "type": "string"}}}, " Duplicate": {"title": "Duplicate", "type": "object", "properties": {"duplicate": {"
1594	title": "Duplicate", "description": "Indicates the probability of packet duplicating.
1595	Range of value: [0, 100]. Specify only the number. NEVER include any units.", "default ": "0", "example": "50", "type": "string"}, "correlation": {"title": "Correlation", "
1596	description": "Indicates the correlation between the probability of current packet
1597	duplicating and the previous time's packet duplicating. Range of value: [0, 100]. Specify only the number NEVER include any units ". "default": "0". "example": "50". "
1598	type": "string"}}}, "Corrupt": {"title": "Corrupt", "type": "object", "properties": {"
1599	corrupt": {"title": "Corrupt", "description": "Indicates the probability of packet corruption. Range of value: [0, 100]. Specify only the number. NEVER include any units
1600	.", "default": "0", "example": "50", "type": "string"}, "correlation": {"title": "
1601	Correlation", "description": "Indicates the correlation between the probability of current packet corruption and the previous time's packet corruption. Range of value:
1602	[0, 100]. Specify only the number. NEVER include any units.", "default": "0", "example
1603	": "50", "type": "string"}}}, "Rate": {"title": "Rate", "type": "object", "properties": {"rate": {"title": "Rate", "description": "Indicates the rate of bandwidth limit
1604	Allows bit, kbit, mbit, gbit, tbit, bps, kbps, mbps, gbps, tbps unit. bps means bytes
1605	per second", "example": "lmbps", "type": "string"}}, "Bandwidth": {"title": "Bandwidth
1606	the rate of bandwidth limit. Allows bit, kbit, mbit, gbit, tbit, bps, kbps, mbps, gbps
1607	, tbps unit. bps means bytes per second", "example": "1mbps", "type": "string"}, "limit
1608	<pre>: { title : "Limit", "description": "indicates the number of bytes waiting in queue", "example": 1, "type": "integer"}, "buffer": {"title": "Buffer", "description": "</pre>
1609	Indicates the maximum number of bytes that can be sent instantaneously", "example": 1,
1610	"type": "integer"}, "peakrate": {"title": "Peakrate", "description": "Indicates the maximum consumption of bucket (usually not set)", "example": 1, "type": "integer"}. "
1611	minburst": {"title": "Minburst", "description": "Indicates the size of peakrate bucket
1612	<pre>(usually not set)", "example": 1, "type": "integer"}}}</pre>
1613	
1614	

detailed\_param\_instructions for DNSChaos (template embedded dynamically)

The output should be formatted as a JSON instance that conforms to the JSON schema below.

1620 As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a 1621 list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]}
the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The 1622 object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted. 1623 1624 Here is the output schema: 1625 {"properties": {"action": {"title": "Action", "description": "Defines the behavior of 1626 DNS fault from 'random' or 'error'. When the value is random, DNS service returns a random IP address; when the value is error, DNS service returns an error.", "example": 1627 "random", "enum": ["random", "error"], "type": "string"}, "mode": {"title": "Mode", " 1628 description": "Specifies the mode of the experiment. The mode options include 'one' (selecting a random Pod), 'all' (selecting all eligible Pods), 'fixed' (selecting a 1629 specified number of eligible Pods), 'fixed-percent' (selecting a specified percentage 1630 of Pods from the eligible Pods), and 'random-max-percent' (selecting the maximum percentage of Pods from the eligible Pods)", "example": "one", "enum": ["one", "all", 1631 "random-max-percent"], "type": "string"}, "value": {"title": " fixed", "fixed-percent", 1632 Value", "description": "Provides parameters for the mode configuration, depending on mode. For example, when mode is set to fixed-percent, value specifies the percentage of Pods.", "example": "1", "type": "string"}, "patterns": {"title": "Patterns", " 1633 1634 description": "Selects a domain template that matches faults. The fault is applyed to these domains. Placeholder ? and wildcard \* are supported, but the wildcard in patterns 1635 configuration must be at the end of string. For example, chaos-mes\*.org, is an invalid 1636 configuration. When patterns is not configured, faults are injected for all domains.", "items": {"type "example": "google.com, chaos-mesh.org, github.com", "type": "array", 1637 ": "string"}}, "selector": {"title": "Selector", "description": "Specifies the target 1638 Pod.", "example": null, "allOf": [{"\$ref": "#/definitions/Selectors"}]}}, "required": ["selector"], "definitions": {"SetBasedRequirements": {"title": "SetBasedRequirements", 1639 "type": "object", "properties": {"Key": {"title": "Key", "description": "Label key", type": "string"}, "operator": {"title": "Operator", "description": "Select an operator 1640 type": "string"}, "operator": {"title": "Operator", "description": "Select an operator .", "enum": ["In", "NotIn", "Exists", "DoesNotExist"], "type": "string"}, "values": {" title": "Values", "description": "Label values. The values set must be non-empty in the case of In and NotIn.", "type": "array", "items": {"type": "string"}}, "required": [" key", "operator", "values"]}, "Selectors": {"title": "Selectors", "type": "object", " properties": {"namespaces": {"title": "Namespaces", "description": "Specifies the namespace of the experiment's target Pod. If this selector is None, Chaos Mesh will set it the promerse of the experiment' ("array", "times", "type", "typ 1641 1642 1643 1644 it to the namespace of the current Chaos experiment.", "type": "array", "items": {" 1645 type": "string"}}, "labelSelectors": {"title": "Labelselectors", "description": ' 1646 Specifies the label-key/value pairs that the experiment's target Pod must have. If multiple labels are specified, the experiment target must have all the labels specified by this selector.", "type": "object", "additionalProperties": {"type": "string"}}, " expressionSelectors": {"title": "Expressionselectors", "description": "Specifies a set 1647 1648 of expressions that define the label's rules to specifyy the experiment's target Pod.", "example": [{"key": "tier", "operator": "In", "values": ["cache"]}, {"key": " environment", "operator": "NotIn", "values": ["dev"]}], "type": "array", "items": {" 1649 1650 \$ref": "#/definitions/SetBasedRequirements"}}, "annotationSelectors": {"title": 1651 Annotationselectors", "description": "Specifies the annotation-key/value pairs that the 1652 experiment's target Pod must have. If multiple annotations are specified, the experiment target must have all annotations specified by this selector.", "type": " 1653 object", "additionalProperties": {"type": "string"}}, "fieldSelectors": {"title": " 1654 Fieldselectors", "description": "Specifies the field-key/value pairs of the experiment' s target Pod. If multiple fields are specified, the experiment target must have all 1655 fields set by this selector.", "example": {"metadata.name": "my-pod", "metadata. 1656 namespace": "dafault"}, "type": "object", "additionalProperties": {"type": "string"}}, "podPhaseSelectors": {"title": "Podphaseselectors", "description": "Specifies the phase 1657 of the experiment's target Pod. If this selector is None, the target Pod's phase is 1658 not limited.", "type": "array", "items": {"enum": ["Pending", "Running", "Succeeded", " Failed", "Unknown"], "type": "string"}}, "nodeSelectors": {"title": "Nodeselectors", 1659 description": "Specifies the node-label-key/value pairs to which the experiment's 1660 target Pod belongs.", "type": "object", "additionalProperties": {"type": "string"}}, nodes": {"title": "Nodes", "description": "Specifies the node to which the experiment's 1661 target Pod belongs. The target Pod can only belong to one node in the configured node 1662 list. If multiple node labels are specified, the node to which the experiment's target Pod belongs must have all labels specified by this selector.", "type": "array", "items 1663 ": {"type": "string"}}, "pods": {"title": "Pods", "description": "Specifies the 1664 namespaces and list of the experiment's target Pods. If you have specified this selector, Chaos Mesh ignores other configured selectors.", "example": {"default": ["pod -0", "pod-2"]}, "type": "object", "additionalProperties": {"type": "array", "items": {" 1665 1666 type": "string"}}}}} 1667

- 1668
- 1669
- 1670 1671

1672

1673

detailed\_param\_instructions for HTTPChaos (template embedded dynamically)

The output should be formatted as a JSON instance that conforms to the JSON schema below.

1674 1675 As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a 1676 list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]} the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The 1677 object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted. 1678 Here is the output schema: 1679 1680 {"properties": {"mode": {"title": "Mode", "description": "Specifies the mode of the experiment. The mode options include one (selecting a random Pod), all (selecting all 1681 eligible Pods), fixed (selecting a specified number of eligible Pods), fixed-percent (1682 selecting a specified percentage of Pods from the eligible Pods), and random-maxpercent (selecting the maximum percentage of Pods from the eligible Pods)", "example": 1683 "one", "enum": ["one", "all", "fixed", "fixed-percent", "random-max-percent"], "type": "string"}, "value": {"title": "Value", "description": "Provides parameters for the mode 1684 configuration, depending on mode. For example, when mode is set to fixed-percent, 1685 value specifies the percentage of Pods.", "example": "1", "type": "string"}, "target": 1686 {"title": "Target", "description": "Specifies whether the target of fault injection is Request or Response. The target-related fields (replace.path, replace.method, replace. 1687 "Request", queries, patch queries) should be configured at the same time.", "example": 1688 enum": ["Request", "Response"], "type": "string"}, "port": {"title": "Port", description": "The TCP port that the target service listens on.", "example": 80, "type 1689 ": "integer"}, "code": {"title": "Code", "description": "Specifies the status code 1690 responded by target. If not specified, the fault takes effect for all status codes by default. This configuration is effective only when the 'target' is set to 'Response'", "example": 200, "type": "integer", "path": {"title": "Path", "description": "Specify 1691 1692 the URI path of the target request. Supports Matching wildcards. If not specified, the fault takes effect on all paths by default.", "example": "/api/\*", "type": "string"}, 1693 method": {"title": "Method", "description": "Specify the HTTP method of the target 1694 request method. If not specified, the fault takes effect for all methods by default." "example": "GET", "type": "string"}, "request\_headers": {"title": "Request Headers", 1695 "example": "GET", "type": "string"}, "request\_headers": {"title": "Request Headers", "
description": "Matches request headers to target.", "example": {"Content-Type": "
application/json"}, "type": "object", "additionalProperties": {"type": "string"}}, "
abort": {"title": "Abort", "description": "Abort fault. Indicates whether to inject the
fault that interrupts the connection.", "default": false, "example": true, "type": "
boolean"}, "delay": {"title": "Delay", "description": "Deplay fault. Specifies the time
for a latency fault.", "default": "Down, "upper: "latency fault.", "replace":
("title": "Delay", "Deplace fault.", "type": "string"}, "replace": 1696 1697 1698 1699 {"title": "Replace", "description": "Replace fault. Specifies replaced contents.", " 1700 allOf": [{"\$ref": "#/definitions/Replace"}]}, "patch": {"title": "Patch", "description
": "Patch fault. Specifies patch contents.", "allOf": [{"\$ref": "#/definitions/Patch 1701 ": "Patch fault. Specifies patch contents.", "allof": [{"\$ref": "#/definitions/Patch
"}]}}, "required": ["mode", "target", "port"], "definitions": {"Replace": {"title": "
Replace", "type": "object", "properties": {"headers": {"title": "Headers", "description 1702 ": "Specifies the key pair used to replace the request headers or response headers.", " example": {"Content-Type": "application/xml"}, "type": "object", "additionalProperties 1703 1704 ": {"type": "string"}}, "body": {"title": "Body", "description": "Specifies request body or response body to replace the fault (Base64 encoded).", "example": " 1705 eyJmb28iOiAiYmFyInOK", "type": "string"}, "path": {"title": "Path", "description": " Specifies the URI path used to replace content.", "example": "/api/v2", "type": "string "}, "method": {"title": "Method", "description": "Specifies the replaced content of the HTTP request method.", "example": "DELETE", "type": "string"}, "queries": {"title": " Queries", "description": "Specifies the replaced key pair of the URI query.", "type": " array", "items": {"type": "array", "items": {"type": "string"}}, "code": {"title": " 1706 1707 1708 1709 Code", "description": "Specifies the replaced content of the response status code. This 1710 configuration is effective only when the 'target' is set to 'Response'.", "example": 404, "type": "integer"}}, "PatchBody": {"title": "PatchBody", "type": "object", " 1711 properties": {"type": {"title": "Type", "description": "Specifies the type of patch 1712 faults of the request body or response body. Currently, it only supports JSON.", example": "JSON", "type": "string"}, "value": {"title": "Value", "description": " 1713 Specifies the fault of the request body or response body with patch faults.", "example 1714 ": "{\"foo\": \"bar\"}", "type": "string"}}, "Patch": {"title": "Patch", "type": "
object", "properties": {"headers": {"title": "Headers", "description": "Specifies the 1715 attached key pair of the request headers or response headers with patch faults.", " 1716 example": [["Set-Cookie", "one cookie"]], "type": "array", "items": {"type": "array", items": {"type": "string"}}}, "body": {"title": "Body", "description": "Patch body.", " 1717 allof": [{"\$ref": "#/definitions/PatchBody"}]}, "queries": {"title": "Queries", 1718 description": "Specifies the attached key pair of the URI query with patch faults.", " example": [["foo", "bar"]], "type": "array", "items": {"type": "array", "items": {"type 1719 ": "string"}}}}} 1720 1721 1722 1723 detailed\_param\_instructions for StressChaos (template embedded dynami-1724 cally)

1726The output should be formatted as a JSON instance that conforms to the JSON schema1727below.

1728 1729 As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a 1730 list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]} the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The 1731 object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted. 1732 Here is the output schema: 1733 1734 {"properties": {"mode": {"title": "Mode", "description": "Specifies the mode of the experiment. The mode options include 'one' (selecting a random Pod), 'all' (selecting 1735 all eligible Pods), 'fixed' (selecting a specified number of eligible Pods), 'fixed-1736 (selecting a specified percentage of Pods from the eligible Pods), and 'random percent' -max-percent' (selecting the maximum percentage of Pods from the eligible Pods)", ' 1737 example": "one", "enum": ["one", "all", "fixed", "fixed-percent", "random-max-percent
"], "type": "string"), "value": {"title": "Value", "description": "Provides parameters 1738 for the mode configuration, depending on mode. For example, when mode is set to fixed-1739 percent, value specifies the percentage of Pods.", "example": "1", "type": "string"}, " stressors": {"title": "Stressors", "description": "Specifies the stress of CPU or 1740 memory", "dafault": null, "allof": [{"\$ref": "#/definitions/Stressors"}]}, 1741 stressngStressors": {"title": "Stressngstressors", "description": "Specifies the stres-1742 ng parameter to reach richer stress injection", "example": "--clone 2, "type": "string "}, "containerNames": {"title": "Containernames", "description": "Specifies the name of the container into which the fault is injected.", "example": ["nginx"], "type": "array 1743 1744 ", "items": {"type": "string"}}, "selector": {"title": "Selector", "description": " Specifies the target Pod.", "allof": [{"\$ref": "#/definitions/Selectors"}]}}, "required 1745 : ["mode", "selector"], "definitions": {"MemoryStressor": {"title": "MemoryStressor", 1746 "type": "object", "properties": {"workers": {"title": "Workers", "description": ' Specifies the number of threads that apply memory stress", "example": 1, "type": " 1747 integer"}, "size": {"title": "Size", "description": "Specifies the memory size to be integer"}, "size": {"title": "Size", "description": "Specifies the memory size to be occupied or a percentage of the total memory size. The final sum of the occupied memory size is size.", "example": "256MB", "type": "string"}, "oomScoreAdj": {"title": " Oomscoreadj", "description": "Specifies the oom\_score\_adj of the stress process.", " example": -1000, "type": "integer"}}, "CPUStressor": {"title": "CPUStressor", "type": "object", "properties": {"workers": {"title": "Workers", "description": "Specifies the number of threads that apply CPU stress", "example": 1, "type": "integer"}, "load": {" title": "Load", "description": "Specifies the percentage of CPU screes"} 1748 1749 1750 1751 title": "Load", "description": "Specifies the percentage of CPU occupied. 0 means that 1752 no additional CPU is added, and 100 refers to full load. The final sum of CPU load is workers \* load.", "example": 50, "type": "integer"}}, "Stressors": {"title": " 1753 Stressors", "type": "object", "properties": {"memory": {"title": "Memory", "description ": "Specifies the memory stress", "allof": [{"\$ref": "#/definitions/MemoryStressor"}]}, 1754 1755 "cpu": {"title": "Cpu", "description": "Specifies the CPU stress", "allof": [{"\$ref": "#/definitions/CPUStressor"}]}}, "SetBasedRequirements": {"title": " 1756 SetBasedRequirements", "type": "object", "properties": {"key": {"title": "Key", " 1757 description": "Label key", "type": "string"}, "operator": {"title": "Operator", " description": "Select an operator.", "enum": ["In", "NotIn", "Exists", "DoesNotExist"], 1758 "type": "string"}, "values": {"title": "Values", "description": "Label values. The 1759 values set must be non-empty in the case of In and NotIn.", "type": "array", "items": 1760 {"type": "string"}}}, "required": ["key", "operator", "values"]}, "Selectors": {"title ": "Selectors", "type": "object", "properties": {"namespaces": {"title": "Namespaces", 1761 "description": "Specifies the namespace of the experiment's target Pod. If this 1762 selector is None, Chaos Mesh will set it to the namespace of the current Chaos experiment.", "type": "array", "items": {"type": "string"}}, "labelSelectors": {"title 1763 ": "Labelselectors", "description": "Specifies the label-key/value pairs that the 1764 experiment's target Pod must have. If multiple labels are specified, the experiment target must have all the labels specified by this selector.", "type": "object", 1765 additionalProperties": {"type": "string"}}, "expressionSelectors": {"title": " 1766 Expressionselectors", "description": "Specifies a set of expressions that define the label's rules to specifiy the experiment's target Pod.", "example": [{"key": "tier", "
operator": "In", "values": ["cache"]}, {"key": "environment", "operator": "NotIn", "
values": ["dev"]}], "type": "array", "items": {"\$ref": "#/definitions/ 1767 1768 SetBasedRequirements"}, "annotationSelectors": {"title": "Annotationselectors", " 1769 description": "Specifies the annotation-key/value pairs that the experiment's target 1770 Pod must have. If multiple annotations are specified, the experiment target must have all annotations specified by this selector.", "type": "object", "additionalProperties": 1771 {"type": "string"}}, "fieldSelectors": {"title": "Fieldselectors", "description": " 1772 Specifies the field-key/value pairs of the experiment's target Pod. If multiple fields are specified, the experiment target must have all fields set by this selector.", 1773 example": {"metadata.name": "my-pod", "metadata.namespace": "dafault"}, "type": ", "additionalProperties": {"type": "string"}}, "podPhaseSelectors": {"title": " "type": "object 1774 Podphaseselectors", "description": "Specifies the phase of the experiment's target Pod. 1775 Podphaseselectors", "description": "specifies the phase of the experiment of the second specifies and the second specifie 1776 items": {"enum": ["Pending", "Running", "Succeded", "Failed", "Unknown"], "type": "
string"}, "nodeSelectors": {"title": "Nodeselectors", "description": "Specifies the 1777 node-label-key/value pairs to which the experiment's target Pod belongs.", "type": " 1778 object", "additionalProperties": {"type": "string"}}, "nodes": {"title": "Nodes", " description": "Specifies the node to which the experiment's target Pod belongs. The 1779 target Pod can only belong to one node in the configured node list. If multiple node 1780 labels are specified, the node to which the experiment's target Pod belongs must have 1781

1782 all labels specified by this selector.", "type": "array", "items": {"type": "string"}}, 1783 "pods": {"title": "Pods", "description": "Specifies the namespaces and list of the experiment's target Pods. If you have specified this selector, Chaos Mesh ignores other 1784 configured selectors.", "example": {"default": ["pod-0", "pod-2"]}, "type": "object", 1785 "additionalProperties": {"type": "array", "items": {"type": "string"}}}}} 1786 1787 1788 1789 detailed\_param\_instructions for IOChaos (template embedded dynamically) 1790 1791 The output should be formatted as a JSON instance that conforms to the JSON schema 1792 below. 1793 As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a 1794 list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]} the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The 1795 object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted. 1796 Here is the output schema: 1797 1798 {"properties": {"action": {"title": "Action", "description": "Indicates the specific type of faults. Only latency, fault, attrOverride, and mistake are supported.", 1799 example": "latency", "enum": ["latency", "fault", "attrOverride", "mistake"], "type": "
string"}, "mode": {"title": "Mode", "description": "Specifies the mode of the 1800 experiment. The mode options include one (selecting a random Pod), all (selecting all 1801 eligible Pods), fixed (selecting a specified number of eligible Pods), fixed-percent (1802 selecting a specified percentage of Pods from the eligible Pods), and random-maxpercent (selecting the maximum percentage of Pods from the eligible Pods)", "example": 1803 one", "enum": ["one", "all", "fixed", "fixed-percent", "random-max-percent"], "type": 1804 "string"}, "selector": {"title": "Selector", "description": "Specifies the target Pod ", "allOf": [{"\$ref": "#/definitions/Selectors"}]}, "value": {"title": "Value", 1805 description": "Provides parameters for the mode configuration, depending on mode. For 1806 example, when mode is set to fixed-percent, value specifies the percentage of Pods.", " example": "1", "type": "string"}, "volumePath": {"title": "Volumepath", "description": 1807 "The mount point of volume in the target container. Must be the root directory of the 1808 mount.", "example": "/var/run/etcd", "type": "string"}, "path": {"title": "Path", "
description": "The valid range of fault injections, either a wildcard or a single file. 1809 If not specified, the fault is valid for all files by default", "example": "/var/run/ etcd/\*/", "type": "string"}, "methods": {"title": "Methods", "description": "Type of 1810 etcd/\*/", "type": "string"}, "methods": {"title": "Methods", "description": "Type of the file system call that requires injecting fault. Supported method types: ['lookup', 'forget', 'getattr', 'setattr', 'readlink', 'mknod', 'mkdir', 'unlink', 'rmdir', ' symlink', 'rename', 'link', 'open', 'read', 'write', 'flush', 'release', 'fsync', ' opendir', 'readdir', 'releasedir', 'fsyncdir', 'statfs', 'setxattr', 'getxattr', ' listxattr', 'removexattr', 'access', 'create', 'getlk', 'setlk', 'bmap']. All Types by default.", "example": ["READ"], "type": "array", "items": {"type": "string"}, "percent ": {"title": "Percent", "description": "Probability of failure per operation, in %.", " default": 100, "example": 100, "type": "integer"}, "containerNames": {"title": " Containernames", "description": "Specifies the name of the container into which the fault is injected.", "type": "array", "items": {"type": "string"}. "deplay": {"title": "approximation": "type": "array", "items": "string"}. "deplay: "title": "specifies the name of the container into which the fault is injected.", "type": "array", "items": {"type": "string"}. "deplay: "title": "specifies the name of the container into which the fault is injected.", "type": "array", "items": {"type": "string"}. "deplay: "title": "specifies the name of the container into which the fault is injected.", "type": "array", "items": {"type": "string"}. "deplay: "title": "specifies the name of the container into which the fault is injected.", "type: "string"}. 1811 1812 1813 1814 1815 1816 fault is injected.", "type": "array", "items": {"type": "string"}}, "deplay": {"title": 1817 "Deplay", "description": "Specify when the 'action' is set to 'latency'. Specific 1818 delay time.", "type": "string"}, "errno": {"title": "Errno", "description": "Specify when the 'action' is set to 'fault'. Returned error number: 1: Operation not permitted, 1819 2: No such file or directory, 5: I/O error, 6: No such device or address, 12: Out of 1820 memory, 16: Device or resource busy, 17: File exists, 20: Not a directory, 22: Invalid argument, 24: Too many open files, 28: No space left on device", "type": "integer"}, ' 1821 attr": {"title": "Attr", "description": "Specify when the 'action' is set to ' 1822 attrOverride'. Specific property override rules.", "allOf": [{"\$ref": "#/definitions/ AttroverrideSpec"}]), "mistake": {"title": "Mistake", "description": "Specify when the 'action' is set to 'mistake'. Specific error rules.", "allOf": [{"\$ref": "#/definitions 1823 /MistakeSpec"}]}, "required": ["action", "mode", "volumePath", "attr", "mistake"], "
definitions": {"SetBasedRequirements": {"title": "SetBasedRequirements", "type": " 1824 1825 object", "properties": {"key": {"title": "Key", "description": "Label key", "type": " 1826 string"}, "operator": {"title": "Operator", "description": "Select an operator.", "enum
": ["In", "NotIn", "Exists", "DoesNotExist"], "type": "string"}, "values": {"title": " 1827 Values", "description": "Label values. The values set must be non-empty in the case of 1828 In and NotIn.", "type": "array", "items": {"type": "string"}}}, "required": ["key", operator", "values"]}, "Selectors": {"title": "Selectors", "type": "object", 1829 properties": {"namespaces": {"title": "Namespaces", "description": "Specifies the 1830 namespace of the experiment's target Pod. If this selector is None, Chaos Mesh will set it to the namespace of the current Chaos experiment.", "type": "array", "items": {" 1831 type": "string"}}, "labelSelectors": {"title": "Labelselectors", "description": " 1832 Specifies the label-key/value pairs that the experiment's target Pod must have. If multiple labels are specified, the experiment target must have all the labels specified "type": "object", "additionalProperties": {"type": "string"}}, " by this selector.", expressionSelectors": {"title": "ExpressionSelectors", "description": "Specifies a set of expressions that define the label's rules to specifiy the experiment's target Pod.", 1834 1835

	"example": [{"key": "tier", "operator": "In", "values": ["cache"]}, {"key": "
	<pre>environment", "operator": "NotIn", "values": ["dev"]}], "type": "array", "items": {"</pre>
	<pre>\$ref": "#/definitions/setBasedRequirements"}}, "annotationselectors": {"title": " Annotationselectors", "description": "Specifies the annotation-key/value pairs that the</pre>
	experiment's target Pod must have. If multiple annotations are specified, the
	experiment target must have all annotations specified by this selector.", "type": "
	object", "additionalProperties": {"type": "string"}}, "fieldSelectors": {"title": " FieldSelectors" "description": "Spacifies the field-key(value pairs of the experiment/
	s target Pod. If multiple fields are specified, the experiment target must have all
	fields set by this selector.", "example": {"metadata.name": "my-pod", "metadata. namespace": "dafault"}, "type": "object", "additionalProperties": {"type": "string"}},
	"podPhaseSelectors": {"title": "PodphaseSelectors", "description": "Specifies the phase
	of the experiment's target Pod. If this selector is None, the target Pod's phase is
	<pre>not limited.", "type": "array", "items": {"enum": ["Pending", "Running", "Succeeded", " Failed", "Unknown"], "type": "string"}}, "nodeSelectors": {"title": "Nodeselectors", "</pre>
	description": "Specifies the node-label-key/value pairs to which the experiment's
	<pre>target Pod belongs.", "type": "object", "additionalProperties": {"type": "string"}}, " nodes": {"title": "Nodes" "description": "Specifies the node to which the experiment's</pre>
	target Pod belongs. The target Pod can only belong to one node in the configured node
	list. If multiple node labels are specified, the node to which the experiment's target Pod belongs must have all labels specified by this selector.", "type": "array", "items
	": {"type": "string"}}, "pods": {"title": "Pods", "description": "Specifies the
	namespaces and list of the experiment's target Pods. If you have specified this
	-0", "pod-2"]}, "type": "object", "additionalProperties": {"type": "array", "items": {"
	type": "string"}})}), "TimeSpec": {"title": "TimeSpec", "type": "object", "properties":
	{"sec": {"title": "Sec", "description": "Timestamp in seconds. Specify either sec or
	<pre>nsec.", "type": "integer"; "nsec": {"title": "Nsec", "description": "limestamp in nanoseconds. Specify either sec or nsec ". "type". "integer"]}; "AttroverrideSpec". {"</pre>
	title": "AttrOverrideSpec", "type": "object", "properties": {"ino": {"title": "Ino", "
	description": "ino number", "type": "integer"}, "size": {"title": "Size", "description
	": "File size", "type": "integer"}, "Dlocks": {"title": "Blocks", "description": " Number of blocks that the file uses" "type". "integer"} "atime". {"title". "litime" "
	<pre>description": "Last access time", "allof": [{"\$ref": "#/definitions/TimeSpec"}]}, "</pre>
	<pre>mtime": {"title": "Mtime", "description": "Last modified time", "allof": [{"\$ref": "#/</pre>
	<pre>definitions/TimeSpec"}]}, "ctime": {"title": "Ctime", "description": "Last status change time" "allof": [/"Sref": "#/definitions/TimeSpec"}]} "kind": /"title": "Kind"</pre>
	"description": "File type, see fuser::FileType", "type": "string", "perm": {"title":
	"Perm", "description": "File permissions in decimal", "type": "integer"}, "nlink": {"
	title": "Nlink", "description": "Number of hard links", "type": "integer"}, "uid": {"
	title: "Gid, description: Group ID of the owner, type: Integer ;, gid: { title: "Gid, "description": "Group ID of the owner", "type: "integer"; "rdey": {"
	title": "Rdev", "description": "Device ID", "type": "integer"}}}, "MistakeSpec": {"
	title": "MistakeSpec", "type": "object", "properties": {"filling": {"title": "Filling",
	"description": "The wrong data to be filled. Only zero (fill U) or random (fill random bytes) are supported " "type": "string"} "mayOccurrences": {"title": "MayOccurrences"
	", "description": "Maximum number of errors in each operation.", "example": 1, "type":
	"integer"}, "maxLength": {"title": "Maxlength", "description": "Maximum length of each
	error (in bytes).", "example": 1, "type": "integer"}}, "required": ["filling", "
	maxucullences , maxiengun]}}
6	

1874	
1875	
1876	The output should be formatted as a JSON instance that conforms to the JSON schema below.
1877	
1878	As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]}
1879	the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The
1880	object (propercies . (100 . (bar , baz))) is not werr-tormatted.
1881	Here is the output schema:
1882	{"properties": {"timeOffset": {"title": "Timeoffset", "description": "Specifies the
1883	<pre>length of time offset.", "example": "-5m", "type": "string"}, "clockIds": {"title": "</pre>
1884	clock_gettime documentation for details.", "default": ["CLOCK_REALTIME"], "example": ["
1885	CLOCK_REALTIME", "CLOCK_MONOTONIC"], "type": "array", "items": {"type": "string"}}, " mode": {"title": "Mode" "description": "Specifies the mode of the experiment The mode
1886	options include 'one' (selecting a random Pod), 'all' (selecting all eligible Pods), '
1887	fixed' (selecting a specified number of eligible Pods), 'fixed-percent' (selecting a specified percentage of Pods from the eligible Pods), and 'random-max-percent' (
1888	selecting the maximum percentage of Pods from the eligible Pods)", "example": "one", "
1889	enum": ["one", "all", "fixed", "fixed-percent", "random-max-percent"], "type": "string

1890	
891	"}, "value": {"title": "Value", "description": "Provides parameters for the mode
892	configuration, depending on mode. For example, when mode is set to fixed-percent, value specifies the percentage of Pods ". "example": "1". "type": "string"}. "containerNames
893	": {"title": "Containernames", "description": "Specifies the name of the container into
894	<pre>which the fault is injected.", "example": ["nginx"], "type": "array", "items": {"type ": "string"}}, "selector": {"title": "Selector", "description": "Specifies the target</pre>
895	<pre>Pod.", "example": null, "allOf": [{"\$ref": "#/definitions/Selectors"}]}}, "required":</pre>
896	["timeOffset", "mode", "selector"], "definitions": {"SetBasedRequirements": {"title": " SetBasedRequirements", "type": "object", "properties": {"key": {"title": "Key", "
897	description": "Label key", "type": "string"}, "operator": {"title": "Operator", "
398	<pre>description": "Select an operator.", "enum": ["In", "NotIn", "Exists", "DoesNotExist"], "type": "string"}. "values": {"title": "Values". "description": "Label values The</pre>
399	values set must be non-empty in the case of In and NotIn.", "type": "array", "items":
900	<pre>{"type": "string"}}}, "required": ["key", "operator", "values"]}, "Selectors": {"title ". "Selectors", "type", "ebject", "presenties", ("presenties", ("title", "Nervenages")</pre>
01	"description": "Specifies the namespace of the experiment's target Pod. If this
02	selector is None, Chaos Mesh will set it to the namespace of the current Chaos
02	experiment.", "type": "array", "items": {"type": "string"}}, "labelSelectors": {"title ": "Labelselectors", "description": "Specifies the label-key/value pairs that the
03	experiment's target Pod must have. If multiple labels are specified, the experiment
04	target must have all the labels specified by this selector.", "type": "object", "
05	Expressionselectors", "description": "Specifies a set of expressions that define the
06	label's rules to specifiy the experiment's target Pod.", "example": [{"key": "tier", "
07	<pre>operator": "In", "values": ["cache"]}, {"key": "environment", "operator": "NotIn", " values": ["dow"]]], "type": "array", "items": ("\$raf": "#/definitions/</pre>
80	SetBasedRequirements"}}, "annotationSelectors": {"title": "Annotationselectors", "
09	description": "Specifies the annotation-key/value pairs that the experiment's target
10	Pod must have. If multiple annotations are specified, the experiment target must have all annotations specified by this selector ", "type": "object", "additionalProperties":
11	<pre>{"type": "string"}}, "fieldSelectors": {"title": "Fieldselectors", "description": "</pre>
12	Specifies the field-key/value pairs of the experiment's target Pod. If multiple fields
13	example": {"metadata.name": "my-pod", "metadata.namespace": "dafault"}, "type": "object
1.1	", "additionalProperties": {"type": "string"}}, "podPhaseSelectors": {"title": "
	Podphaseselectors", "description": "Specifies the phase of the experiment's target Pod.
C	items": {"enum": ["Pending", "Running", "Succeeded", "Failed", "Unknown"], "type": "
16	<pre>string"}}, "nodeSelectors": {"title": "Nodeselectors", "description": "Specifies the</pre>
7	node-Label-key/value pairs to which the experiment's target Pod belongs.", "type": "
18	description": "Specifies the node to which the experiment's target Pod belongs. The
19	target Pod can only belong to one node in the configured node list. If multiple node
20	abels are specified, the node to which the experiment's target Pod belongs must have all labels specified by this selector.", "type": "array", "items": {"type": "string"}},
21	"pods": {"title": "Pods", "description": "Specifies the namespaces and list of the
22	experiment's target Pods. If you have specified this selector, Chaos Mesh ignores other configured selectors.", "example": {"default": ["pod-0", "pod-2"]}. "type": "object".
23	"additionalProperties": {"type": "array", "items": {"type": "string"}}}}}
24	

B.1.3 EXPERIMENT

2-0: Agent for determining time schedule
System:
You are a helpful AI assistant for Chaos Engineering.
Given k8s manifests that define a network system, its steady states, and faults that
may affect the steady states in the system, you will design a Chaos Engineering experiment for them.
First, you will determine the time schedule for the Chaos Engineering experiment.
Always keep the following rules: - The experiment is divided into three phases: pre-validation, fault-injection, and
post-validation phases: pre-validation to ensure that the system satisfies the stead
states fault injection; fault-injection to observe the system's behavior during fault injection; post-validation to ensure that the system has returned to its steady state
after fault injection.
- The output should be formatted as a JSON instance that conforms to the JSON schema
Detow.
As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
<pre>description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \"</pre>
<pre>string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a</pre>
1944 well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\", 1945 \"baz\"]}} is not well-formatted. 1946 Here is the output schema: 1947 1948 {\"properties\": {\"thought\": {\"Thought\", \"describe\": \"Think about the total time and the reasonable time allocation for each phase that you are about to 1949 design, and explain your thought process in detail.\", \"type\": \"string\"}, \" 1950 total\_time\": {\"title\": \"Total Time\", \"description\": \"Total time of the entire chaos experiment. total\_time should equal to the sum of pre\_validation\_time, 1951 fault\_injection\_time, and post\_validation\_time.\", \"example\": \"10m\", \"type\": \"
string\"}, \"pre\_validation\_time\": {\"title\": \"Pre Validation Time\", \"description 1952 \": \"Total time of validation before fault injection.\", \"example\": \"2m\", \"type \": \"string\"}, \"fault\_injection\_time\": {\"title\": \"Fault Injection Time\", \" 1953 1954 description\": \"Total time of fault injection.\", \"example\": \"6m\", \"type\": \"
string\"}, \"post\_validation\_time\": {\"title\": \"Post Validation Time\", \" 1955 description\": \"Total time of validation after fault injection.\", \"example\": \"2m 1956 \", \"type\": \"string\"}}, \"required\": [\"thought\", \"total\_time\", \" pre\_validation\_time\", \"fault\_injection\_time\", \"post\_validation\_time\"]} 1957 1958 Human: 1959 # Here is the overview of my system: 1960 {user input2} 1961 # Steady states of my system: 1962 {steady\_states} 1963 # A fault scenario that may occur in my system and may affect the steady states: 1964 {detailed fault scenario} 1965 # Please follow the instructions below regarding Chaos Engineering as necessary: 1966 {ce instructions} 1967 Now, please plan a Chaos Engineering experiment to check the network system's 1968 resiliency that the steady states are remained during fault injection. 1969 1970 1971 Example text embedded to **detailed\_fault\_scenario** 1972 1973 An assumed fault scenario is as follows: 1974 - Event: Cyber Attack Simulation\n- Used Chaos Engineering tool: Chaos Mesh - Faults to simulate the event: [[Fault(name='PodChaos', name\_id=0, params={'action': 'pod-kill', 'mode': 'one', 'selector': {'namespaces': ['default'], 'labelSelectors': {' 1975 1976 app': 'example'}})], [Fault(name='NetworkChaos', name\_id=0, params={'action': 'delay', 'mode': 'all', 'selector': {'namespaces': ['default'], 'labelSelectors': {'app': ' 1977 example'}}, 'direction': 'to', 'delay': {'latency': '100ms', 'jitter': '10ms'}})]] 1978 - Description: Given the system's weaknesses, a cyber attack targeting the web server 1979 could be highly impactful. The Pod's restart policy set to 'Never' and the single Pod deployment without redundancy are critical vulnerabilities. If the Pod fails, it will 1980 not restart, leading to downtime, and the lack of redundancy means there is no backup 1981 to handle traffic. To simulate a cyber attack, we can inject faults that exploit these weaknesses. First, we will use PodChaos to simulate a Pod failure, which will test the 1982 system's ability to maintain the 'example-pod-running-state'. Since the Pod will not 1983 restart automatically, this will directly impact the steady state. Next, we will use NetworkChaos to simulate network latency, which will test the system's ability to 1984 maintain the 'example-service-http-response-state'. This sequence simulates a cyber 1985 attack where the Pod is targeted first, followed by network disruptions, revealing the system's vulnerabilities in handling such events. 1986 1987 1988 1989 # 2-1: Agent for scheduling each experiment phase (pre-validation, failure-injection, and post-validation phases) 1990 1991 1992 System: You are a helpful AI assistant for Chaos Engineering. 1993 Given k8s manifests that define a network system, its steady states, and faults that

may affect the steady states in the system, you will design a Chaos Engineering
 experiment for them.
 The experiment is divided into three phases: pre-validation, fault-injection, and post-validation phases: pre-validation to ensure that the system satisfies the steady states
 fault injection; fault-injection to observe the system's behavior during fault

Here, you will detail the {phase\_name}.
Always keep the following rules:
- {phase\_planning\_instructions}

Here is the overview of my system:

Steady states of my system:

{detailed\_fault\_scenario}

1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
0050

1009

Example text embedded to phase\_name

pre-validation phase

after fault injection.

Human:

{user\_input}

{steady\_states}

{ce\_instructions}

phase total time}.

Example text embedded to phase\_total\_time

10s

phase\_planning\_instructions for the pre-validation and post-validation phases

injection; post-validation to ensure that the system has returned to its steady states

A fault scenario that may occur in my system and may affect the steady states:

Please follow the instructions below regarding Chaos Engineering as necessary:

Now, please detail the {phase\_name}. Note that the phase's total time is {

The output should be formatted as a JSON instance that conforms to the JSON schema below.

As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \" description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \" string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\", \"baz\"]} is not well-formatted.

Here is the output schema:

{\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Describe in detail the timeline for when each fault injection and each unit test (for verifying steady-state) will be executed. For example, explain which fault injections/unit tests will be executed simultaneously, and whether certain fault injections/unit tests will be executed at staggered timings. Additionally, explain the thought process that led you to this approach.', \"type\": \"string\"}, \"unit\_tests\": {\"title\": \"Unit Tests\", \"description\": \"The list of unit test schedule.\", \"type\": \"array\", \" items\": {\"foref\": \"#/definitions/UnitTest\"}}, \"required\": [\"thought\", \" unit\_tests\"], \"definitions\": {\"UnitTest\": {\"title\": \"UnitTest\", \"type\": \" object\", \"properties\": {\"name\": {\"title\": \"Name\", \"description\": \"Steady state name to be verified by a unit test.\", \"type\": \"string\"}, \"grace\_period\": {\"title\": \"Grace Period\", \"description\": \"Time elapsed from the start of the current phase to the beginning of the unit test.", \"example\": \"Ouration of the unit test. (grace\_period + duration) should not exceed the current phase's total time .\", \"example\": \"2m\", \"type\": \"string\"}, \"required\": [\"name\", \" grace\_period\", \"duration\"]}}

2052	phase_planning_instructions for the fault-injection phases
2055	
2054	The output should be formatted as a JSON instance that conforms to the JSON schema below.
2056	As an example, for the schema {\"properties\", {\"foo\", {\"title\", \"Foo\", \"
2057	<pre>description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \"</pre>
2058	<pre>string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\".</pre>
2059	<pre>\"bac\"]}} is not well-formatted.</pre>
2060	Here is the output scheme.
2061	
2062	{\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Describe in
2063	steady-state) will be executed. For example, explain which fault injections/unit tests
2064	will be executed simultaneously, and whether certain fault injections/unit tests will
2065	be executed at staggered timings. Additionally, explain the thought process that led vou to this approach.\", \"type\": \"string\"}, \"fault injection\": {\"title\": \"
2066	Fault Injection\", \"description\": \"The list of fault injection schedules.\", \"type
2067	\": \"array\", \"items\": {\"\$ref\": \"#/definitions/FaultInjection\"}}, \"unit_tests \": {\"title\": \"Unit Tests\", \"description\": \"The list of unit test schedule.\",
2068	<pre>\"type\": \"array\", \"items\": {\"\$ref\": \"#/definitions/UnitTest\"}}, \"required\":</pre>
2069	[\"thought\", \"fault_injection\", \"unit_tests\"], \"definitions\": {\"FaultInjection
2070	{\"title\": \"Name\", \"description\": \"Select a fault type from [\\\"PodChaos\\\",
2071	\\\"NetworkChaos\\\", \\\"DNSChaos\\\", \\\"HTTPChaos\\\", \\\"StressChaos\\\", \\\"
2072	DNSChaos\", \"HTTPChaos\", \"StressChaos\", \"IOChaos\", \"TimeChaos\"], \"type\": \"
2073	string\"}, \"name_id\": {\"title\": \"Name Id\", \"description\": \"An identifier to
2074	sequential order to prevent name conflicts./", \"type\": \"integer\"}, \"grace_period
2075	\": {\"title\": \"Grace Period\", \"description\": \"Time elapsed from the start of the
2076	<pre>\": \"string\"}, \"duration\": {\"title\": \"Duration\", \"example\": \"Duration\", \"example\": \"Duration\", \"duration\": \"Duration of</pre>
2077	the unit test. (grace_period + duration) should not exceed the current phase's total
2078	<pre>name_id\", \"grace_period\", \"duration\"]}, \"UnitTest\": {\"title\": \"UnitTest\", \"</pre>
2079	type\": \"object\", \"properties\": {\"name\": {\"title\": \"Name\", \"description\":
2080	<pre>\"steady state name to be verified by a unit test.\", \"type\": \"string\"}, \" grace_period\": {\"title\": \"Grace Period\", \"description\": \"Time elapsed from the</pre>
2081	start of the current phase to the beginning of the unit test.\", \"example\": \"0s\",
2082	Duration of the unit test. (grace_period + duration) should not exceed the current
2083	<pre>phase's total time.\", \"example\": \"2m\", \"type\": \"string\"}}, \"required\": [\" prove \", \"understand \", \"duration \", \"</pre>
2084	<pre>name\", \"grace_period\", \"duration\"j}}}</pre>
2085	

2-2: Agent for summarizing the planned experiment

System:

	System.
2090	You are a helpful AI assistant for Chaos Engineering.
2091	Given a Chaos-Engineering-experiment plan, you will summarize it in detail according to
2092	the following rules: - In each phase, describe in detail the timeline for when each fault injection/unit
2093	test (for verifying steady-state) will be executed. For example, summarize which fault
2094	injections/unit tests will be executed simultaneously, and whether certain fault injections/unit tests will be executed at staggered timings.
2095	- Be sure to specify both each fault injection/unit test and their corresponding
2096	Workliow names. - When explaining the timeline, provide a detailed description using specific values
2097	for duration, grace period, etc. Rephrase the specific values in a way that everyone
2098	can easily understand. - The meanings of each value are as follows:
2099	- Grace Period: Time elapsed from the start of the current phase to the beginning of
2100	the fault injection/unit test. - Duration: Duration of the fault injection/unit test. (grace period + duration)
2101	should not exceed the corresponding phase's total time.
2102	- Never output bullet points. - The output should be formatted as a JSON instance that conforms to the JSON schema
2103	below.
2104	As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
2105	<pre>string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a</pre>

2106	
2107	well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\",
2108	<pre>\"baz\"]}} is not well-formatted.</pre>
2109	Here is the output schema:
2110	()
2111	of the given Chaos-Engineering-experiment plan.\", \"type\": \"string\"}}, \"required
2112	\": [\"summary\"]}
2112	
2113	Human:
2114	# Here is my Chaos-Engineering-experiment plan: ## Time Schedule
2115	{time_schedule_overview}
2110	## Pre-validation Phase
2117	{pre_validation_overview}
2118	## Equit injection Dhace
2119	<pre>## Fault_injection_overview}</pre>
2120	
2121	<pre>## Post-validation phase {post validation overview}</pre>
2122	
2123	Please summarize the above plan.
2124	
2125	
2126	Example text embedded to time_schedule_overview
2127	
2128	Given the constraints of the experiment needing to be completed within 1 minute, we
2129	need to carefully allocate time to each phase to ensure that we can effectively validate the system's steady states before and after the fault injection, as well as
2130	observe the system's behavior during the fault injection. The pre-validation phase is
2131	crucial to establish a baseline that the system is in its expected steady state before
2132	observe how the system behaves under stress. Finally, the post-validation phase is
2133	necessary to ensure that the system returns to its steady state after the faults are
2134	seconds for pre-validation, 40 seconds for fault injection, and 10 seconds for po
2135	validation. This allocation allows us to have a brief but sufficient observation period
2136	for each phase, ensuring that we can gather meaningful insights from the experiment.
2137	
2138	Example text embedded to pro validation everyiew
2139	
2140	
2141	In the pre-validation phase, we need to ensure that the system is in its expected
21/12	seconds to perform these checks. We have two steady states to verify: the 'example-pod
2142	-running-state' and the 'example-service-http-response-state'.
21//	The 'example-pod-running-state' requires us to check that the Pod is in the 'Running'
2174	state at least 90% of the time. We will use the provided Python script to verify this.
214J	Since the script checks the Fod status every second, we can run it for 5 seconds to gather sufficient data for validation.
2140	
2147	The 'example-service-http-response-state' requires us to ensure that 95% of HTTP requests to requests to simulate HTTP requests to
2148	the service. The script is configured to run for 5 seconds with 5 virtual users, which
2149	should provide enough data to validate this steady state.
2150	Both unit tests will be executed simultaneously to maximize the use of the 10-second
2151	window. This approach ensures that we efficiently validate both steady states within the given time constraint, allowing us to proceed confidently to the fault injection
2152	phase.
2153	- Verified Steady State #0: ```example-pod-running-state```
2154	- workitow Name:pre-unittest-example-pod-running-state - Grace Period: ```Os```
2155	- Duration: ```5s```
2156	- Verified Steady State #1: ```example-service-http-response-state``` - Workflow Name: ```pre-unittest-example-service-http-response-state```
2157	- Grace Period: ```Os```
2158	- Duration: '''5s'''
0150	

2160	Example text embedded to fault_injection_overview
2161	
2162	In this fault-injection phase, we aim to simulate a suber attack by injecting two types
2163	of faults: PodChaos and NetworkChaos. The goal is to observe how these faults impact
2164	the system's steady states. Given the 40-second time constraint, we will stagger the
2165	fault injections to maximize the observation of their effects.
2166	First, we will inject the PodChaos fault to simulate a Pod failure. This will directly
2167	to 'Never', meaning it will not automatically recover. We will start this fault
2168	injection immediately at the beginning of the phase and let it run for 20 seconds.
2169	Simultaneously, we will run the unit test for the 'example-pod-running-state' to verify
2170	the impact of the PodChaos fault on the Pod's availability.\n\nAfter 20 seconds, we
2171	example-service-http-response-state' steady state by introducing delays in HTTP
2172	responses. We will let this fault run for the remaining 20 seconds of the phase.
2173	During the NetworkChaos fault injection, we will also run the unit test for the '
2174	example-service-http-response-state' to verify the impact of network latency on the
2175	Service's ability to respond to hilf requests.
2176	This staggered approach allows us to isolate the effects of each fault on the respective steady states, providing clear insights into the system's unlocabilities.
2177	and resilience.
2178	- Verified Steady State #0: ```example-pod-running-state```
2179	- Grace Period: ```Os```
2180	- Duration: ```20s``` - Verified Steady State #1: ```evample-service-http-response-state```
2181	- Workflow Name: ```fault-unittest-example-service-http-response-state```
2182	- Grace Period: ```20s```
2183	Duración. 203
2184	- Injected Faults #0: ```PodChaos```
2185	- Grace Period: ```Os```
2186	- Duration: ```20s```
2187	- Injected Faults #1: ```NetworkChaos```
2188	- Workflow Name: ```fault-networkchaos``` - Grace Period: ```20s```
2189	- Duration: ```20s```
2190	
2191	
2192	Example text embedded to post_validation_overview
2193	
2194	In the nost-validation phase, we need to ensure that the system has returned to its
2195	steady states after the fault injection. Given the 10-second total time for this phase,
2196	we will conduct unit tests to verify each steady state. The tests will be executed sequentially due to the short duration, ensuring that each steady state is verified
2197	independently. The first test will check the 'example-pod-running-state', ensuring the
2198	Pod is running. The second test will verify the 'example-service-http-response-state', ensuring the Service responds with a 200 OK status. Each test will have a brief grace
2199	period to allow the system to stabilize after the fault injection, followed by a short
2200	duration to perform the checks. This approach ensures that both steady states are
2201	- Verified Steady State #0: ```example-pod-running-state```
2202	- Workflow Name: ```post-unittest-example-pod-running-state``` - Grace Period: ```ls```
2203	- Duration: ```4s```
2204	- Verified Steady State #1. \\\evample-service-http-response-state\\\
2205	- Workflow Name: ```post-unittest-example-service-http-response-state```
2206	- Grace Period: ```5s``` - Duration: ```4s```
2207	

2-3: Agent for adjusting a failure scope

 $\ensuremath{ \textbf{System:}}$ You are a helpful AI assistant for Chaos Engineering.

2214	
2215	Given a previous K8s manifests, a Chaos-Engineering-experiment plan for it, and the current K8s manifests, you will determine whether we need to adujst the scope of fault
2216	injections for the current K8s manifests.
2217	Always keep the following rules: - Cosider how you must change or keep the scope (i.e., target) of the fault injection
2218	comparing the previous K8s manifests and the current K8s manifests.
2219	- You only make minor adjustments related to resource changes, metadata change, etc, so
2220	NEVER make any scope changes that alter the original goal of the chaos experiment. - The output should be formatted as a JSON instance that conforms to the JSON schema
2221	below.
2222	As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
2223	<pre>description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \" string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a</pre>
2224	<pre>well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\", \"baz\"]}} is not well-formatted.</pre>
2225	Here is the output scheme:
2227	nere is the output schema:
2221	{\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Describe why
2228	<pre>you need to change/keep the scope of the fault injection for the current K8s manifests .\", \"type\": \"string\"}, \"selector\": {\"title\": \"Selector\", \"description\": \"</pre>
2223	Adjust the scope (target) of the fault injection comparing the differeneces between the
2230	<pre>current and previous manifests. \", \"allOf\": [{\"\$ref\": \"#/definitions/Selectors \"}]}}, \"required\": [\"thought\", \"selector\"], \"definitions\": {\"</pre>
2232	SetBasedRequirements\": {\"title\": \"SetBasedRequirements\", \"type\": \"object\", \"
2233	<pre>propercress : { / key : { / crcress : / "key / "; / "description /": / "Label Key /", / "type /":</pre>
2234	operator.\", \"enum\": [\"In\", \"NotIn\", \"Exists\", \"DoesNotExist\"], \"type\": \"
2235	values set must be non-empty in the case of In and NotIn.\", \"type\": \"array\", \"
2236	<pre>items\": {\"type\": \"string\"}}, \"required\": [\"key\", \"operator\", \"values\"]}, \"colortery\", \"title\", \"colortery\", \"values\"]},</pre>
2237	namespaces\": {\"title\": \"Namespaces\", \"description\": \"Specifies the namespace of
2238	the experiment's target Pod. If this selector is None, Chaos Mesh will set it to the
2239	<pre>\": \"string\"}}, \"labelSelectors\": {\"title\": \"Labelselectors\", \"description\":</pre>
2240	\"Specifies the label-key/value pairs that the experiment's target Pod must have. If
2241	by this selector.\", \"type\": \"object\", \"additionalProperties\": {\"type\": \"
2242	<pre>string\"}}, \"expressionSelectors\": {\"title\": \"Expressionselectors\", \"description \": \"Specifies a set of expressions that define the label's rules to specific the</pre>
2243	experiment's target Pod.\", \"example\": [{\"key\": \"tier\", \"operator\": \"In\", \"
2244	<pre>values\": [\"cache\"]}, {\"key\": \"environment\", \"operator\": \"NotIn\", \"values\": [\"dev\"]}], \"type\": \"array\", \"items\": {\"\$ref\": \"#/definitions/</pre>
2245	SetBasedRequirements\"}}, \"annotationSelectors\": {\"title\": \"Annotationselectors\",
2246	target Pod must have. If multiple annotations are specified, the experiment target must
2247	have all annotations specified by this selector.\", \"type\": \"object\", \"
2248	<pre>additionalProperties\": {\"type\": \"string\"}}, \"lieldselectors\": {\"title\": \" Fieldselectors\", \"description\": \"Specifies the field-key/value pairs of the</pre>
2249	experiment's target Pod. If multiple fields are specified, the experiment target must
2250	<pre>nave air rierds set by this serector.\", \"example\": {\"metadata.name\": \"my-pod\", \"metadata.namespace\": \"dafault\"}, \"type\": \"object\", \"additionalProperties\":</pre>
2251	<pre>{\"type\": \"string\"}}, \"podPhaseSelectors\": {\"title\": \"PodphaseSelectors\", \"</pre>
2252	<pre>aescription\": \"Specifies the phase of the experiment's target Pod. If this selector is None, the target Pod's phase is not limited.\", \"type\": \"array\", \"items\": {\"</pre>
2253	enum\": [\"Pending\", \"Running\", \"Succeeded\", \"Failed\", \"Unknown\"], \"type\":
2254	<pre>\"suring\"}}, \"nodeselectors\": {\"title\": \"Nodeselectors\", \"description\": \" Specifies the node-label-key/value pairs to which the experiment's target Pod belongs</pre>
2255	.\", \"type\": \"object\", \"additionalProperties\": {\"type\": \"string\"}}, \"nodes
2256	<pre>x : {\ cruce\": \"Nodes\", \"description\": \"Specifies the node to which the experiment's target Pod belongs. The target Pod can only belong to one node in the</pre>
2257	configured node list. If multiple node labels are specified, the node to which the
2258	experiment's target Pod belongs must have all labels specified by this selector.\", \" type\": \"array\", \"items\": {\"type\": \"string\"}}, \"pods\": {\"title\": \"Pods\",
2259	\"description\": \"Specifies the namespaces and list of the experiment's target Pods.
2260	If you have specified this selector, Chaos Mesh ignores other configured selectors.\", \"example\": {\"default\": [\"pod-0\", \"pod-2\"]}, \"type\": \"object\", \"
2261	additionalProperties\": {\"type\": \"array\", \"items\": {\"type\": \"string\"}}}}
2262	
2263	Human:
2264	<pre># Here is the previous K8s manifests of my system:</pre>
2265	(hreavealants)
2266	# Here is a planned Chaos Engineering:
	(experiment pian summary)

{experiment\_plan\_summary}

Here is the current K8s menifests of my system: {curr\_k8s\_yamls} $\ensuremath{\texttt{\#}}$ Here is the scope of a fault injection for the previous manifests. {curr\_fault\_injection} Now, please adjust the scope of the fault injection for the current manifests. Note that you here focus on the 'selector' parameter (i.e., scope). {format\_instructions}

2-4: Agent for adjusting a VaC script

2280	System:
2281	You are a helpful AI assistant for Chaos Engineering.
2282	state satisfies the threshold, and the reconfigured K8s manifests, you will determine
2283	whether the unit test requires adjustment to account for the changes in the
2284	reconfigured manifests, and adjust it as necessary. Always keep the following rules:
2285	- First, consider which K8s manifest resource is the target of the unit test. If there
2286	are changes to that manifest, update the unit test as necessary. If there are no changes, the unit test should not require modification.
2287	- You may only make minor adjustments to K8s API, HTTP, or DNS request to account for
2288	changes in resource types, parameter seetings, metadata, etc. - The reconfiguration was made so that the system satisfy the threshold value in the
2289	previous unit test, so the threshold value or other parameters must remain unchanged in
2290	to 3 in order to maintain a steady state with more than 1 active pod at all times. In
2291	such cases, changing the threshold value from 1 to 3 would alter the intent of th steady state, so the threshold value must remain unchanged (i.e., more than 1 act pod)."
2292	
2293	- If redundancy has been newly added, the unit test should verify whether the steady
2294	- If the unit test's content needs no changes and only function or variable names need
2295	to be changed, leave them as they are to save output costs.
2296	below.
2297	λ_{2} an example for the scheme (\"properties\", (\"fee\", (\"title\", \"Eee\", \"
2298	<pre>description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \"</pre>
2299	<pre>string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a well=formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\"]}</pre>
2300	<pre>\"bac\"]}} is not well-formatted.</pre>
2301	Here is the output scheme.
2302	
2303	{\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Describ your thought process for determining whether the unit test requires adjustment to account for the changes in the reconfigured manifests: First, consider which K8s
2304	
2305	manifest resource is the target of the unit test. If there are changes to that manifest , update the unit test as necessary. If there are no changes, the unit test should not
2306	require modification. If the unit test needs updating, describe also how you modify the
2307	inspection method according to the differences between the previous and reconfigured manifests. If the modification is not required, describe the reason.\", \"type\": \"
2308	<pre>string\"}, \"code\": {\"title\": \"Code\", \"description\": \"If the unit test needs</pre>
2309	updating, write a new unit test code with the inspection method modified. Write only the content of the code without enclosing it in a code block. If not, this field is not
2310	required.\", \"type\": \"string\"}}, \"required\": [\"thought\"]}
2311	
2312	Human:
2312 2313	Human: # Here is the previous K8s manifests of my system: { prev_k8s_yamls }
2312 2313 2314	Human: # Here is the previous K8s manifests of my system: {prev_k8s_yamls} # Here is the reconfigured K2s manifests of my system:
2312 2313 2314 2315	Human: # Here is the previous K8s manifests of my system: {prev_k8s_yamls} # Here is the reconfigured K8s manifests of my system: {curr_k8s_yamls}
2312 2313 2314 2315 2316	Human: # Here is the previous K8s manifests of my system: {prev_k8s_yamls} # Here is the reconfigured K8s manifests of my system: {curr_k8s_yamls} # Here is the unit test for the previous manifests
2312 2313 2314 2315 2316 2317	<pre>Human: # Here is the previous K8s manifests of my system: {prev_k8s_yamls} # Here is the reconfigured K8s manifests of my system: {curr_k8s_yamls} # Here is the unit test for the previous manifests. {prev_unittest}</pre>
2312 2313 2314 2315 2316 2317 2318	<pre>Human: # Here is the previous K8s manifests of my system: {prev_k8s_yamls} # Here is the reconfigured K8s manifests of my system: {curr_k8s_yamls} # Here is the unit test for the previous manifests. {prev_unittest} Now_please determine whether the unit test requires adjustment to account for the</pre>
2312 2313 2314 2315 2316 2317 2318 2319	<pre>Human: # Here is the previous K8s manifests of my system: {prev_k8s_yamls} # Here is the reconfigured K8s manifests of my system: {curr_k8s_yamls} # Here is the unit test for the previous manifests. {prev_unittest} Now, please determine whether the unit test requires adjustment to account for the changes in the reconfigured manifests, and adjust it as necessary.</pre>
2312 2313 2314 2315 2316 2317 2318 2319 2320	<pre>Human: # Here is the previous K8s manifests of my system: (prev_k8s_yamls) # Here is the reconfigured K8s manifests of my system: (curr_k8s_yamls) # Here is the unit test for the previous manifests. {prev_unittest} Now, please determine whether the unit test requires adjustment to account for the changes in the reconfigured manifests, and adjust it as necessary.</pre>

	In the verification loop, the prompts below will be stacked as history
	AI:
	{output}
	Ilumon .
	Your current unit test causes errors when conducted.
	The error message is as follows:
1	{error_message}
	This unit test should be succeeded.
I	Please analyze the reason why the errors occur, then fix the errors.
I	Always keep the following rules:
	- NEVER repeat the same fixes that have been made in the past. - Fix only the parts related to the errors without changing the original intent.
	- {format_instructions}
_	
	Example text embedded to experiment_plan_summary
	The Chaos Engineering experiment is structured into three phases: pre-validation, fau
	injection, and post-validation, all to be completed within a total of 1 minute.
	In the pre-validation phase, which lasts for 10 seconds, two unit tests are executed
	simultaneously to verify the system's steady states before any faults are introduced.
	ine 'example-pod-running-state' is checked using a Python script to ensure the Pod is in the 'Running' state at least 90% of the time. This test runs for 5 seconds
1	Concurrently, the 'example-service-http-response-state' is verified using a K6 script
	to simulate HTTP requests, ensuring 95% of requests return a 200 OK status. This test
ć	also runs for 5 seconds. Both tests start immediately at the beginning of the phase.
	The fault injection phase spans 40 seconds and involves two staggered fault injection
	Initially, the PodChaos fault is injected to simulate a Pod failure, running for the
-	first 20 seconds. Simultaneously, the 'example-pod-running-state' unit test is
C	is introduced to simulate network latency, running for the remaining 20 seconds.
Ι	During this period, the 'example-service-http-response-state' unit test is executed t
ė	assess the effect of network delays. This staggered approach allows for isolated
C	Socracion of each faute o impact on the system.
	Finally, the post-validation phase, lasting 10 seconds, ensures the system returns to
	its steady states after fault removal. The tests are conducted sequentially. The ' example-pod-running-state' is verified first, with a 1-second grace period followed b
	a 4-second test duration. Subsequently, the 'example-service-http-response-state' is
	checked, starting after a 5-second grace period and running for 4 seconds. This
	sequence confirms the system's recovery to its expected steady states.
	1.4 ANALYSIS
Ī	# 2.0. A gent for analyzing an appariment results
	# 5-0. Agent for analyzing an experiment results
	System:
	You are a helpful AI assistant for Chaos Engineering.
	Given Kös manifests for a network system, its hypothesis, the overview of a Chaos- Engineeering experiment, and the experimental results, you will analyze the
	experimental results.
	Always keep the following rules:
	- Analyze step by step why the test(s) failed, based on the system configurations (
	- Specify the cause while mentioning the corresponding system configurations and the
	corresponding phenomena in the Chaos-Engineering experiment.
	- The analysis report here will be used for reconfiguring the system later to avoid the following and improve analysis of the system of the sy
	indiffuences and improve resiliency. Therefore, make carefully the report rich in insign so that it will be helpful at that time.
	When providing indights and reconfiguration recommendations limit them to areas
	- when providing insights and reconfiguration recommendations, fimit them to areas
	related to the failed test.
	 When providing insights and reconfiguration recommendations, finit them to areas related to the failed test. The output should be formatted as a JSON instance that conforms to the JSON schema below.
	- When providing insights and reconfiguration recommendations, finit them to areas related to the failed test. - The output should be formatted as a JSON instance that conforms to the JSON schema below.
	As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"

```
2376
             string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a
2377
             well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar\",
2378
             \"baz\"]}} is not well-formatted.
2379
             Here is the output schema:
2380
             {\"properties\": {\"report\": {\"title\": \"Report\", \"description\": \"Analysis of
             the experiment result.\", \"type\": \"string\"}}, \"required\": [\"report\"]}
2382
             Human:
2384
             # Here is the overview of my system:
             {user_input2}
2385
2386
             # Here is the hypothesis for my system:
             The hypothesis is "The steady states of the sytem are maintained even when the fault
2387
             scenario occurs (i.e., when the faults are injected)".
2388
             The steady states here are as follows:
             {steady_states}
2389
2390
             The fault scenario here is as follows:
             {detailed_fault_scenario}
2391
2392
             # Here is the overview of my Chaos-Engineering experiment to verify the hypothesis:
2393
             {experiment_plan_summary}
2394
2395
             For the first analysis, the following prompt is added
2396
2397
2398
             # The experiment's results are as follows:
2399
             {experiment_result}
2400
             Now, please analyze the results and provide an analysis report rich in insights.
2401
2402
              For the second and subsequent analyses, the following prompt is added
2403
2404
2405
             # The update history for the above K8s manifests is the following:
2406
             {reconfig_history}
2407
             # The experiment's results in the latest K8s manifests are as follows:
2408
             {experiment_result}
2409
             Now, please analyze the results and provide an analysis report rich in insights.
2410
2411
2412
             Example text embedded to experiment_result
2413
2414
             Passed unittests:
2415
             - pre-unittest-example-pod-running-state
2416
             - pre-unittest-example-service-http-response-state
2417
             Failed unittests:
2418
             - fault-unittest-example-pod-running-state
```log
2419
 Exception when calling CoreV1Api->read_namespaced_pod: (404)
2420
 Reason: Not Found\nHTTP response headers: HTTPHeaderDict({'Audit-Id': '8ale6c00-ebd9-43
 ee-9522-6399ce015252', 'Cache-Control': 'no-cache, private', 'Content-Type': '
2421
 application/json', 'X-Kubernetes-Pf-Flowschema-Uid': 'c4624bd9-7fc7-42c6-bcb8-4235110
2422
 a860d', 'X-Kubernetes-Pf-Prioritylevel-Uid': '4706085f-6263-43ae-93f5-b4a61de8b6be', '
 Date': 'Sun, 24 Nov 2024 12:06:18 GMT', 'Content-Length': '190'})
HTTP response body: {\"kind\":\"Status\"...', 'X-Kubernetes-Pf-Flowschema-Uid':
2423
2424
 c4624b9-7fc7-42c6-bcb8-4235110a860d', 'X-Kubernetes-Pf-Prioritylevel-Uid': '4706085f
-6263-43ae-93f5-b4a61de8b6be', 'Date': 'Sun, 24 Nov 2024 12:06:37 GMT', 'Content-Length
2425
 ': '190'})\nHTTP response body: {\"kind\":\"Status\",\"apiVersion\":\"v1\",\"metadata
\":{},\"status\":\"Failure\",\"message\":\"pods \\\"example-pod\\\" not found\",\"
2426
 reason\":\"NotFound\",\"details\":{\"name\":\"example-pod\",\"kind\":\"pods\"},\"code
2427
 \":404}
2428
 Pod was running 0 out of 20 seconds, which is 0.00% of the time.
2429
```

2430 • • • 2431 2432 - fault-unittest-example-service-http-response-state 2433 time=\"2024-11-24T12:06:38Z\" level=warning msg=\"Request Failed\" error=\"Get \\\"http 2434 :///example-service.default.svc.cluster.local:80\\\": dial tcp 10.96.255.84:80: connect: connection refused\"\ntime=\"2024-11-24T12:06:382\" level=warning msg=\" 2435 Request Failed\" error=\"Get \\\"http:///example-service.default.svc.cluster.local 2436 :80///": dial tcp 10.96.255.84:80: connect: connection refused/"/ntime=/"2024-11-24T12 :06:38Z\" level=warning msg=\"Request Failed\" error=\"Get \\\"http:///example-service 2437 .default.svc.cluster.local:8... level=error msg=\"thresholds on metrics ' 2438 http_req_failed' have been crossed 2439 2440

#### B.1.5 IMPROVEMENT

2441 2442

2443 2444

ſ	# 4-0: Agent for reconfiguring K8s manifests
	System:
You are a helpful AI assistant for Chaos Engineering. Given K8s manifests that define a network system, its hypothesis, the overvie Chaos-Engineeering experiment, and the experiment's results, you will reconfi system based on analysis of the experiment's results. Always keep the following rules:	You are a helpful AI assistant for Chaos Engineering. Given K8s manifests that define a network system, its hypothesis, the overview of a
	Chaos-Engineeering experiment, and the experiment's results, you will reconfigure the
<ul> <li>system.</li> <li>NEVER do the same reconfiguration as in the history.</li> <li>Start with simple reconfiguration, and if the hypothesis is still not satisfied gradually try more complex reconfigurations.</li> <li>The output should be formatted as a JSON instance that conforms to the JSON so below.</li> </ul>	system. - NEVER do the same reconfiguration as in the history
	gradually try more complex reconfigurations.
	below.
	<pre>description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \"</pre>
	<pre>string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a vall formatted instance of the scheme. The shirt (\"required.", ("bar\", \"bar\")</pre>
	<pre>weil-tormatted instance of the schema. The object {\"properties\": {\"IOO\": [\"Dar\" \"baz\"]}} is not well-formatted.</pre>
	Here is the output scheme.
	nere is the output schema:
	{\"properties\": {\"thought\": {\"title\": \"Thought\", \"description\": \"Describe
	<pre>\": {\"title\": \"Modified K8S Yamls\", \"description\": \"The list of modified K8s</pre>
	manifests (yamls). If you create a new manifest to modify resources in an existing
	<pre>type\": \"array\", \"items\": {\"\$ref\": \"#/definitions/ModK8sYAML\"}}, \"required`</pre>
	[\"thought\", \"modified_k8s_yamls\"], \"definitions\": {\"ModK8sYAML\": {\"title\"
	Type\", \"description\": \"Modification type. Select from ['replace', 'create', 'del
	']. The 'replace' replaces/overwites the content of an exisiting yaml. The 'create'
	<pre>\"creates a new yami. The defecte defectes an existing yami.(, \ enum( : [\ replace( \"create\", \"delete\"], \"type\": \"string\"}, \"fname\": {\"title\": \"Fname\", \"</pre>
	description\": \"The file name of the modified yaml. If mod_type is 'replace' or '
	file appropriately to avoid overlapping with existing yamls' names. ", "type.": "
<pre>string\"}, \"explanation\": {\"title\": \"Explanation\", \"description\": \"I</pre>	string\"}, \"explanation\": {\"title\": \"Explanation\", \"description\": \"If mod_t
	explain which part you should modify from the original conde and why. If mod_type is
	create', explain whether it is a completely new resource or a replacement resource fo
	reasons for them, just like with 'replace'.\", \"type\": \"string\"}, \"code\": {\"
	title\": \"Code\", \"description\": \"If mod_type is 'delete', this field is not
	unit tests. Write only the content of the code, and for dictionary values, enclose t
	<pre>within a pair of single double quotes (\\\").\", \"type\": \"string\"}}, \"required [\"and type\". \"from \", \"from \", \"required</pre>
<pre>[\"mod_type\", \"fname\", \"explanation\"]}}</pre>	<pre>[\"mod_type\", \"iname\", \"explanation\"]}}</pre>
нuman: # Here is the overview of my system (original version):	<b>numan:</b> # Here is the overview of my system (original version):
	{user_input2}

2483 # Here is the hypothesis for my system:

```
2484
 The hypothesis is "The steady states of the sytem are maintained even when the fault
2485
 scenario occurs (i.e., when the faults are injected)".
2486
 The steady states here are as follows:
 {steady_states}
2487
2488
 The fault scenario here is as follows:
 {detailed_fault_scenario}
2489
2490
 # Here is the overview of my Chaos-Engineering experiment to verify the hypothesis:
 {experiment_plan_summary}
2491
2492
 # The experiment's results of the original system are as follows:
 {experiment_result}
2493
2494
 First, please analyze the results and provide an analysis report rich in insights.
2495
 AI:
2496
 # Here is my analysis report:
 {analysis_report}
2497
2498
 Human:
 Then, please reconfigure the system to avoid the fails (improve resiliency).
2499
2500
2501
 In the improvement loop, the prompts below will be stacked as improvement history
2502
2503
2504
 AI:
            ```json
2505
            {output}
2506
2507
            Human:
            # Here is the K8s menifests of the modified system (version={mod version}):
            {k8s_yamls_mod}
2509
2510
            # The experiment's results of the modified system were as follows:
            {experiment_result_mod}
2511
2512
            Please analyze the results and provide an analysis report rich in insights again.
2513
            AI:
2514
            # Here is my analysis report:
            {analysis_report_mod}
2515
2516
            Human:
            Then, please reconfigure the system to avoid the fails (improve resiliency).
2517
2518
2519
             In the verification loop, the prompts below will be stacked as verification history
2520
             Verification history is kept within a single verification loop and is reset in every
             improvement iteration
2521
2522
2523
            AI:
2524
            ```json
 {output}
2526
 User:
2527
 Your current unittest causes errors when conducted.
2528
 The error message is as follows:
 {error_message}
2529
2530
 Please analyze the reason why the errors occur, then fix the errors.
 Always keep the following rules:
2531
 - Ensure that the implementation supports variable durations again.
2532
 - NEVER repeat the same fixes that have been made in the past.
 - Fix only the parts related to the errors without changing the original content.
2533
 - the same format instruction as in the System role
2534
2535
2536
```

2538	Example text embedded to analysis report
2539	
2540	
2541	The Chaos Engineering experiment aimed to test the system's resilience by simulating a cyber attack through PodChaos and NetworkChaos. The experiment results indicate
2542	failures in maintaining the defined steady states during and after the fault injection
2543	phase. Here is a detailed analysis of the results:
2544	1. **Pre-Validation Phase:**
2545	- Both pre-unittests, `example-pod-running-state` and `example-service-http-response- state`, passed successfully. This indicates that the system was in a healthy state
2546	before the fault injection, with the Pod running 100% of the time and the Service
2547	responding with a 200 OK status for all requests.
2548	2. **Fault Injection Phase:**
2549	- **PodChaos Fault:** The 'fault-unittest-example-pod-running-state' failed because
2550	PodChaos action 'pod-kill', which terminated the Pod. Since the Pod's restart policy
2551	is set to 'Never', it did not restart, leading to 0% uptime during the test. This bightights the critical issue of having a single Pod with no redundancy or automatic
2552	recovery mechanism.
2553	- **NetworkChaos Fault:** The `fault-unittest-example-service-http-response-state`
2554	Service could not route traffic to the Pod, as it was not running. This failure is a
2555	direct consequence of the Pod being unavailable, demonstrating the lack of redundancy
2556	and the impact of a single point of failure.
2557	3. **Post-Validation Phase:**
2558	-state', failed. The Pod was still not found, and the Service continued to refuse
2559	connections. This indicates that the system did not recover to its steady states
2560	absence of a controller to manage Pod lifecycle and redundancy.
2561	LiTurights and Decommondations.
2562	- **Pod Restart Policy:** The Pod's restart policy should be changed from 'Never' to
2563	'Always' or 'OnFailure' to ensure automatic recovery in case of failures.
2564	Pod. This will provide redundancy and ensure that a minimum number of Pods are always
2565	running, improving the system's resilience to failures.
2566	Pod fails by having multiple replicas. This can be achieved by scaling the Deployment
2567	to have more than one replica.
2568	and respond to Pod failures promptly, minimizing downtime.
2569	Du addressing these issues the sustem can improve its resilience and minimum
2570	the system can improve its resilience and maintain its steady states even during fault scenarios.\nHuman: Then, please reconfigure the system
2571	to avoid the fails (improve resiliency).
2572	

## B.1.6 POST-PROCESSING

$\int$	# EX: Agent for summarizing a completed CE cycle
	System:
	Given a summary of a Chaos Engineering cycle, please elaborate the summary. The output should be formatted as a JSON instance that conforms to the JSON schema
	below.
	As an example, for the schema {\"properties\": {\"foo\": {\"title\": \"Foo\", \"
	<pre>description\": \"a list of strings\", \"type\": \"array\", \"items\": {\"type\": \" string\"}}, \"required\": [\"foo\"]}\nthe object {\"foo\": [\"bar\", \"baz\"]} is a well-formatted instance of the schema. The object {\"properties\": {\"foo\": [\"bar"</pre>
<pre>\"baz\"]}} is not well-formatted.</pre>	
	Here is the output schema.
	{\"properties\": {\"summary\": {\"title\": \"Summary\", \"type\": \"string\"}}, \"
requirea\": [\"summary\"]}	requirea\": [\"summary\"]}
	_
	Human: Here is the overview of a Chaos Engineering Cycle:
1	# Here is a Chaos Engineering cycle

2592	
2593	## Step 0. User-input understanding
2594	<pre>### Here is the overview of user inputs: {user input2}</pre>
2595	(doci_inpuc)
2596	## Step 1. Hypothesis definition
2597	The hypothesis is "The steady states of the system are maintained even when the fault
2598	scenario occurs (i.e., when the faults are injected)".
2599	{steady_states}
2600	
2601	The fault scenario here is as follows: {detailed fault scenario}
2602	
2002	## Step 2.1. Chaos-Engineering experiment ### Here is the overview of my Chaos-Engineering experiment to verify the hypothesis:
2003	<pre>{experiment_plan_summary}</pre>
2004	## Step 2.2.3.4 Experiment execution analysis and improvement (reconfiguring the
2605	system to satisfy the hypothesis)
2606	<pre>### Here is the improvement history:     (improvement history)</pre>
2607	{improvement_nistory}
2608	Please elaborate the above summary of a Chaos Engineering Cycle.
2609	
2610	
2611	Example text embedded to <b>improvement_history</b>
2612	
2613	### Experiment result (1st try)
2614	the same content as experiment_result
2615	### Analysis report (1st try)
2616	the same content as analysis_report
2617	### Improvement result (1st try)
2618	1 K8s manifests are modified:
2619	- The Kos manifest "hginx/pod.yami" was repraced.
2620	Replace the Pod manifest with a Deployment manifest to provide redundancy and automatic
2621	number of replicas are always running, which addresses both the restart policy and
2622	redundancy issues.
2623	apiVersion: apps/v1
2624	kind: Deployment
2625	metadata:
2626	labels:
2627	app: example
2628	replicas: 3
2629	selector:
2630	app: example
2631	template:
2632	labels:
2032	app: example
2000	spec: containers:
2034	- name: example-container
2030	<pre>image: nginx:1.1/.1 ports:</pre>
2030	- containerPort: 80
2637	
2638	### Experiment result (2nd try)
2639	Passed unittests:
2640	- pre-unittest-example-service-http-response-state
2641	- fault-unittest-example-pod-running-state
2642	- raure-unittest-example-service-nttp-response-state - post-unittest-example-pod-running-state
2643	- post-unittest-example-service-http-response-state
2644	Failed unittests:
2645	

# 2646 C FULL RESULTS OF CASE STUDIES

In this section, we provide full inputs and outputs for each run studied in the case study. The inputs include instructions, Skaffold configuration files, K8s manifests. The out-puts include VaC scripts, Chaos Mesh workflow manifests, reconfigured K8s manifests, ad-justed VaC scripts in the experiment replanning phase. In the case study, we pick up the fourth run for NGINX and the fifth run for SOCKSHOP. For complete dialogues, see casestudy_complete_dialogues/Nginx/ChaosEater_GPT4o_nginx_4.pdf for NGINX and cases-tudy_complete_dialogues/Sockshop/ChaosEater_GPT4o_sockshop_5.pdf in the Supplementary Ma-terial. The results of other runs can also be found in the same folders. Table 3 shows the statistics of NGINX and SOCKSHOP. 

**Table 3:** Statistics of the systems for the case study. Tokens are counted by the tokenizer cl100k_base.

System	# manifests	# lines	# words	# tokens
NGINX	2	24	373	115
SOCKSHOP	29	869	17696	4605

C.1 NGINX

2670 C.1.1 INPUTS

Instructions

The Chaos-Engineering experiment must be completed within 1 minute.

#### skaffold.yaml

1 apiVersion: skaffold/v3
2 kind: Config
3 metadata:
4 name: nginx-app
5 manifests:
6 rawYaml:
7 - pod.yaml
8 - service.yaml

#### pod.yaml

1 apiVersion: v1 kind: Pod metadata: name: example-pod labels: app: example spec: restartPolicy: Never containers: - name: example-container image: nginx:1.17.1 ports: - containerPort: 80

#### service.yaml

apiVersion: v1

kind: Service

selector: app: example

ports:

name: example-service

- protocol: TCP port: 80

targetPort: 80

metadata:

spec:

2701	
2702	
2703	
2704	3
2705	
2706	6
2707	
2708	9
2709	10
2710	
2711	C
2712	~
2713	Ċ.
2714	$ \subset $
2715	
2716	_
	1

```
27
27
 1.2 OUTPUTS
27
27
 VaC script for the steady state #1: The Pod should be running at least 90% of the time during
27
 the check period
27
2717
 import os
2718
 2 import time
2719
 3 import argparse
 4 from kubernetes import client, config
2720
 5 from unittest_base import K8sAPIBase
2721
 6
 7
 class TestPodRunningState(K8sAPIBase):
2722
 8
 def __init__(self):
2723
 9
 super().__init__()
 10
2724
 def check_pod_status(self, namespace, pod_name):
 11
2725
 12
 try:
 13
 pod = self.vl.read_namespaced_pod(name=pod_name, namespace=namespace)
2726
 return pod.status.phase == 'Running'
 14
2727
 15
 except client.exceptions.ApiException as e:
 print (f"Exception when calling CoreV1Api->read_namespaced_pod: {e}")
 16
2728
 17
 return False
2729
 18
 19
 def test pod running state(self, duration):
2730
 namespace = 'default'
 20
 pod_name = 'example-pod'
2731
 21
 22
 running_count = 0
2732
 23
2733
 24
 # Check the pod status every second for the specified duration
 25
 for _ in range(duration):
2734
 26
 if self.check_pod_status(namespace, pod_name):
2735
 27
 running_count += 1
 28
 time.sleep(1)
2736
 29
2737
 # Calculate the running percentage
 30
 running_percentage = (running_count / duration) * 100
 31
2738
 32
2739
 33
 # Assert that the running percentage is at least 90%
 assert running_percentage >= 90, f"Pod '{pod_name}' running percentage is below
threshold: {running_percentage}%"
 34
2740
2741
 35
 36
 print(f"Pod '{pod_name}' running status checked {duration} times. Running
2742
 percentage: {running_percentage}%.")
 37
2743
 38
2744
 39 def main():
 parser = argparse.ArgumentParser(description='Test if a pod is running at least 90%
2745
 40
 of the time.')
2746
 41
 parser.add_argument('--duration', type=int, default=5, help='Duration to check the
2747
 pod status in seconds.')
 42
 args = parser.parse_args()
2748
 43
2749
 44
 test = TestPodRunningState()
 45
 test.test_pod_running_state(args.duration)
2750
 46
 47
2751
 48 if _____ == '___main__':
2752
 49
 main()
2753
```

```
sponse status of 200
 1 import http from 'k6/http';
 2 import { check } from 'k6';
 3
 4 export const options = {
 5
 vus: 1,
 duration: '5s',
 6
 thresholds: {
 7
 // Ensure that the service availability is at least 99.9%
 8
 // Indic the Service availability is at least 99.9%
'http_req_failed': ['rate<=0.001'], // 0.1% failure rate corresponds to 99.9%
availability</pre>
 9
10
 },
11 };
12
13 export \textbf{default} function () {
 const res = http.get('http://example-service.default.svc.cluster.local:80');
check(res, {
14
15
16
 'status is 200': (r) => r.status === 200,
 });
17
18 }
```

VaC script for the steady state #2: Service availability should be at least 99.9% with a re-

#### Chaos Mesh workflow manifest

2774	
2775	1 apiVersion: chaos-mesh.org/vlalphal
2776	2 kind: Workflow 3 metadata:
2777	4 name: chaos-experiment-20241124-132854
2778	5 spec: 6 entry: the-entry
2779	7 templates:
2780	8
2781	10 #
2782	11 - name: the-entry
2783	13 deadline: 30m51s
0704	14 children:
2704	15 - pre-validation-phase
2785	10 - fault-injection-phase
2786	18
2787	19 #
2788	20  # Entry point of pre-validation-phase 21  #
2789	22 - name: pre-validation-phase
2790	23 templateType: Serial
2701	24 deadline: lomios 25 children:
2731	26 - pre-validation-overlapped-workflows
2792	27 28 - name: pro-validation-suspend-workflow
2793	29 templateType: Serial
2794	30 deadline: 5ml0s
2795	31 children:
2796	<ul> <li>32 - pre-validation-suspend</li> <li>33 - pre-unittest-example-service-availability</li> </ul>
2797	34
2798	35 - name: pre-validation-suspend 26 - harmlateTurget Guerrand
2799	37 deadline: 5s
2200	38
2000	39 - name: pre-validation-overlapped-workflows 40 templateType: Parallel
2801	41 deadline: 5ml0s
2802	42 children:
2803	43 - pre-unittest-example-pod-running 44 - pre-validation-suspend-workflow
2804	45
2805	46 # Definitions of children of pre-validation-phase
2806	<ul> <li>4/ - name: pre-unittest-example-pod-running</li> <li>48 templateType: Task</li> </ul>
2807	49 deadline: 5m5s

2808	l .	
2809	50	task:
2810	51	container:
2811	53	image: chaos-eater/k8sapi:1.0
2812	54	imagePullPolicy: IfNotPresent
2813	56	args: ["python /chaos-eater/sandbox/cycle 20241124 132128/unittest example-
2013		pod-running_mod0.pyduration 5"]
2814	57	volumeMounts:
2815	59	mountPath: /chaos-eater
2816	60	volumes:
2817	61	- name: pvc-volume
2818	63	claimName: pvc
2819	64	
2820	66	- name: pre-unittest-example-service-availability templateType: Task
2821	67	deadline: 5m5s
2822	68	task:
2823	70	name: pre-unittest-example-service-availability-container
2020	71	<pre>image: grafana/k6:latest</pre>
2024	72	command: ["k6", "run", "duration", "5s", "quiet", "/chaos-eater/sandbox/ cvcle 20241124 132128/unittest example-service-availability mod0 is"]
2020	73	volumeMounts:
2826	74	- name: pvc-volume
2827	76	volumes:
2828	77	- name: pvc-volume
2829	78	persistentVolumeClaim:
2830	80	
2831	81	#
2832	82	<pre># Entry point of fault-injection-phase #</pre>
2833	84	- name: fault-injection-phase
2834	85	templateType: Serial
2835	87	children:
2836	88	- fault-injection-overlapped-workflows
2837	89	- name: fault-injection-parallel-workflow
2037	91	templateType: Parallel
2030	92	deadline: 5m10s
2839	95	- fault-unittest-example-pod-running
2840	95	- fault-podchaos
2841	96	- name: fault-injection-suspend-workflow
2842	98	templateType: Serial
2843	99	deadline: 5m30s children:
2844	101	- fault-injection-suspend
2845	102	- fault-injection-parallel-workflows
2846	103	- name: fault-injection-suspend
2847	105	templateType: Suspend
2848	106	deadline: 10s
2849	107	- name: fault-injection-parallel-workflows
2850	109	templateType: Parallel
2050	110	deadline: 5m2Us children:
2031	112	- fault-unittest-example-service-availability
2002	113	- fault-networkchaos
2853	115	- name: fault-injection-overlapped-workflows
2854	116	templateType: Parallel
2855	111/	aeaaiine: 5m3Us children:
2856	119	- fault-injection-parallel-workflow
2857	120	- fault-injection-suspend-workflow
2858	122	# Definitions of children of pre-validation-phase
2859	123	# unit tests
2860	124	<pre>- name: rautt-unittest-example-pod-running templateType: Task</pre>
2861		

2862		
2863	126 deadline: 5m10s	
2864	127 task: 128 container:	
2865	129 name: fault-unittest-example-pod-running-container	
2866	130 image: chaos-eater/k8sapi:1.0	
2000	131 imagePullPolicy: liNotPresent	
2007	133 args: ["python /chaos-eater/sandbox/cycle_20241124_	_132128/unittest_example-
2868	pod-running_mod0.pyduration 10"]	
2869	134 volumemounts: 135 – name: pvc-volume	
2870	136 mountPath: /chaos-eater	
2871	137 volumes:	
2872	139 persistentVolumeClaim:	
2873	140 claimName: pvc	
2874	141 142 – name: fault-unittest-example-service-availability	
2875	143 templateType: Task	
2876	144 deadline: 5m20s	
2077	145 task: 146 container:	
2077	147 name: fault-unittest-example-service-availability-c	container
2878	148 image: grafana/k6:latest	t" "(chang gater (gardboy (
2879	cycle_20241124_132128/unittest_example-service-avai	lability_mod0.js"]
2880	150 volumeMounts:	
2881	151 - name: pvc-volume 152 mountPath: /chaos-eater	
2882	153 volumes:	
2883	154 - name: pvc-volume	
2884	155 persistentVolumeClaim: 156 claimName: pvc	
2885	157	
2886	158 # fault_injections	
2887	160 templateType: PodChaos	
2007	161 deadline: 10s	
2000	162 podChaos:	
2889	164 mode: one	
2890	165 selector:	
2891	160 labelSelectors: 167 app: example	
2892	168 namespaces:	
2893	169 - default	
2894	171	
2895	172 - name: fault-networkchaos	
2896	173 templateType: NetworkChaos	
2897	175 networkChaos:	
2898	176 action: delay	
2800	177 delay: 178 correlation: '50'	
2000	179 jitter: 10ms	
2300	180 latency: 100ms	
2901	182 direction: to	
2902	183 mode: all	
2903	184 selector: 185 labelSelectors:	
2904	186 app: example	
2905	187 namespaces:	
2906	189 target:	
2907	190 mode: all	
2908	191 selector: 192 labelSelectors:	
2909	193 app: example	
2910	194 namespaces:	
2011	195 - derault 196	
2311	197	
2912	198 #	
2913	200 #	
2914	201 - name: post-validation-phase	
2915		

202	templateType: Serial
203	deadline: 10mlls
204	children:
205	- post-validation-overlapped-workilows
200	- name: post-validation-suspend-workflow
208	templateType: Serial
09	deadline: 5m8s
0	children:
1	- post-validation-suspend
12	- post-unittest-example-pod-running
4	- name: post-validation-suspend
15	templateType: Suspend
6	deadline: 2s
7	
5	- name: post-validation-suspend-workflow2
9 0	deadline Smils
1	children:
22	- post-validation-suspend2
23	- post-unittest-example-service-availability
24	
5	- name: post-validation-suspend2 templateType: Suspend
27	deadline: 6s
28	
9	- name: post-validation-overlapped-workflows
30	templateType: Parallel
31	deadline: 5mlls
2	children:
4	<pre>- post-validation-suspend-workflow2</pre>
5	poor variateron babpena kornitoki
6	# Definitions of children of pre-validation-phase
37	- name: post-unittest-example-pod-running
38	templateType: Task
9	aeaaiine: SMbS task.
1	container:
2	name: post-unittest-example-pod-running-container
3	image: chaos-eater/k8sapi:1.0
4	imagePullPolicy: IfNotPresent
45 46	command: ["/bin/bash", "-c"] args: ["nuthon /chaos-pater/gandhoy/guale 20241124 122129/unittest guarrale
0	args. [ python /chaos-eater/sandbox/cycre_20241124_152128/unittest_eXample- pod-rupping mod0 pyduration 6"]
17	volumeMounts:
48	- name: pvc-volume
19	mountPath: /chaos-eater
50	volumes:
52	- name: pvc-volume
.52 253	claimName: pvc
54	Statimane. Pro
55	- name: post-unittest-example-service-availability
56	templateType: Task
57	deadline: 5m5s
58	task:
59 60	CONTAINER:
261	image: grafana/k6:latest
62	command: ["k6", "run", "duration", "5s", "quiet", "/chaos-eater/sandbox/
	cycle_20241124_132128/unittest_example-service-availability_mod0.js"]
63	volumeMounts:
64	- name: pvc-volume
.65	mountPath: /chaos-eater
266	volumes:
268	<pre>procession - name: procession - name: processi</pre>
69	claimName: pvc
	•

Reconfigured pod.yaml

1 apiVersion: apps/v1
2 kind: Deployment

3 4 5 6 7 8 9	metadata.
4 5 6 7 8 9	metadata.
5 6 7 8 9	name: example-deployment
7 8 9	app: example
8 9	specification
9	replicas: 3
10	selector:
10	matchiadeis:
12	template:
13	metadata:
14	labels:
15	app: example
17	containers:
18	- name: example-container
19	image: nginx:1.17.1
20	ports: - containerPort: 80
21	
_	
	Adjusted VaC script for the steady state #1: The Pod should be running at least 90% of the
	time during the check period
1	import of
2	import dime
3	import argparse
4	from kubernetes import client, config
5	<b>from</b> unittest_base <b>import</b> K8sAPIBase
0	class TestDenloymentRunningState (K&sAPTRase) .
8	def init (self):
9	<pre>super()init()</pre>
10	
11	<pre>def check_deployment_pods_status(self, namespace, deployment_name):     territe</pre>
12	<b>try:</b> pods = self vi list namespaced pod(namespace=namespace, label selector=f'
	app={deployment_name/).items
14	<pre>running_pods = [pod for pod in pods if pod.status.phase == 'Running']</pre>
15	<pre>return len(running_pods) &gt; 0</pre>
10	except client.exceptions.ApiException as e:
18	return False
19	
20	<pre>def test_deployment_running_state(self, duration):</pre>
21	namespace = 'default'
23	$\frac{deployment_iname}{deployment_iname} = \frac{example}{deployment_iname}$
24	
25	# Check the deployment pods status every second for the specified duration
26	<pre>for _ in range(duration):     if calls about doubles, in the interval of the context of the</pre>
27	<pre>ir seif.cneck_depioyment_pods_status(namespace, deployment_name):     running_count += 1</pre>
27 28	time.sleep(1)
27 28 29	-
27 28 29 30	
27 28 29 30 31	# Calculate the running percentage
27 28 29 30 31 32 33	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100</pre>
27 28 29 30 31 32 33 34	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90%</pre>
27 28 29 30 31 32 33 34 35	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running</pre>
27 28 29 30 31 32 33 34 35	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%"</pre>
27 28 29 30 31 32 33 34 35 36 37	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '(deployment name)' running status sheeked (duration) times)</pre>
27 28 29 30 31 32 33 34 35 36 37	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '(deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running percentage}%.")</pre>
27 28 29 30 31 32 33 34 35 36 37 38	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '(deployment_name)' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.")</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.")</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main():</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main(): parser = argparse.ArgumentParser(description='Test if a deployment has at least one running percentage) for the time ')</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main():     parser = argparse.ArgumentParser(description='Test if a deployment has at least one     pod running at least 90% of the time.')     parser.add argument('duration'. type=int. default=5. help='Duration to check the </pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main(): parser = argparse.ArgumentParser(description='Test if a deployment has at least one pod running at least 90% of the time.') parser.add_argument('duration', type=int, default=5, help='Duration to check the deployment pods status in seconds.')</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main(): parser = argparse.ArgumentParser(description='Test if a deployment has at least one pod running at least 90% of the time.') parser.add_argument('duration', type=int, default=5, help='Duration to check the deployment pods status in seconds.') args = parser.parse_args()</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main(): parser = argparse.ArgumentParser(description='Test if a deployment has at least one pod running at least 90% of the time.') parser.add_argument('duration', type=int, default=5, help='Duration to check the deployment pods status in seconds.') args = parser.parse_args()</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main(): parser = argparse.ArgumentParser(description='Test if a deployment has at least one pod running at least 90% of the time.') parser.add_argument('duration', type=int, default=5, help='Duration to check the deployment pods status in seconds.') args = parser.parse_args() test = TestDeploymentRunningState() test test deployment running state(args duration)</pre>
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47	<pre># Calculate the running percentage running_percentage = (running_count / duration) * 100 # Assert that the running percentage is at least 90% assert running_percentage &gt;= 90, f"Deployment '{deployment_name}' running percentage is below threshold: {running_percentage}%" print(f"Deployment '{deployment_name}' running status checked {duration} times. Running percentage: {running_percentage}%.") def main(): parser = argparse.ArgumentParser(description='Test if a deployment has at least one pod running at least 90% of the time.') parser.add_argument('duration', type=int, default=5, help='Duration to check the deployment pods status in seconds.') args = parser.parse_args() test = TestDeploymentRunningState() test.test_deployment_running_state(args.duration)</pre>

__name__ == '__main__':

3033 3034

3035 3036

3037

3038

3039

3040

3041

3042

3043

3044

3045

3046

C.2 SOCKSHOP

main()

48

50

49 if

C.2.1 INPUTS

Instructions - The Chaos-Engineering experiment must be completed within 1 minute. - When using k6 in steady-state definition, always select a request URL from the following options (other requests are invalid): 1. http://front-end.sock-shop.svc.cluster.local/ 2. http://front-end.sock-shop.svc.cluster.local/catalogue?size=10 3. http://front-end.sock-shop.svc.cluster.local/detail.html?id=<ID> Replace <ID> with an available ID: [03fef6ac-1896-4ce8-bd69-b798f85c6e0b, 3395a43e-2 d88-40de-b95f-e00e1502085b, 510a0d7e-8e83-4193-b483-e27e09ddc34d, 808a2de1-1aaa-4c25 -a9b9-6612e8f29a38, 819e1fbf-8b7e-4f6d-811f-693534916a8b, 837ab141-399e-4c1f-9abcbace40296bac, a0a4f044-b040-410d-8ead-4de0446aec7e, d3588630-ad8e-49df-bbd7-3167 f7efb246, zzz4f044-b040-410d-8ead-4de0446aec7e] 4. http://front-end.sock-shop.svc.cluster.local/category/ 5. http://front-end.sock-shop.svc.cluster.local/category?tags=<TAG> Replace <TAG> with an available tag: [magic, action, blue, brown, black, sport, formal, red, green, skin, geek] 6. http://front-end.sock-shop.svc.cluster.local/basket.html

#### skaffold.yaml

<pre>apiVersion: skaffold/v3 kind: Config metadata:     name: sock-shop-app manifests:     rawYaml:         manifests/00-sock-shop-ns.yaml         manifests/01-carts-dep.yaml         manifests/02-carts-wc.yaml         manifests/04-carts-db-dep.yaml         manifests/04-carts-db-dep.yaml         manifests/06-catalogue-dep.yaml         manifests/07-catalogue-db-svc.yaml         manifests/08-catalogue-db-svc.yaml         manifests/08-catalogue-db-svc.yaml         manifests/08-catalogue-db-svc.yaml         manifests/10-front-end-dep.yaml         manifests/10-front-end-dep.yaml         manifests/12-orders-db-dep.yaml         manifests/12-orders-db-dep.yaml         manifests/12-orders-db-dep.yaml         manifests/14-orders-db-dep.yaml         manifests/15-payment-dep.yaml         manifests/15-payment-dep.yaml         manifests/19-rabbitmq-dep.yaml         manifests/19-rabbitmq-dep.yaml         manifests/19-rabbitmq-dep.yaml         manifests/20-rabbitmq-dep.yaml         manifests/21-session-db-dep.yaml         manifests/22-session-db-dep.yaml         manifests/22-session-db-dep.yaml         manifests/22-session-db-dep.yaml         manifests/22-session-db-dep.yaml         manifests/22-session-db-dep.yaml         manifests/22-session-db-dep.yaml         manifests/22-session-db-dep.yaml         manifests/23-shipping-dep.yaml         manifests/24-shipping-dep.yaml         manifests/24-shipping-dep.yaml</pre>			
<pre>apiVersion: skaffold/v3 kind: Config metadata:     name: sock-shop-app manifests:     rawYaml:         - manifests/00-sock-shop-ns.yaml         manifests/01-carts-dep.yaml         manifests/02-carts-svc.yaml         manifests/04-carts-db-svc.yaml         manifests/04-carts-db-svc.yaml         manifests/06-catalogue-svc.yaml         manifests/06-catalogue-svc.yaml         manifests/07-catalogue-db-dep.yaml         manifests/08-catalogue-db-dep.yaml         manifests/08-catalogue-db-dep.yaml         manifests/09-front-end-dep.yaml         manifests/10-front-end-svc.yaml         manifests/11-orders-dep.yaml         manifests/12-orders-svc.yaml         manifests/12-orders-db-svc.yaml         manifests/14-orders-db-svc.yaml         manifests/15-payment-dep.yaml         manifests/18-queue-master-dep.yaml         manifests/19-rabbitmq-dep.yaml         manifests/21-session-db-dep.yaml         manifests/22-session-db-svc.yaml         manifests/23-shipping-dep.yaml         manifests/24-shipping-svc.yaml         manifests/</pre>			
<pre>kind: Config metadata: name: sock-shop-app manifests: rawYaml: manifests/00-sock-shop-ns.yaml manifests/01-carts-dep.yaml manifests/02-carts-db-dep.yaml manifests/02-carts-db-dep.yaml manifests/04-carts-db-svc.yaml manifests/05-catalogue-dp-yaml manifests/06-catalogue-db-dep.yaml manifests/06-catalogue-db-dep.yaml manifests/08-catalogue-db-svc.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/12-orders-db-dp.yaml manifests/14-orders-db-svc.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/18-queue-master-dep.yaml manifests/18-queue-master-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/26-user-db-dep.yaml manifests/26-user-db-dep.yaml manifests/26-user-db-dep.yaml manifests/26-user-db-dep.yaml manifests/28-user-db-dep.yaml manifests/28-user-db-dep.yaml manifests/28-user-db-dep.yaml manifests/28-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	1	apiVersion: skaffold/v3	
<pre>metadata: name: sock-shop-app manifests: rawYaml: manifests/01-carts-dep.yaml manifests/02-carts-svc.yaml manifests/02-carts-db-dep.yaml manifests/03-carts-db-svc.yaml manifests/04-carts-db-svc.yaml manifests/06-catalogue-db-svc.yaml manifests/06-catalogue-db-gyml manifests/08-catalogue-db-svc.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-dp.yaml manifests/12-orders-dp.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-dp.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/22-session-db-dep.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/25-user-dp.yaml manifests/26-user-svc.yaml manifests/26-user-dp.yaml manifests/26-user-dp.yaml manifests/26-user-dp.yaml manifests/26-user-dp.yaml manifests/26-user-dp.yaml manifests/26-user-dp.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	2	kind: Config	
<pre>name: sock-shop-app manifests: rawYanl: manifests/00-sock-shop-ns.yaml manifests/01-carts-dep.yaml manifests/02-carts-svc.yaml manifests/03-carts-db-dep.yaml manifests/04-carts-db-dep.yaml manifests/05-catalogue-db-dep.yaml manifests/06-catalogue-db-dep.yaml manifests/07-catalogue-db-dep.yaml manifests/07-catalogue-db-dep.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/14-orders-db-svc.yaml manifests/16-payment-dep.yaml manifests/18-payment-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/26-user-db-svc.yaml manifests/26-user-db-svc.yaml manifests/26-user-db-svc.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	3	metadata:	
<pre>manifests: rawYaml: manifests/01-carts-dep.yaml manifests/02-carts-svc.yaml manifests/02-carts-db-dep.yaml manifests/04-carts-db-svc.yaml manifests/04-carts-db-svc.yaml manifests/06-catalogue-dbp.yaml manifests/06-catalogue-db-dep.yaml manifests/07-catalogue-db-svc.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/14-orders-dep.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/24-shipping-dvc.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-yaml manifests/25-user-db-yaml manifests/26-user-svc.yaml manifests/26-user-svc.yaml manifests/26-user-db-yaml manifests/28-user-db-yaml manifests/28-user-db-svc.yaml</pre>	4	name: sock-shop-app	
<pre>rawYaml: manifests/00-sock-shop-ns.yaml manifests/01-carts-dep.yaml manifests/02-carts-svc.yaml manifests/03-carts-db-dep.yaml manifests/04-carts-db-svc.yaml manifests/05-catalogue-dep.yaml manifests/06-catalogue-db-yaml manifests/07-catalogue-db-dep.yaml manifests/09-front-end-dep.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/14-orders-db-svc.yaml manifests/16-payment-exc.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-yaml manifests/25-user-db-yaml manifests/26-user-svc.yaml manifests/26-user-svc.yaml manifests/26-user-db-yaml manifests/28-user-db-svc.yaml</pre>	5	manifests:	
<pre>- manifests/00-sock-shop-ns.yaml manifests/01-carts-dep.yaml manifests/02-carts-db-dep.yaml manifests/03-carts-db-dep.yaml manifests/04-carts-db-svc.yaml manifests/05-catalogue-dbp.yaml manifests/05-catalogue-db-dep.yaml manifests/08-catalogue-db-svc.yaml manifests/10-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/12-orders-db-dep.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/16-payment-end-vyml manifests/16-payment-svc.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	6	rawYaml:	
<pre>- manifests/01-carts-dep.yaml manifests/02-carts-svc.yaml manifests/03-carts-db-dep.yaml manifests/04-carts-db-svc.yaml manifests/06-catalogue-db-yaml manifests/06-catalogue-db-dep.yaml manifests/08-catalogue-db-svc.yaml manifests/10-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/14-orders-db-svc.yaml manifests/14-orders-db-svc.yaml manifests/16-payment-dep.yaml manifests/17-queue-master-dep.yaml manifests/18-queue-master-svc.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/23-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-svc.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	7	<ul> <li>manifests/00-sock-shop-ns.yaml</li> </ul>	
<pre>- manifests/02-carts-svc.yaml manifests/03-carts-db-dep.yaml manifests/04-carts-db-svc.yaml manifests/05-catalogue-dep.yaml manifests/06-catalogue-db-yaml manifests/08-catalogue-db-svc.yaml manifests/08-catalogue-db-svc.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/14-orders-db-dep.yaml manifests/15-payment-dep.yaml manifests/15-payment-svc.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-svc.yaml manifests/26-user-svc.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	8	<ul> <li>manifests/01-carts-dep.yaml</li> </ul>	
<pre>- manifests/03-carts-db-dep.yaml manifests/04-carts-db-svc.yaml manifests/05-catalogue-dep.yaml manifests/06-catalogue-svc.yaml manifests/08-catalogue-db-dep.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-dep.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/26-user-dep.yaml manifests/27-user-db-dep.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	9	- manifests/02-carts-svc.yaml	
<pre>- manifests/04-carts-db-svc.yaml manifests/05-catalogue-dep.yaml manifests/06-catalogue-db-yaml manifests/07-catalogue-db-dep.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-dep.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-db-svc.yaml manifests/25-user-dep.yaml manifests/25-user-db-yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	10	<ul> <li>manifests/03-carts-db-dep.yaml</li> </ul>	
<pre>manifests/05-catalogue-dep.yaml manifests/06-catalogue-svc.yaml manifests/07-catalogue-db-dep.yaml manifests/08-catalogue-db-svc.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/14-orders-db-svc.yaml manifests/14-orders-db-svc.yaml manifests/16-payment-dep.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-svc.yaml manifests/25-user-db-svc.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	11	<ul> <li>manifests/04-carts-db-svc.yaml</li> </ul>	
<pre>manifests/06-catalogue-svc.yaml manifests/07-catalogue-db-dep.yaml manifests/08-catalogue-db-svc.yaml manifests/10-front-end-dep.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/13-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-dep.yaml manifests/17-queue-master-dep.yaml manifests/18-queue-master-svc.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	12	<ul> <li>manifests/05-catalogue-dep.yaml</li> </ul>	
<pre>- manifests/07-catalogue-db-dep.yaml manifests/08-catalogue-db-svc.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/12-orders-dep.yaml manifests/12-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/18-queue-master-vc.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/26-user-db-svc.yaml manifests/27-user-db-dep.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	13	<ul> <li>manifests/06-catalogue-svc.yaml</li> </ul>	
<pre>- manifests/08-catalogue-db-svc.yaml manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-svc.yaml manifests/14-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/16-queue-master-dep.yaml manifests/18-queue-master-svc.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-dep.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/27-user-db-dep.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	14	<ul> <li>manifests/07-catalogue-db-dep.yaml</li> </ul>	
<pre>- manifests/09-front-end-dep.yaml manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	15	- manifests/08-catalogue-db-svc.yaml	
<pre>- manifests/10-front-end-svc.yaml manifests/11-orders-dep.yaml manifests/12-orders-db-dep.yaml manifests/14-orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-dep.yaml manifests/25-user-dep.yaml manifests/25-user-dep.yaml manifests/26-user-dep.yaml manifests/27-user-db-dep.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	16	- manifests/09-front-end-dep.yaml	
<pre>- manifests/l1-orders-dep.yaml manifests/l2-orders-svc.yaml manifests/l3-orders-db-dep.yaml manifests/l4-orders-db-svc.yaml manifests/l6-payment-dep.yaml manifests/l7-queue-master-dep.yaml manifests/l8-queue-master-svc.yaml manifests/l9-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	17	- manifests/10-front-end-svc.yaml	
<pre>- manifests/12-orders-svc.yaml manifests/13-orders-db-qp.yaml manifests/14-orders-db-svc.yaml manifests/16-payment-dep.yaml manifests/16-payment-svc.yaml manifests/17-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/22-session-db-svc.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/27-user-db-svc.yaml manifests/28-user-db-svc.yaml</pre>	18	- manifests/ll-orders-dep.yaml	
<pre>- manifests/13-orders-db-dep.yaml manifests/14-orders-db-avc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/16-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/20-rabbitmq-svc.yaml manifests/22-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/26-user-db-yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	19	- manifests/12-orders-svc.yaml	
<pre>- Manifests/14-Orders-db-svc.yaml manifests/15-payment-dep.yaml manifests/16-payment-svc.yaml manifests/18-queue-master-dep.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-dep.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	20	- manifests/13-orders-db-dep.yam1	
<pre>manifests/16-payment-dep.yam1 manifests/16-payment-svc.yam1 manifests/18-queue-master-dep.yam1 manifests/19-rabbitmq-dep.yam1 manifests/20-rabbitmq-dep.yam1 manifests/21-session-db-dep.yam1 manifests/22-session-db-svc.yam1 manifests/23-shipping-dep.yam1 manifests/24-shipping-dep.yam1 manifests/25-user-dep.yam1 manifests/26-user-svc.yam1 manifests/27-user-db-dep.yam1 manifests/28-user-db-svc.yam1</pre>	$ ^{21}_{22}$	- manifests/14-orders-db-svc.yami	
<pre>manifests/10-payment-soc.yaml manifests/17-queue-master-dep.yaml manifests/18-queue-master-soc.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-soc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-soc.yaml manifests/25-user-dep.yaml manifests/26-user-soc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-soc.yaml</pre>	22	- manifests/15-payment-dep.yami	
<pre>manifests/18-queue-master-svc.yaml manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-dep.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	$\frac{23}{24}$	- manifests/10-payment-svc.yami	
<pre>manifests/19-rabbitmq-dep.yaml manifests/20-rabbitmq-svc.yaml manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	25	- manifests/1/-queue-master-uep.yami	
<pre>- manifests/20-rabbitmq-svc.yaml - manifests/21-session-db-dep.yaml - manifests/22-session-db-svc.yaml - manifests/23-shipping-dep.yaml - manifests/24-shipping-svc.yaml - manifests/25-user-dep.yaml - manifests/26-user-svc.yaml - manifests/27-user-db-dep.yaml - manifests/28-user-db-svc.yaml</pre>	26	- manifests/19-rabbitmg-dep.vaml	
<pre>- manifests/21-session-db-dep.yaml manifests/22-session-db-svc.yaml manifests/23-shipping-dep.yaml manifests/24-shipping-svc.yaml manifests/25-user-dep.yaml manifests/26-user-svc.yaml manifests/27-user-db-dep.yaml manifests/28-user-db-svc.yaml</pre>	27	- manifests/20-rabbitmg-svc.yaml	
<pre>- manifests/22-session-db-svc.yaml - manifests/23-shipping-dep.yaml - manifests/24-shipping-svc.yaml - manifests/25-user-dep.yaml - manifests/26-user-svc.yaml - manifests/27-user-db-dep.yaml - manifests/28-user-db-svc.yaml</pre>	28	- manifests/21-session-db-dep.vaml	
<ul> <li>manifests/23-shipping-dep.yaml</li> <li>manifests/24-shipping-svc.yaml</li> <li>manifests/25-user-dep.yaml</li> <li>manifests/26-user-svc.yaml</li> <li>manifests/27-user-db-dep.yaml</li> <li>manifests/28-user-db-svc.yaml</li> </ul>	29	- manifests/22-session-db-svc.vaml	
<ul> <li>manifests/24-shipping-svc.yaml</li> <li>manifests/25-user-dep.yaml</li> <li>manifests/26-user-svc.yaml</li> <li>manifests/27-user-db-dep.yaml</li> <li>manifests/28-user-db-svc.yaml</li> </ul>	30	- manifests/23-shipping-dep.yaml	
<ul> <li>manifests/25-user-dep.yaml</li> <li>manifests/26-user-svc.yaml</li> <li>manifests/27-user-db-dep.yaml</li> <li>manifests/28-user-db-svc.yaml</li> </ul>	31	- manifests/24-shipping-svc.yaml	
- manifests/26-user-svc.yaml - manifests/27-user-db-dep.yaml - manifests/28-user-db-svc.yaml	32	- manifests/25-user-dep.yaml	
- manifests/27-user-db-dep.yaml - manifests/28-user-db-svc.yaml	33	- manifests/26-user-svc.yaml	
- manifests/28-user-db-svc.yaml	34	<ul> <li>manifests/27-user-db-dep.yaml</li> </ul>	
	35	<ul> <li>manifests/28-user-db-svc.yaml</li> </ul>	
	_		

#### manifests/00-sock-shop-ns.yaml

1 apiVersion: v1
2 kind: Namespace
3 metadata:

4 name: sock-shop

### manifests/01-carts-dep.yaml

1	apiVersion: apps/v1
2	kind: Deployment
	metadata:
5	labels:
6	name: carts
7	namespace: sock-shop
8	spec: replicas: 2
10	selector:
11	matchLabels:
12	name: carts
13	template:
14	metadata:
15	labels:
16	name: carts
1/	spec:
10	
20	image: weaveworksdemos/carts:0.4.8
21	env:
22	- name: JAVA_OPTS
23	value: -Xms64m -Xmx128m -XX:+UseG1GC -Djava.security.egd=file:/dev/urandom -
24	Dspring.zipkin.enabled=false
24	resources:
25 26	cout: 300m
20	menory 500Mi
28	requests:
29	cpu: 100m
30	memory: 200Mi
31	ports:
32	- containerPort: 80
33	securityContext:
34	runAsNonRoot: true
35	runasuser: 10001
37	dron.
38	
39	add:
40	- NET_BIND_SERVICE
41	readOnlyRootFilesystem: true
42	volumeMounts:
43	- mountPath: /tmp
44	name: tmp-volume
45	volumes:
46 47	- name: tmp-volume
4/ 18	emptypir:
+0 40	neutor, Memory
50	beta.kubernetes.io/os: linux
[ ³	
L	

### manifests/02-carts-svc.yaml

1 apiVersion: v1 2 kind: Service 3 metadata: name: carts prometheus.io/scrape: 'true' labels: name: carts 

# the port that this service should serve on

namespace: sock-shop

targetPort: 80

0

11

12

13

14

10 spec:

ports:

- port: 80

3137	15	selector:
3138	16	name: carts
3139	L	
3140	_	
3141		manifests/03-carts-db-dep.yaml
3142		
3143	1	apiVersion: apps/vl
3144	2	kind: Deployment
0145		metadata:
3145	5	labels:
3146	6	name: carts-db
3147	7	namespace: sock-shop
3148	8	spec:
21/0	10	replicas: 2
3149	11	matchLabels:
3150	12	name: carts-db
3151	13	template:
3152	14	metadata:
0150	15	labels:
3153	17	spec:
3154	18	containers:
3155	19	- name: carts-db
3156	20	image: mongo
0457	$\begin{vmatrix} 21 \\ 22 \end{vmatrix}$	ports:
3157	23	containerPort: 27017
3158	24	securityContext:
3159	25	capabilities:
3160	26	drop:
0100	21	- all
3101	29	- CHOWN
3162	30	- SETGID
3163	31	- SETUID
3164	32	readOnlyRootFilesystem: true
3165	34	- mountPath: /tmp
0100	35	name: tmp-volume
3166	36	volumes:
3167	37	- name: tmp-volume
3168	39	medium: Memory
3169	40	nodeSelector:
3170	41	beta.kubernetes.io/os: linux
3171	l	
0470	_	
3172		

3173 3174

3175

3176

3177

3178

3179

3180

3181

3182

3183

3184 3185

#### manifests/04-carts-db-svc.yaml

1 apiVersion: v1 2 kind: Service 3 metadata: name: carts-db 5 labels: 6 name: carts-db 7 namespace: sock-shop 8 spec: 9 ports: # the port that this service should serve on 10 - port: 27017 11 targetPort: 27017 12 13 selector: 14 name: carts-db

_	manifests/05-catalogue-dep.yaml	
1	l apiVersion: apps/vl	
3	3 metadata:	
4	4 name: catalogue	
5	5 labels:	
6	6 name: catalogue	
8	/ namespace: sock-shop	
9	9 replicas: 2	
10	0 selector:	
11	1 matchLabels:	
12	2 name: catalogue	
13	4 metadata:	
15	5 labels:	
16	6 name: catalogue	
17	7 spec:	
18	8 containers:	
20	0 image: weaveworksdemos/catalogue:0.3.5	
21	1 command: ["/app"]	
22	2 args:	
23	3port=80	
24 25	4 resources: 5 limits:	
26	6 cpu: 200m	
27	7 memory: 200Mi	
28	8 requests:	
29	9 cpu: 100m	
31	1 ports:	
32	2 - containerPort: 80	
33	3 securityContext:	
34	4 runAsNonRoot: true	
35 36	6 capabilities:	
37	7 drop:	
38	8 – all	
39	9 add:	
40 41	U – NET_BIND_SERVICE	
42	2 livenessProbe:	
43	3 httpGet:	
44	4 path: /health	
45 46	port: 80	
47	7 periodSeconds: 3	
48	8 readinessProbe:	
49	9 httpGet:	
50	0 path: /health	
52	1 port: 80 2 initialDelaySeconds: 180	
53	3 periodSeconds: 3	
54	4 nodeSelector:	
55	5 beta.kubernetes.io/os: linux	
	manifests/06-catalogue-syc yaml	
	mannesis, oo catalogue-sve.yann	
1	l apiVersion: vl	
2	2 kind: Service	
3	3 metadata:	
4	4 name: catalogue	
5 6	annotations:	
υ	o prometheus.ro/scrape: true.	

labels: name: catalogue 9 namespace: sock-shop 10 spec: ports: # the port that this service should serve on - port: 80 targetPort: 80 selector: 15

name: catalogue

_	
	manifests/06-catalogue-svc.yaml
1	apiVersion. vl
2	kind: Service
3	metadata:
4	name: catalogue
5	annotations:
6	prometheus.io/scrape: 'true'
7	labels:
8	name: catalogue
9	namespace: sock-shop
10	spec:
11	ports:
12	# the port that this service should serve on
13	- port: 80
14	targetPort: 80
15	selector:
16	name: catalogue

### manifests/07-catalogue-db-dep.yaml

1	apiVersion: apps/vl
2	kind: Deployment
3	metadata:
4	name: catalogue-db
5	labels:
6	name: catalogue-db
7	namespace: sock-shop
8	spec:
9	replicas: 2
10	selector:
11	matchLabels:
12	name: catalogue-db
13	template:
14	metadata:
15	labels:
16	name: catalogue-db
17	spec:
18	containers:
19	- name: catalogue-db
20	<pre>image: weaveworksdemos/catalogue-db:0.3.0</pre>
21	env:
22	- name: MYSQL_ROOT_PASSWORD
23	value: fake_password
24	- name: MYSQL_DATABASE
25	value: socksdb
26	ports:
27	– name: mysql
28	containerPort: 3306
29	nodeSelector:
30	beta.kubernetes.jo/os: linux

#### manifests/08-catalogue-db-svc.yaml

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4
 name: catalogue-db
5
 labels:
6
 name: catalogue-db
 namespace: sock-shop
7
8 spec:
9
 ports:
10
 # the port that this service should serve on
 - port: 3306
11
12
 targetPort: 3306
```

13 selector: 14 name: catalogue-db

# manifests/09-front-end-dep.yaml

	maintests, of none end aep.jum	
1	apiVersion: apps/vl	
$\begin{bmatrix} 2 \\ 2 \end{bmatrix}$	kind: Deployment	
3 4	name: front-end	
5	namespace: sock-shop	
6	spec:	
7	replicas: 1	
8	selector:	
9	matchLabels:	
10	name: iront-end	
11	metadata:	
13	labels:	
14	name: front-end	
15	spec:	
16	containers:	
17	- name: front-end	
18	image: weaveworksdemos/front-end:0.3.12	
$\frac{19}{20}$	resources: limits:	
21	cpu: 300m	
22	memory: 1000Mi	
23	requests:	
24	cpu: 100m	
25	memory: 300Mi	
20	ports: - containerPort: 8079	
28	env:	
29	- name: SESSION_REDIS	
30	value: "true"	
31	securityContext:	
32	runAsNonRoot: true	
34	capabilities:	
35	drop:	
36	- all	
37	readOnlyRootFilesystem: true	
38	livenessProbe:	
39	nttpGet:	
41	port: 8079	
42	initialDelaySeconds: 300	
43	periodSeconds: 3	
44	readinessProbe:	
45	httpGet:	
40	path: /	
48	initialDelaySeconds: 30	
49	periodSeconds: 3	
50	nodeSelector:	
51	beta.kubernetes.io/os: linux	
1		
	manifests/10-front-end-syc yaml	
	mannests/10-nont-end-sve.yann	
Ι.		

1	apiVersion: v1	
2	kind: Service	
3	metadata:	
4	name: front-end	
5	annotations:	
6	prometheus.io/scrape: 'true'	'
7	labels:	
8	name: front-end	
9	namespace: sock-shop	
10	spec:	
11	type: NodePort	
12	ports:	
13	- port: 80	

3352 3353

14	targetPort: 8079
15	nodePort: 30001
16	selector:

16 17 name: front-end

# manifests/11-orders-dep.yaml

3354	manifests/11-orders-dep.yami
3355	
3356	1 apiVersion: apps/v1
0057	2 kind: Deployment
3337	3 metadata:
3358	4 name: orders
3359	5 labels: 6 name: orders
3360	7 namespace: sock-shop
0000	8 spec:
3361	9 replicas: 2
3362	10 selector:
3363	11 matchildets:
226/	13 template:
3304	14 metadata:
3365	15 labels:
3366	16 name: orders
3367	1/ spec:
2260	10 containers: 19 - name: orders
3300	20 image: weaveworksdemos/orders:0.4.7
3369	21 env:
3370	22 - name: JAVA_OPTS
3371	Dspring zinkin enabled = Astrosense = Djava.security.egd=rife:/dev/drandom =
3372	24 resources:
3373	25 limits:
0070	20 cpu: 500m
3374	28 requests:
3375	29 cpu: 100m
3376	30 memory: 300Mi
3377	31 ports:
0077	32 - containerPort: 80
3378	34 ruhashonRoot: true
3379	35 runAsUser: 10001
3380	36 capabilities:
3381	37 arop: 38 - all
3382	39 add:
3383	40 - NET_BIND_SERVICE
3384	42 volumeMounts:
3385	43 - mountPath: /tmp
2206	44 name: tmp-volume 45 volumes:
3300	46 - name: tmp-volume
3387	47 emptyDir:
3388	48 medium: Memory
3389	49 nodeSelector:
3300	JU Dela.Kubernetes.10/OS: 11NUX
3390	
0000	manifects/12 orders ave vam
3392	mannesis/12-oruers-sve.yann

#### manifests/12-orders-svc.yaml

3393 3394

3395

3396

3397

3398

3399

3400

3401

1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: orders 5 annotations: prometheus.io/scrape: 'true' labels: 6 7 , 8 9 name: orders namespace: sock-shop 10 spec: 11 ports:

12 # the port that this service should serve on 13 - port: 80 14 targetPort: 80 15 selector:

15 selector: 16 name: orders

## manifests/13-orders-db-dep.yaml

1	apiVersion: apps/v1
2	kind: Deployment
3	metadata:
4	name: orders-db
5	labels:
6	name: orders-db
7	namespace: sock-shop
8	spec:
9	replicas: 2
10	selector:
11	matchLabels:
12	name: orders-db
13	template:
14	metadata:
15	labels:
16	name: orders-db
17	spec:
18	containers:
19	- name: orders-db
20	image: mongo
21	ports:
22	- name: mongo
23	containerPort: 27017
24	securityContext:
25	capabilities:
26	drop:
27	- all
28	add:
29	- CHOWN
30	- SETGID
31	- SETUID
32	readOnlyRootFilesystem: true
33	volumeMounts:
34	- mountPath: /tmp
35	name: tmp-volume
36	volumes:
37	- name: tmp-volume
38	emptyDir:
39	medium: Memory
40	nodeSelector:
41	beta.kubernetes.io/os: linux

## manifests/14-orders-db-svc.yaml

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4
 name: orders-db
5
 labels:
 name: orders-db
6
7
 namespace: sock-shop
8 spec:
9
 ports:
10
 # the port that this service should serve on
- port: 27017
11
 targetPort: 27017
12
13
 selector:
14
 name: orders-db
```

3456		
3457		manifests/15-payment-dep.yaml
3458		
3459	1	apiVersion: apps/vl
2460	2	kind: Deployment
0404	4	name: payment
3401	5	labels:
3462	6	name: payment
3463	8	spec:
3464	9	replicas: 2
3465	11	selector: matchLabels:
3466	12	name: payment
3467	13	template:
3468	15	labels:
3469	16	name: payment
3/170	17	spec:
0474	19	- name: payment
3471	20	<pre>image: weaveworksdemos/payment:0.4.3</pre>
3472	21	resources: limits:
3473	23	cpu: 200m
3474	24	memory: 200Mi
3475	25	cpu: 99m
3476	27	memory: 100Mi
3477	28	ports: - containerPort: 80
3478	30	securityContext:
3479	31	runAsNonRoot: true
3480	32	runAsUser: 10001 capabilities:
3/81	34	drop:
2401	35	- all
3402	37	- NET_BIND_SERVICE
3483	38	readOnlyRootFilesystem: true
3484	39	livenessProbe:
3485	41	path: /health
3486	42	port: 80
3487	43	periodSeconds: 3
3488	45	readinessProbe:
3489	46	httpGet:
3490	48	port: 80
3491	49	initialDelaySeconds: 180
3492	50	periodseconds: 3 nodeSelector:
3493	52	beta.kubernetes.io/os: linux
3494		
2/05	$\sum$	
3433		manifests/16-payment-svc.yaml
3496		
3497	1	aniVersion, w1
3498	2	kind: Service
3499	3	metadata:
3500	4 5	name: payment annotations:
3501	6	prometheus.io/scrape: 'true'
3502	7	labels:
3503		name. payment namespace: sock-shop
3504	10	spec:
3505	11	ports: # the port that this service should serve on
3506	13	- port: 80
3507	14	targetPort: 80
3508	16	name: payment
0000	1	

_	manifests/17-queue-master-dep.yaml
1	apiVersion: apps/vl
2	kind: Deployment
3	metadata:
5	labels:
6	name: queue-master
7	namespace: sock-shop
8 9	spec: replicas: 2
10	selector:
11	matchLabels:
12	name: queue-master
14	metadata:
15	labels:
16	name: queue-master
17	spec:
19	- name: gueue-master
20	image: weaveworksdemos/queue-master:0.3.1
21	env:
22	- name: JAVA_OPTS
23	Dspring.zipkin.enabled=false
24	resources:
25	limits:
26 27	cpu: 300m memory: 500Mi
28	requests:
29	cpu: 100m
30	memory: 300Mi
31	ports: - containerPort: 80
33	nodeSelector:
34	beta.kubernetes.io/os: linux
_	
	manifests/18-queue-master-svc.yaml
1	aniVersion: v1
2	kind: Service
3	metadata:
4	name: queue-master
6	prometheus.io/scrape: 'true'
7	labels:
8	name: queue-master
9	namespace: sock-shop
11	ports:
12	# the port that this service should serve on
13	- port: 80
14 15	targetPort: 80
15 16	name: gueue-master
10	name. Nacao mascoz
	manifests/19-rabbitmo_den vaml
	mannestor 17 1abbitniq-uep.yann
1	apiVersion: apps/vl
2	kind: Deployment
3 4	netacala:
7	Tame. Fassienty

> name: rabbitmq
5 labels:
6 name: rabbitmq
7 namespace: sock-shop
8 spec:
9 replicas: 2
10 selector:
11 matchLabels:

<pre>13 template: 14 metadata: 15 labels: 16 name: rabbitmq 17 annotations: 18 prometheus.is/scrape: "false" 19 spec: 20 containers: 21 - name: rabbitmq 3.6.8-management 23 ports: 24 - containerPort: 15672 25 name: management 26 - containerPort: 5572 27 name: rabbitmq 28 security/Ontext: 29 computerx: 20 computerx: 20 computerx: 20 computerx: 21 - containerPort: 5672 22 name: rabbitmq 23 add: 33 - CHOWN 34 - SETGID 35 - SETGID 36 - DAC_OVERIDE 37 readOlyMooFFilesystem: true 38 - name: rabbitmq-exporter 39 image: khuberFilesystem: true 38 - name: exporter 40 ports: 41 - containerPort: 9090 42 name: exporter 43 nadeSelector: 44 beta.kubernetes.io/scrape: 'true' 7 prometheus.io/port: '0030' 38 labels: 9 name: rabbitmq 1 apiVersion: v1 2 kind: Service 38 metadata: 4 name: rabbitmq 19 name: rabbitmq 10 name: rabbitmq 11 - port: 9672 13 name: rabbitmq 14 - port: 9672 15 name: rabbitmq 15 annotations: 16 prometheus.io/port: '0030' 17 port: 9090 18 name: exporter 19 targetPort: 6672 17 - port: 9090 18 name: rabbitmq 19 targetPort: s672 17 - port: 9090 18 name: rabbitmq 19 targetPort: s672 19 anne: rabbitmq 10 targetPort: s672 10 anne: rabbitmq 11 anne: rabbitmq 12 anne: rabbitmq 13 anne: rabbitmg 14 anne: rabbitmq 15 anne: rabbitmg 15 anne: rabbitmg 16 targetPort: s672 17 anne: rabbitmq 17 anne: rabbitmq 18 anne: rabbitmq 19 targetPort: s672 19 anne: rabbitmq 10 anne: rabbitmq 11 anne: rabbitmq 12 selector: 13 anne: rabbitmq 14 anne: rabbitmq 15 anne: rabbitmq 15 anne: rabbitmq 15 anne: rabbitmq 16 targetPort: s672 17 anne: rabbitmq 17 anne: rabbitmq 18 anne: rabbitmq 19 anne: rabbitmq 19 anne: rabbitmq 19 anne: rabbitmq 10 anne: rabbitmq 10 anne: rabbitmq 11 anne: rabbitmq 12 anne: rabbitmq 13 anne: rabbitmq 14 anne: rabbitmq 15 anne: rabbitmq 15 anne: rabbitmq 15 anne: rabbitmq 16 anne: rabbitmq 17 anne: rabbitmq 18 anne: rabbitmq 19 anne:</pre>	13	
<pre>14 metadata: 15 labels: 16 name: rabbitmq 17 annotations: 18 prometheus.io/scrape: "false" 19 spec: 20 containers: 21 - name: rabbitmq 22 image: rabbitmq 3.6.8-management 23 ports: 24 - containerPort: 15672 25 name: management 26 - containerPort: 5672 27 name: rabbitmq 28 securityContext: 29 capabilities: 30 drop: 31 - all 32 add: 33 - SETUID 35 - DAC_OVERNIE 36 - DAC_OVERNIE 37 readOnlyBootFilesystem: true 38 - name: rabbitmq-exporter 39 image: kbudde/rabbitmq-exporter 40 ports: 41 - containerPort: 9090 42 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 8 name: rabbitmq 5 name: rabbitmq 6 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmg 15 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: scr2 17 aname: rabbitmg 14 and the port is down and the porter 15 aname: rabbitmg 15 targetPort: scr2 16 targetPort: scr2 17 aname: rabbitmg 18 name: exporter 19 targetPort: scr2 10 proteol: TCP 12 selector: 12 name: rabbitmg</pre>		template:
<pre>15 labels: 16 name: rabbitmq 17 annotations: 18 prometheus.io/scrape: "false" 19 spec: 20 containers: 21 - name: rabbitmq: 3.6.0-management 22 inage: rabbitmq: 23 - containerPort. 15672 24 name: rabbitmq 25 acoutilyContext: 29 capabilities: 20 drop: 20 acpabilities: 20 drop: 21 - all 22 add: 33 - CHOWN 24 - all 25 - OAC_OVERIDE 36 - OAC_OVERIDE 37 - readOnlyRootFilesystem: true 38 - name: rabbitmq-exporter 39 image: kbudde/rabbitmq-exporter 40 ports: 41 - containerPort: 9090 42 name: exporter 43 nodeSelector: 44 beta.kubernetes.io/os: linux 56 prometheus.io/port: '9090' 18 labels: 1 apiVersion: v1 2 kind: Service 38 metadata: 4 name: rabbitmq 1 apiVersion: v1 2 kind: Service 39 name: sobtreq 10 name: sobtreq 11 apiVersion: v1 2 kind: Service 31 name: rabbitmq 11 apiVersion: v1 32 iabels: 33 metadata: 34 the port that this service should serve on 35 annotations: 35 annotations: 36 prometheus.io/port: '9090' 37 name: rabbitmq 38 athe: rabbitmq 39 name: rabbitmq 39 name: rabbitmq 30 name: rabbitmq 30 name: rabbitmq 31 apiCt: 32 name: rabbitmq 33 athe: sporter 34 the port store 35 annotations: 35 annotations: 36 prometheus.io/port: '9090' 37 name: rabbitmq 38 name: rabbitmq 39 annotations: store should serve on 39 athe: sporter 30 annotations: 30 athe: sporter 31 apiCti: 32 name: rabbitmq 33 athe: sporter 34 annotations: 35 annotations: 36 annotations: 37 annotations: 38 athe: sporter 39 athe: sporter 30 athe: rabbitmq 30 athe: sporter 30 athe: rabbitmq 31 annotations: 32 athe: rabbitmq 33 athe: rabbitmq 34 the port store sporter 35 athe: rabbitmq 36 athe: sporter 37 athe: rabbitmq 38 athe: sporter 39 athe: rabbitmq 30 athe: rabbitmq 30 athe: rabbitmq 31 athe: rabbitmq 32 athe: rabbitmq 33 athe: rabbitmq 34 athe: rabbitmq 35 athe: rabbitmq 35 athe: rabbitmq 36 athe: rabbitmq 37 athe: rabbitmq 38 athe: rabbitmq 39 athe: rabbitmq 30 athe: rabbitmq 31 athe: rabbitmq 32 athe: rabbitmq 33 athe: rabbitmq 34 at</pre>	14	metadata:
<pre>16</pre>	15	labels:
<pre>17 annotations: 18 promethews.io/scrape: "false" 19 spec: 20 containers: 21 - name: rabbitmq: 3.6.8-management 23 ports: 24 - containerPort: 15672 25 name: management 26 - containerPort: 5672 27 name: rabbitmq 28 securityContext: 29 capabilities: 20 drop: 31 - all 32 add: 33 - CHOWN 34 - SETGID 35 - SETGID 36 - D.C.OVERRIDE 37 readOnlyRooFFlesystem: true 38 - name: rabbitmq-exporter 39 limage: kbude/rabbitmq-exporter 30 inage: kbude/rabbitmq-exporter 30 inage: kbude/rabbitmq-exporter 30 inage: kbude/rabbitmq-exporter 31 - containerPort: 9090 41 aname: rabbitmq 42 name: rabbitmq 5 annotations: 5 promethews.io/port: '9090' 5 labels: 9 name: rabbitmq 5 annotations: 9 name: rabbitmq 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 9 name: rabbitmq 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 9 name: rabbitmq 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 9 name: rabbitmq 1 apiVersion: v1 2 ports: 13 * the port that this service should serve on 14 - port: S672 15 name: rabbitmq 16 targetPort: S672 17 - port: 9090 17 and exporter 18 selector: 19 targetPort: seprer 10 targetPort: seprer 10 targetPort: seprer 10 selector: 11 apiVersion: 12 porter 12 selector: 13 name: rabbitmq 14 apiVersion: 12 porter 15 name: rabbitmq 15 name: rabbitmq 16 name: rabbitmq 17 apit: 9090 18 apide: 9000 19 apide: 9000 19 apide: 9000 10 ap</pre>	16	name: rabbitmq
<pre>name: rabbitmq spec: containers:</pre>	17	annotations:
<pre>9 spec: containers: 1 - name: rabbitmq:3.6.8-management 23 ports: 4 - containerPort: 15672 5 - name: management 6 - containerPort: 5572 7 - name: rabbitmq 3 - anne: rabbitmq 3 - all 3 - all 3 - all 3 - all 3 - chown 4 - SETGID 3 bAC_OVERRIDE 7 - readOnlyRootFilesystem: true 9 - image: kbudde/rabbitmq-exporter 1 - ontainerPort: 9090 1 - anne: rabbitmq 7 - readOnlyRootFilesystem: true 9 - image: kbudde/rabbitmq-exporter 1 - ontainerPort: 9090 1 - anne: crabbitmq-exporter 4 - containerPort: 9090 1 - mame: rabbitmq 9 - name: rabbitmq 1 apiVersion: vl 2 kind: Service 1 apiVersion: vl 2 hame: rabbitmq 1 apiVersion: vl 2 kind: Service 3 metadta: 1 ane: rabbitmq 1 apiVersion: vl 2 kind: Service 3 metadta: 1 ane: rabbitmq 1 apiVersion: vl 2 kind: Service 3 metadta: 1 ane: rabbitmq 1 an</pre>	18	prometheus.lo/scrape: "false"
<pre>20 containers: 21 name: rabbitmq 22 image: rabbitmq36.8-management 23 ports: 24 containerPort: 15672 25 name: rabbitmq 26 c- containerPort: 5572 27 name: rabbitmq 28 securityContext: 29 c- capabilities: 30 drop: 31 all 32 add: 3 COOW 34 SETGID 35 SETGID 36 DAC_OVERIDE 37 readOn1PACOFFILesystem: true 38 name: rabbitmq-exporter 39 image: kbude/rabbitmq-exporter 40 ports: 41 containerPort: 9090 42 name: exporter 43 nodeSelector: 44 beta.kubernetes.io/os: linux 54 rame: rabbitmq 5 annotalions: 44 rame: rabbitmq 5 annotalions: 45 labels: 9 name: rabbitmq 1 apiVersion: v1 2 kind: Service 3 metadata: 9 name: rabbitmq 1 apiVersion: v2 2 labels: 9 name: rabbitmq 1 apiVersion: v2 3 labels: 10 namespace: sock-shop 11 spec: 12 ports: 13</pre>	19	spec:
<pre>1 - name: rabbitmq:3.6.8-management 2     ports: 2     rontainerPort: 15672 3     name: management 2     containerPort: 5672 7     name: rabbitmq 2     capabilities: 3</pre>	20	containers:
<pre>image: rabblemg.rabblemg.rabblemg. prosts: - containerPort: 15672 - containerPort: 5672 - containerPort: 5672 - containerPort: 5672 - containerPort: 5672 - containerPort: 5672 - containerPort: 900 - sETGID - sETGID - sETGID - sETUID - setUitmg-exporter - containerPort: 9090 - name: rabbitmg-exporter - containerPort: 9090 - name: rabbitmg-svc.yaml manifests/20-rabbitmg-svc.yaml - containerPort: 9090 - name: rabbitmg - name: rabbitmg</pre>	21	- name: rabbitmg:2 6 9 management
<pre>puts: puts: containerPort: 15672 name: management containerPort: 5672 name: rabbitmq securityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: generityContext: g</pre>	22	Inage: rabblend:5.6.8-management
<pre></pre>	23	ports:
<pre>immer immersor: 5672 containerPort: 5672 capabilities: capabilities</pre>	25	
<pre>context and the second se</pre>	26	- containerPort - 5672
<pre>securityContext: generative addition additi</pre>	27	name rabitmo
<pre>29</pre>	28	securityContext:
<pre>30 drop: 31 - all 32 add: 33 - CHOWN 44 - SETGID 55 - DAC_OVERRIDE 56 - DAC_OVERRIDE 57 readOnlyRootFilesystem: true 58 - name: rabbitmq-exporter 59 image: kbudde/rabbitmq-exporter 50 ports: 50 - containerPort: 9090 51 andesSelector: 52 beta.kubernetes.io/os: linux 53 metadata: 54 name: rabbitmq 54 anotations: 55 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 9 name: rabbitmq 1 apiVersion: 54 anotations: 55 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 9 name: rabbitmq 1 aname: rabbitmq 1 apiversion: 56 prometheus.io/scrape: 'true' 77 ports: 57 name: rabbitmq 58 athels: 59 name: rabbitmq 50 name: rabbitmg 50 targetPort: \$672 50 name: exporter 50 protocol: TCP 50 selector: 50 name: rabbitmq 51 selector: 52 name: rabbitmg 53 aname: rabbitmg 54 targetPort: \$672 55 name: rabbitmg 55 na</pre>	29	capabilities:
<pre>31 - 'all 32 add: 33 - CHONN 34 - SETGID 35 - SETUID 36 - DAC_OVERIDE 37 readOnlyRootFilesystem: true 38 - name: rabbitmq-exporter 39 image: kbudde/rabbitmq-exporter 30 ords: 41 - containerPort: 9090 41 - containerPort: 9090 42 name: exporter 43 nodeSelector: 44 beta.kubernetes.io/os: linux manifests/20-rabbitmq-svc.yaml 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9030 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	30	drop:
<pre>32</pre>	31	- all
<pre>33 - CHOWN 34 - SETGID 35 - SETUID 36 - DAC_OVERRIDE 37 readOnlyRootFilesystem: true 38 - name: rabbitmq-exporter 40 ports: 41 - containerPort: 9090 42 name: exporter 43 nodeSelector: 44 beta.kubernetes.io/os: linux manifests/20-rabbitmq-svc.yaml 1 apiVersion: v1 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 ptocol: TCP 20 spector: 21 portocol: TCP 21 selector: 22 name: rabbitmq 3 anne: rabbitmq 4 name: rabbitmq 5 name: rabbitmg 5 nam</pre>	32	add:
<pre>34 - SETCID 35 - SETCID 36 - DAC_OVERRIDE 37 readOnlyRootFilesystem: true 39 image: kbudde/rabbitmq-exporter 39 image: kbudde/rabbitmq-exporter 30 ports: 41 - containerPort: 9090 42 name: exporter 44 beta.kubernetes.io/os: linux <b>manifests/20-rabbitmq-svc.yaml</b> 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 1 apiVersion: v1 2 kind: Service should serve on 1 spec: 12 ports: 13 # the port that this service should serve on 1 - port: S672 15 name: rabbitmq 1 apiVersion: exporter 15 prometheus.io/port: '1000' 16 aprespace: sock-shop 17 - port: S672 18 name: rabbitmq 19 name: rabbitmq 19 name: rabbitmq 10 name: port: S672 11 port: S672 12 port: S672 13 af the port that this service should serve on 14 - port: S672 15 name: rabbitmq 16 targetPort: S672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 10 targetPort: s672 11 spect: 22 name: rabbitmq 33 af the porter 34 af the porter 34 af the porter 35 anatice true the porter 36 anatice true true the porter 37 anatice true true true true true true true tru</pre>	33	- CHOWN
<pre>35 - SETUID 36 - DAC_OVERIDE 37 readOnlyRootFilesystem: true 38 - name: rabbitmq-exporter 39 jimage: kbudde/rabbitmq-exporter 40 ports: 41 - containerPort: 9090 41 name: exporter 42 nodeSelector: 44 beta.kubernetes.io/os: linux <b>manifests/20-rabbitmq-svc.yaml</b> 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 anotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 1 spec: 12 ports: 14 the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 10 name: rabbitmq 10 name: exporter 10 targetPort: scr2 10 name: rabbitmq 11 apiCall Selector: 12 ports: 13 met rabbitmq 14 targetPort: scr2 15 name: rabbitmq 15 name: rabbitmq 16 targetPort: scr2 17 - port: 9090 18 name: rabbitmq 18 name: rabbitmq 19 name: rabbitmq 10 na</pre>	34	- SETGID
<pre>36 - DAC_OVERNIDE 7 readOnJRAOTFILESystem: true 38 - name: rabbitmq-exporter 39 image: kbudde/rabbitmq-exporter 30 opers: 41 - containerPort: 9090 42 name: exporter 43 nodeSelector: 44 beta.kubernetes.io/os: linux manifests/20-rabbitmq-svc.yaml 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 1 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 10 name: exporter 11 spect: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: seporter 17 - port: 9090 18 name: rabbitmq 19 targetPort: exporter 10 spector: 21 selector: 21 name: rabbitmq 31 settor: 32 name: rabbitmq 33 name: rabbitmq 34 name: rabbitmq 35 name: rabbitmq 35 name: rabbitmq 36 name: rabbitmq 37 name: rabbitmq 38 name: rabbitmq 39 name: rabbitmq 30 name: rabbitmq 30 name: rabbitmq 31 settor: 32 name: rabbitmq 33 settor: 34 name: rabbitmq 35 name: rabbitmq 35 name: rabbitmq 36 name: rabbitmq 37 name: rabbitmq 37 name: rabbitmq 38 name: rabbitmq 39 name: rabbitmq 30 name: rabbitmq 30 name: rabbitmq 30 name: rabbitmq 31 name: rabbitmq 31 name: rabbitmq 32 name: rabbitmq 33 name: rabbitmq 34 name: rabbitmq 35 name: rabbitmq 35 name: rabbitmq 36 name: rabbitmq 37 name: rabbitmq 38 name: rabbitmq 38 name: rabbitmq 39 name: rabbitmq 30 name: rabbitmq 30 name: rabbitmq 30 name: rabbitmq 30 name: rabbitmq 31 name: rabbitmq 32 name: rabbitmq 33 name: rabbitmq 34 name: rabbitmq 35 name: rabbitmq 36 name: rabbitmq 37 name: rabbitmq 38 name: rabbitmq 38 name: rabbitmq 39 name: rabbitmq 30 name: rabbitmq 31 name: rabbitmq 31 name: rabbitmq 32 n</pre>	35	- SETUID
<pre>37 readOnlyRootFilesystem: true 38 - name: rabbitmq-exporter 39 image: kbudde/rabbitmq-exporter 40 ports: 41 - containerPort: 9090 42 name: exporter 43 nodeSelector: 44 beta.kubernetes.io/os: linux</pre>	36	- DAC_OVERRIDE
<pre>38 - name: rabbitmq-exporter 99 image: kbudde/rabbitmq-exporter 90 ports: 11 - containerPort: 9090 1 aname: exporter 13 nodeSelector: 14 beta.kubernetes.io/os: linux 1 apiVersion: v1 2 kind: Service 3 metadata: 1 anamet rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 1 apec: 10 ports: 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 10 protecl: TCP 11 spec: 21 ports: 21 aname: rabbitmq 22 name: rabbitmq 23 name: rabbitmq 24 name: rabbitmq 25 name: rabbitmq 25 name: rabbitmq 26 name: rabbitmq 27 name: rabbitmq 28 name: rabbitmq 29 name: rabbitmq 20 name: rabbitmq 20 name: rabbitmq 20 name: rabbitmq 21 name: rabbitmq 22 name: rabbitmq 23 name: rabbitmq 24 name: rabbitmq 25 name: rabbitmq 26 name: rabbitmq 27 name: rabbitmq 28 name: rabbitmq 29 name: rabbitmq 20 name: rabbitmq 21 selector: 22 name: rabbitmq 23 name: rabbitmq 24 name: rabbitmq 25 name: rabbitmq 26 name: rabbitmq 27 name: rabbitmq 28 name: rabbitmq 29 name: rabbitmq 20 name:</pre>	37	readOnlyRootFilesystem: true
<pre>39 image: kbudde/rabbitmq-exporter 40 ports: 41 - containerPort: 9090 42 name: exporter 43 nodeSelector: 44 beta.kubernetes.io/os: linux manifests/20-rabbitmq-svc.yaml 1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 19 targetPort: exporter 19 targetPort: mane: rabbitmq 10 name: rabbitmq 11 apiCenter 2000 12 selector: 13 mane: rabbitmq 14 name: rabbitmq 15 name: rabbitmq 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: rabbitmq 19 targetPort: exporter 19 targetPort: exporter 19 targetPort: exporter 10 protocol: TCP 10 selector: 11 name: rabbitmq 12 selector: 13 name: rabbitmq 14 name: rabbitmq 15 name: rabbitmq 15 name: rabbitmq 16 name: rabbitmq 17 name: rabbitmq 18 name: rabbitmq 19 name: rabbitmq 10 name: ra</pre>	38	- name: rabbitmq-exporter
<pre>40     ports: 1     - containerPort: 9090 42     name: exporter 33     nodeSelector: 44     beta.kubernetes.io/os: linux manifests/20-rabbitmq-svc.yaml 1 apiVersion: v1 2 kind: Service 3 metadata: 4     name: rabbitmq 5     annotations: 6     prometheus.io/scrape: 'true' 7     prometheus.io/port: '9090' 8     labels: 9     name: rabbitmq 10     namespace: sock-shop 11 spec: 12     ports: 13     # the port that this service should serve on 14     - port: 5672 15     name: rabbitmq 16     targetPort: 5672 17     - port: 9090 18     name: exporter 19     targetPort: exporter 19     targetPort: exporter 10     anme: rabbitmq 11     service 12     port: 9090 13     name: rabbitmq 14     targetPort: scorter 15     port: 9090 16     targetPort: exporter 17     port: 9090 18     name: exporter 19     targetPort: exporter 19     targetPort: exporter 10     protocol: TCP 11     selector: 21     name: rabbitmq 22     ports: 23     name: rabbitmq 24     port: 900 25      name: rabbitmq 26     protocol: TCP 27     potcol: TCP 28     potcol: TCP 29     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 21     potcol: TCP 22     potcol: TCP 23     pane: rabbitmq 24     potcol: TCP 25     potcol: TCP 26     potcol: TCP 27     potcol: TCP 28     potcol: TCP 29     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 21     potcol: TCP 22     potcol: TCP 23     potcol: TCP 24     potcol: TCP 25     potcol: TCP 25     potcol: TCP 26     potcol: TCP 27     potcol: TCP 28     potcol: TCP 29     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 21     potcol: TCP 22     potcol: TCP 23     potcol: TCP 24     potcol: TCP 25     potcol: TCP 26     potcol: TCP 27     potcol: TCP 28     potcol: TCP 29     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 20     potcol: TCP 2</pre>	39	image: kbudde/rabbitmq-exporter
<pre>41 - containerPort: 9090     name: exporter 42    nodeSelector: 44    beta.kubernetes.io/os: linux  44    manifests/20-rabbitmq-svc.yaml  55    labels: 56    prometheus.io/scrape: 'true' 77    prometheus.io/port: '9090' 18    labels: 50    name: rabbitmq 10    namespace: sock-shop 11 spec: 12    ports: 13</pre>	40	ports:
<pre>12</pre>	41	- containerPort: 9090
<pre>manifests/20-rabbitmq-svc.yaml  apiVersion: v1 kind: Service metadata: name: rabbitmq name: rabbitmq name: rabbitmq name: rabbitmq namespace: sock-shop spec: ports:</pre>	42	name: exporter
<pre>manifests/20-rabbitmq-svc.yaml  apiVersion: v1 kind: Service manations: manations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: mame: rabbitmq namespace: sock-shop spec: ports:</pre>	43 44	hote selector:
<pre>manifests/20-rabbitmq-svc.yaml  apiVersion: v1 kind: Service metadata: ame: rabbitmq annotations:     prometheus.io/scrape: 'true'     prometheus.io/port: '9090' labels:     name: rabbitmq namespace: sock-shop ls spec:     ports:     # the port that this service should serve on     - port: 5672 name: rabbitmq lame: exporter     targetPort: exporter     ports: TCP l selector:     name: rabbitmq </pre>	44	beta.Kubernetes.10/05. IInux
<pre>manifests/20-rabbitmq-svc.yaml  1 apiVersion: v1 2 kind: Service 3 metadata: 4   name: rabbitmq 5   annotations: 6      prometheus.io/scrape: 'true' 7      prometheus.io/port: '9090' 8   labels: 9      name: rabbitmq 10   namespace: sock-shop 11 spec: 12   ports: 13</pre>		
<pre>manifests/20-rabbitmq-svc.yaml  1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13  # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq </pre>		
<pre>1 apiVersion: v1 2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>		manifests/20-rabbitmq-svc.yami
<pre>1 apiVersion: v1 2 kind: Service 3 metadata: 4  name: rabbitmq 5  annotations: 6     prometheus.io/scrape: 'true' 7     prometheus.io/port: '9090' 8 labels: 9     name: rabbitmq 10     namespace: sock-shop 11 spec: 12     ports: 13</pre>		
<pre>2 kind: Service 3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	1	apiVersion: v1
<pre>3 metadata: 4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq 24 and 25 and 2</pre>	2	kind: Service
<pre>4 name: rabbitmq 5 annotations: 6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	3	metadata:
<pre>5 annotations: 6         prometheus.io/scrape: 'true' 7         prometheus.io/port: '9090' 8 labels: 9         name: rabbitmq 10         namespace: sock-shop 11 spec: 12         ports: 13</pre>	4	name: rabbitmq
<pre>6 prometheus.io/scrape: 'true' 7 prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	•	
<pre>/ prometheus.io/port: '9090' 8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13  # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq </pre>	5	annotations:
<pre>8 labels: 9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13</pre>	5	annotations: prometheus.io/scrape: 'true'
<pre>9 name: rabbitmq 10 namespace: sock-shop 11 spec: 12 ports: 13  # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7	annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090'
<pre>namespace: sock-snop spec: ports:     # the port that this service should serve on     # the port that this service should serve on     # the port that this service should serve on     # the port sof72     name: rabbitmq     targetPort: 5672     protocol: TCP     selector:     name: rabbitmq </pre>	5 6 7 8	annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels:
<pre>11 spec: 12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7 8 9	annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq
<pre>12 ports: 13 # the port that this service should serve on 14 - port: 5672 15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7 8 9 10	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop</pre>
<pre>13  # the port that this service should serve on 14  - port: 5672 15  name: rabbitmq 16  targetPort: 5672 17  - port: 9090 18  name: exporter 19  targetPort: exporter 20  protocol: TCP 21  selector: 22  name: rabbitmq</pre>	5 6 7 8 9 10 11	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: prometheus.io/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/species/</pre>
<pre>15 name: rabbitmq 16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7 8 9 10 11 12 13	<pre>annotations:</pre>
<pre>16 targetPort: 5672 17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7 8 9 10 11 12 13 14	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672</pre>
<pre>17 - port: 9090 18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7 8 9 10 11 12 13 14	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmg</pre>
<pre>18 name: exporter 19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7 8 9 10 11 12 13 14 15 16	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672</pre>
<pre>19 targetPort: exporter 20 protocol: TCP 21 selector: 22 name: rabbitmq</pre>	5 6 7 8 9 10 11 12 13 14 15 16 17	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090</pre>
20 protocol: TCP 21 selector: 22 name: rabbitmq	5 6 7 8 9 10 11 12 13 14 15 16 17 18	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090 name: exporter</pre>
21 selector: 22 name: rabbitmq	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090 name: exporter targetPort: exporter</pre>
22 name: rabbitmq	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090 name: exporter targetPort: exporter targetPort: TCP</pre>
	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090 name: exporter targetPort: exporter protcol: TCP selector:</pre>
	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090 name: exporter targetPort: exporter targetPort: exporter protocol: TCP selector: name: rabbitmq</pre>
	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090 name: exporter targetPort: exporter protocol: TCP selector: name: rabbitmq</pre>
	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	<pre>annotations: prometheus.io/scrape: 'true' prometheus.io/port: '9090' labels: name: rabbitmq namespace: sock-shop spec: ports: # the port that this service should serve on - port: 5672 name: rabbitmq targetPort: 5672 - port: 9090 name: exporter targetPort: exporter protocol: TCP selector: name: rabbitmq</pre>

#### manifests/21-session-db-dep.yaml

1 apiVersion: apps/v1
2 kind: Deployment 3 metadata: 4 name: session-db 5 labels: 6 name: session-db 7 namespace: sock-shop 8 spec: 9 replicas: 2 10 selector: 11 matchLabels: 12 name: session-db

3608 3609

3610

3611

3612

3613

3614

3615

3616

3 4 5	template: metadata:
4 5	metadata:
5	
	labels:
.6	name: sesion-db
. /	annotations: prometheus io scrape: "false"
9	sper.
20	containers:
21	- name: session-db
2	image: redis:alpine
:3	ports:
24	- name: redis
25 26	containerPort: 63/9
.0 7	capabilities:
28	drop:
29	- all
60	add:
\$1	- CHOWN
52 12	- SETGID
24	- SETULD
14 15	readoniykootrilesystem: true
36	beta kubernetes jo/os· linux
.0	beed waberneees 10, 05. Think
1	manifests/22-session-db-svc.yaml
	· · · · · · · · · · · · · · · · · · ·
1 8	apiVersion: vl
2 1	kind: Service
5 I 4	metadata:
4 5	labels.
6	name: session-db
7	namespace: sock-shop
	A A
8 :	spec:
8 s 9	spec: ports:
8 9 0	<pre>spec: ports: # the port that this service should serve on</pre>
8 9 0 1	<pre>spec: ports: # the port that this service should serve on - port: 6379 topretPort. 6270</pre>
8 9 0 1 2 3	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector:</pre>
8 9 0 1 2 3 4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db</pre>
8 9 0 1 2 3 4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db</pre>
8 9 0 1 2 .3 4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db</pre>
8 9 0 1 2 .3 .4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db</pre>
8 9 10 .1 .2 .3 .4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml</pre>
8 9 10 1 2 .3 .4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml</pre>
8 9 0 1 2 3 4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml priVersion: epre/ctl</pre>
8 + 9 9 10 1 - 2 .3 .4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment</pre>
8 + 9 9 10 11 .2 .3 .4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata:</pre>
8 : 9 9 0 11 : 2 .3 .4 1 : 2 2 : 1 3 : 1 4	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping</pre>
8 : 9 9 0 1 1 2 2 .3 .4 1 2 2 1 3 r 4 5	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels:</pre>
8 : 9 9 0 11 12 .3 .4 1 2 1 3 1 4 5 6 6	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping</pre>
8 = 9 9 0 1 1 2 3 .4 1 2 2 1 3 1 4 5 6 7 7 0	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping name: shipping name: shipping name: sock-shop</pre>
8 : 9 9 10 11 22 3 .4 1 2 3 r 4 5 6 6 7 7 8 s	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping name: shipping name: shipping name: shipping name: sock-shop spec: replicate: 2</pre>
8 : 9 9 0 1 1 2 2 3	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector:</pre>
8 : 9 9 10 11 12 13 13 4 1 2 13 1 1 2 13 1 1 2 1 3 1 1 4 5 6 7 8 8 9 0 1	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matriblabels:</pre>
8 9 10 12 13 4 1 2 13 3 1 4 5 6 7 8 9 0 1 2	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping</pre>
8       1         1       2         1       2         1       3         1       4         1       5         6       7         8       5         9       0         1       2         3       1         2       1         3       1         2       2         3       1         2       3	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template:</pre>
8       1         1       2         13       14         1       3         1       3         1       4         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3         1       3 <td< th=""><th><pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: mame: shipping template: metadata:</pre></th></td<>	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: mame: shipping template: metadata:</pre>
8       1         9       10         11       12         13       14         13       14         14       15         6       7         8       9         0       1         2       3         1       2         3       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       5	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: metadata: labels: treplate: metadata: labels: treplate: metadata: labels: treplate: metadata: labels: treplate:</pre>
8       1         12       13         14       1         1       2         1       4         5       6         7       8         9       0         1       2         3       1         2       3         4       5         6       7         8       5         6       7         6       7         7       5         6       7	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: mame: shipping template: template: template: template: template: template: template: template: template: template:</pre>
8       1         9       0         11       1         12       1         13       1         14       1         13       1         14       1         15       6         7       8         9       0         12       3         4       5         6       7         8       6         7       8         9       0         12       3         4       5         6       7         8       6         7       8         9       0         12       3         4       5         6       7         8       8         9       0         12       3         4       5         6       7         8       8         1       1         1       1         1       1         1       1         1       1         1       1      1	<pre>spec: ports: # the port that this service should serve on - port: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: metadata: labels: name: shipping spec: containers:</pre>
8       1         9       0         1       1         12       1         13       1         1       1         1       2         1       1         1       2         1       1         1       2         3       1         1       2         3       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: name: shipping template: name: shipping spec: containers: _ name: shipping spec: containers: _ name: shipping</pre>
8       -         9       0         1       1         12       1         3       -         4       -         5       6         7       8         9       0         1       2         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       0	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db</pre>
8       :         9       0         1       1         1       2         1       3         1       2         1       3         1       2         1       3         1       2         1       2         1       2         1       2         1       2         1       2         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       2         3       4         5       6         7       8         9       0         1       1	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: name: shipping template: name: shipping template: name: shipping spec: containers: - name: shipping image: weaveworksdemos/shipping:0.4.8 env:</pre>
8       :         9       0         12       13         14       1         1       2         13       14         1       2         13       14         1       2         13       1         2       3         1       2         3       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       2	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: mate: shipping template: mate: shipping spec: containers: - name: shipping image: weaveworksdemos/shipping:0.4.8 env: - name: ZIPKIN</pre>
8       :         9       0         1       12         13       14         1       23         14       1         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23         1       23	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: metadata: labels: name: shipping template: mate: shipping spec: containers: - name: shipping image: weaveworksdemos/shipping:0.4.8 env: - name: ZIPKIN value: zipkin.jaeger.svc.cluster.local</pre>
8       :         9       0         1       12         12       3         14       1         12       3         14       1         12       3         14       1         12       3         12       3         13       4         5       6         7       8         9       0         1       2         3       4         5       6         7       8         9       0         1       2         3       4         5       6         7       8         9       0         1       2         3       4         5       6         7       8         9       0         1       2         3       4         5       3         6       7         8       9         9       1         1       2         1       2	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping labels: name: shipping template: matchLabels: name: shipping template: mate: shipping template: mate: shipping template: mame: shipping spec: containers: - name: shipping image: weaveworksdemos/shipping:0.4.8 env:</pre>
8       :         9       0         1       12         12       3         14       1         1       2         1       3         1       2         1       3         1       2         1       3         1       2         1       3         1       2         1       3         1       2         1       3         1       2         2       3         1       2         1       2         2       3         1       2         1       2         1       2         2       3         1       2         2       3         2       3         2       3         2       3         2       3         2       3         2       3         2       3         2       3         2       3         2       3 <t< th=""><th><pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml matic shipping labels: name: shipping labels: name: shipping labels: name: shipping labels: name: shipping labels: name: shipping template: matchtabels: name: shipping template: rename: shipping spec: containers: - name: shipping image: weaveworksdemos/shipping:0.4.8 env:</pre></th></t<>	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml matic shipping labels: name: shipping labels: name: shipping labels: name: shipping labels: name: shipping labels: name: shipping template: matchtabels: name: shipping template: rename: shipping spec: containers: - name: shipping image: weaveworksdemos/shipping:0.4.8 env:</pre>
8       :         9       0         11       1         12       1         13       4         1       2         1       3         1       2         1       3         1       2         1       3         1       2         1       3         1       2         1       3         1       2         1       3         1       2         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         0       1         2       3         4       5         6       7         8       9         1       2         2       3         4       5         6       7         8       9         1       2 <t< th=""><th><pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping name: shipping name: shipping templica: 2 selector: matchLabels: name: shipping templica: 2 selector: matchLabels: name: shipping templice: metadata: labels: name: shipping spec: containers: - name: Shipping image: veaveorksdemos/shipping:0.4.8 env: - name: JIPKIN value: zipkin.jaeger.svc.cluster.local - name: JAVA_OPTS value: zipkin.jaeger.svc.cluster.local - name: JAVA_OPTS value: .zipkin.enabled=false</pre></th></t<>	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping name: shipping name: shipping templica: 2 selector: matchLabels: name: shipping templica: 2 selector: matchLabels: name: shipping templice: metadata: labels: name: shipping spec: containers: - name: Shipping image: veaveorksdemos/shipping:0.4.8 env: - name: JIPKIN value: zipkin.jaeger.svc.cluster.local - name: JAVA_OPTS value: zipkin.jaeger.svc.cluster.local - name: JAVA_OPTS value: .zipkin.enabled=false</pre>
8       :         9       0         12       1         12       1         12       1         12       1         12       1         12       1         12       1         12       1         12       1         12       3         12       3         12       3         12       3         12       3         12       3         12       3         12       3         13       4         5       6         7       8         9       0         12       3         12       3         14       5         6       7         8       9         12       3         13       4         14       5         15       6         16       7         17       12         18       12         19       12         10       12         11       12	<pre>spec: ports: # the port that this service should serve on - port: 6379 targetPort: 6379 selector: name: session-db manifests/23-shipping-dep.yaml apiVersion: apps/v1 kind: Deployment metadata: name: shipping labels: name: shipping namespace: sock-shop spec: replicas: 2 selector: matchLabels: name: shipping template: metadata: labels: name: shipping template: metadata: labels: name: shipping spec: containers: - name: shipping image: weaveworksdemos/shipping:0.4.8 env: - name: ZIPKIN value: ZIPKIN value: JANA_OPTS value: -Xms64m -Xmx128m -XX;+UseGICC -Djava.security.egd=file:/dev/urandom - Dspring.ipkin.enabled=false resources: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits: limits:</pre>

cpu: 300m memory: 500Mi requests: cpu: 100m memory: 300Mi ports: - containerPort: 80 securityContext: runAsNonRoot: true runAsUser: 10001 capabilities: drop: - all add: - NET_BIND_SERVICE readOnlyRootFilesystem: true volumeMounts: - mountPath: /tmp name: tmp-volume volumes: - name: tmp-volume emptyDir: medium: Memory nodeSelector: beta.kubernetes.io/os: linux manifests/24-shipping-svc.yaml 1 apiVersion: v1 kind: Service 3 metadata: name: shipping annotations: prometheus.io/scrape: 'true' labels: name: shipping namespace: sock-shop 10 spec: ports: # the port that this service should serve on - port: 80 targetPort: 80 selector: name: shipping manifests/25-user-dep.yaml 1 apiVersion: apps/v1 kind: Deployment metadata: name: user labels: name: user namespace: sock-shop 8 spec:

replicas: 2 selector: matchLabels: name: user template: metadata: labels: name: user spec: containers: - name: user image: weaveworksdemos/user:0.4.7 resources: 23 limits: cpu: 300m 25 memory: 200Mi

requests:

	26		
121	20	cpu: 100m	
728	2/	memory: 100M1	
	28	ports:	
729	30	- Containerport: 80	
730	31	- name: mongo	
731	32	value: user-db:27017	
720	33	securityContext:	
32	34	runAsNonRoot: true	
33	35	runAsUser: 10001	
24	36	capabilities:	
34	37	drop:	
35	38	- all	
26	39	add:	
50	40	- NET_BIND_SERVICE	
37	41	readOnlyRootFilesystem: true	
20	42	livenessProbe:	
50	43	httpGet:	
39	44	path: /health	
10	45	port: 80	
10	46	initialDelaySeconds: 300	
11	4/	periodSeconds: 3	
2	48	readinessProbe:	
-	49	nttpGet:	
3	50	path: /health	
4	51	port: Bu	
_	52	InitialDelaySeconds: 180	
C	54	perfouseconds. 5	
5	55	hote kubernetes is/ost linux	
7	55	Deta.Kubernetes.10/05. Tinuk	
1			
8			-
9	ma	anifests/26-user-svc.yaml	
50			٦
51	1 200	iVersion. vl	
0	2 ki	nd: Service	
~	3 met	tadata:	
53	4	name: user	
54	5	annotations:	

4	name: user
5	annotations:
6	prometheus.io/scrape: 'true'
7	labels:
8	name: user
9	namespace: sock-shop
10	spec:
11	ports:
12	# the port that this service should serve on
13	- port: 80
14	targetPort: 80
15	selector:
16	name: user

# manifests/27-user-db-dep.yaml

3765		
3766	1	apiVersion: apps/vl
3767	2	kind: Deployment
3768		metadata:
2760	5	labels:
3709	6	name: user-db
3770	7	namespace: sock-shop
3771	8	spec:
0770	9	replicas: 2
3772	10	selector:
3773	11	matchLabels:
0774	12	name: user-db
3774	13	template:
3775	14	metadata:
2776	15	labels:
3//0	16	name: user-db
3777	17	spec:
3778	18	containers:
5110	19	- name: user-db
3779	20	<pre>image: weaveworksdemos/user-db:0.3.0</pre>

21	ports:
22	- name: mongo
23	containerPort: 27017
24	securityContext:
25	capabilities:
26	drop:
27	- all
28	add:
29	- CHOWN
30	- SETGID
31	- SETUID
32	readOnlyRootFilesystem: true
33	volumeMounts:
34	<pre>- mountPath: /tmp</pre>
35	name: tmp-volume
36	volumes:
37	- name: tmp-volume
38	emptyDir:
39	medium: Memory
40	nodeSelector:
41	beta.kubernetes.io/os: linux

#### manifests/28-user-db-svc.yaml

1 apiVersion: v1 2 kind: Service 3 metadata: name: user-db labels: name: user-db namespace: sock-shop 8 spec: ports:  $\ensuremath{\texttt{\#}}$  the port that this service should serve on - port: 27017 targetPort: 27017 selector: name: user-db

#### C.2.2 OUTPUTS

VaC script for the steady state #1 (carts-db-dep.yaml): At least 1 ready replica 100% of the time and 2 ready replicas 80% of the time during the monitoring period 1 import os
2 import time
3 import argparse
4 from kubernetes import client, config
5 from unittest_base import K8sAPIBase
6
7 class TestCartsDBReplicas(K8sAPIBase):
8 def __init__(self, namespace='sock-shop', deployment_name='carts-db', duration=5):
9 super().__init__()

```
10
 self.namespace = namespace
 11
 self.deployment_name = deployment_name
 12
 self.duration = duration
 13
 # Use the correct API client for deployments
 14
 self.apps_v1 = client.AppsV1Api()
 15
 16
 def test_replicas_threshold(self):
 17
 # Initialize counters for ready replicas
 total_checks = 0
 18
 19
 ready_replicas_count = 0
 20
 fully_ready_replicas_count = 0
 21
 22
 for _ in range(self.duration):
 23
24
 try:
 # Read the deployment status using the correct API client
 25
 deployment = self.apps_v1.read_namespaced_deployment(self.
 deployment_name, self.namespace)
3833
```

26	replicas = deployment.status.replicas
27	ready_replicas = deployment.status.ready_replicas
.8	<pre>print(f"Total replicas: {replicas}, Ready replicas: {ready_replicas}")</pre>
	# Increment the total checks
	total_checks += 1
	<pre># Check if at least 1 replica is ready</pre>
	ready replicas >= 1: ready replicas count += 1
	Icadcopiicab_count ( 1
	<pre># Check if both replicas are ready</pre>
	<pre>if ready_replicas == 2:     fully ready replicas count += 1</pre>
	rarry_ready_repress_count ( r
	<b>except</b> client.exceptions.ApiException as e:
	<pre>print(f"Exception when calling AppsVlApi-&gt;read_namespaced_deployment: {</pre>
	<pre># Wait for 1 second before the next check</pre>
	time.sleep(1)
	# Calculate the percentage of time conditions are met
	one_ready_percentage = (ready_replicas_count / total_checks) * 100
	<pre>two_ready_percentage = (fully_ready_replicas_count / total_checks) * 100</pre>
	# Assart the threshold conditions
	assert one_ready_percentage == 100, "At least 1 ready replica was not available
	100% of the time."
	assert two_ready_percentage >= 80, "2 ready replicas were not available at
	TEASE 00% OF CHE CIME.
	<pre>print("Test passed: Steady state conditions are satisfied.")</pre>
lei	f main()·
	parser = argparse.ArgumentParser(description='Test carts-db replicas threshold')
	parser.add_argument('duration', <b>type=int</b> , default=5, <b>help=</b> 'Duration to check the
	replicas') args = parser parse args()
	# Create an instance of the test class with the specified duration
	test = lestcartsDBReplicas(duration=args.duration) # Run the test
	test.test_replicas_threshold()
if	name == ' main '.
Va	C script for the steady state #2 (front-end-dep.yaml): At least 1 ready replica must be
pre	esent 100% of the time during the monitoring period
r	
im	port os
im	port time
fre	on kubernetes import client, config
fro	om unittest_base import K8sAPIBase
-	
cla	ass TestFrontEndReplica(K8sAPIBase):
	super(). init ()
	self.namespace = namespace
	<pre>self.deployment_name = deployment_name self_durationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationdurationduratio</pre>
	seif.auration = auration # Use AppsVlApi to interact with deployments
	<pre>self.apps_v1 = client.AppsV1Api()</pre>
	<pre>def test_steady_state(self):     ready replicant content = 0</pre>
	ready_replicas_count = 0
	# Loop for the specified duration
	<pre># Loop for the specified duration for in range(self.duration):</pre>
```
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3930
3931
3932
3933
3934
3935
3936
```

```
22
 resp = self.apps_v1.read_namespaced_deployment_status(self.deployment_name,
 self.namespace)
23
 ready_replicas = resp.status.ready_replicas or 0
24
 print(f"Ready replicas for {self.deployment_name}: {ready_replicas}")
25
26
 # Check if the number of ready replicas is at least 1
27
 if ready_replicas >= 1:
28
 ready_replicas_count += 1
29
30
 # Wait for 1 second before the next check
31
 time.sleep(1)
32
33
 # Calculate the percentage of time the deployment was ready
34
 readiness_percentage = (ready_replicas_count / self.duration) * 100
 print(f"{self.deployment_name} was ready {ready_replicas_count}/{self.duration}
35
 times.")
36
37
 # Assert that the deployment was ready 100% of the time
 assert readiness_percentage == 100, f"{self.deployment_name} readiness was {
38
 readiness_percentage}%, expected 100%."
39
40
41 def main():
 parser = argparse.ArgumentParser(description='Test front-end replica readiness')
42
 parser.add_argument('--duration', type=int, default=5, help='Duration to check the
43
 replicas')
44
 args = parser.parse_args()
45
46
 # Create a test instance and run the test
47
 test = TestFrontEndReplica('sock-shop', 'front-end', args.duration)
48
 test.test_steady_state()
49
50
51 if _____ name___ == '____main___':
52
 main()
```

## Chaos Mesh workflow manifest

```
1 apiVersion: chaos-mesh.org/vlalphal
 2 kind: Workflow
 3 metadata:
 4 name: chaos-experiment-20241127-045539
 5 spec:
 6
 entry: the-entry
 7
 templates:
 8
 #---
 9
 # entry point of whole workflow
 10
 #--
 11
 - name: the-entry
 12
 templateType: Serial
 13
 deadline: 30m45s
 14
 children:
 15
 - pre-validation-phase
 16
 - fault-injection-phase
 - post-validation-phase
 17
 18
 19
 #--
 20
 # Entry point of pre-validation-phase
 21
 22
 - name: pre-validation-phase
 23
 templateType: Serial
 24
 deadline: 10m20s
 25
 children:
 - pre-validation-parallel-workflows
 26
 27
 28
 - name: pre-validation-parallel-workflows
 29
 templateType: Parallel
 30
 deadline: 5m20s
 31
 children:
3937
 32
 - pre-unittest-carts-db-replicas
3938
 33
 - pre-unittest-front-end-replica
 34
3939
 35
 # Definitions of children of pre-validation-phase
3940
 36
 - name: pre-unittest-carts-db-replicas
 37
 templateType: Task
3941
```

3942	1	
3943	38	deadline: 5m20s
3944	39	task:
3945	40	name: pre-unittest-carts-db-replicas-container
3946	42	image: chaos-eater/k8sapi:1.0
3047	43	command: ["/bin/bash", "-c"]
2010	45	args: ["python /chaos-eater/sandbox/cycle_20241127_043136/unittest_carts-db-
3940	46	replicas_mod0.pyduration 20"]
3949	47	- name: pvc-volume
3950	48	mountPath: /chaos-eater
3951	50	- name: pvc-volume
3952	51	persistentVolumeClaim:
3953	52	claimName: pvc
3954	54	- name: pre-unittest-front-end-replica
3955	55	templateType: Task
3956	57	task:
3957	58	container:
3958	59 60	name: pre-unittest-front-end-replica-container image: chaos-eater/k8sapi:1.0
3959	61	imagePullPolicy: IfNotPresent
3960	62	command: ["/bin/bash", "-c"] args: ["python /chaos-eater/sandbox/cycle 20241127 043136/unittest front-end-
3961	05	replica_mod0.pyduration 20"]
3962	64	volumeMounts:
3963	66	mountPath: /chaos-eater
3964	67	volumes:
3065	69	<pre>- name: pvc-volume persistentVolumeClaim:</pre>
2066	70	claimName: pvc
2067	71 72	#
2060	73	# Entry point of fault-injection-phase
3900	74	#
3909	76	templateType: Serial
3970	77	deadline: 10m15s
3971	79	- fault-injection-overlapped-workflows
3972	80	
3973	81	<pre>- name: rault-injection-parallel-workflow templateType: Parallel</pre>
3974	83	deadline: 5m10s
3975	84 85	cnlldren: - fault-unittest-carts-db-replicas
3976	86	- fault-stresschaos
3977	87	- name: fault-injection-suspend-workflow
3978	89	templateType: Serial
3979	90	deadline: 5m15s children:
3980	92	- fault-injection-suspend
3981	93 94	- fault-injection-parallel-workflows
3982	94	- name: fault-injection-suspend
3983	96	templateType: Suspend
3984	97	deadline: lus
3985	99	- name: fault-injection-parallel-workflows
3986	100	templateType: Parallel deadline: 5m5s
3987	102	children:
3988	103	- fault-unittest-front-end-replica - fault-podchaos
3989	105	· · · · · · ·
3990	106	- name: fault-injection-overlapped-workflows templateType: Parallel
3991	108	deadline: 5m15s
3992	109	children:
3993	111	<pre>- rauterinjection-paratier-workflow - fault-injection-suspend-workflow</pre>
3994	112	
3005	113	# Definitions of children of pre-validation-phase
3000	1	

3996		
3997	114	# unit tests
3998	115 116	<pre>- name: fault-unittest-carts-db-replicas templateTupe: Task</pre>
3999	117	deadline: 5m10s
4000	118	task:
4004	119	container:
4001	120	image: chaos-eater/k8sapi:1.0
4002	122	imagePullPolicy: IfNotPresent
4003	123	command: ["/bin/bash", "-c"] args: ["nython /chaos-eater/sandboy/cycle 20241127 043136/unittest carts-db-
4004	121	replicas_mod0.pyduration 10"]
4005	125	volumeMounts:
4006	126	<pre>- name: pvc-volume mountPath: /chaos-eater</pre>
4007	128	volumes:
4008	129	- name: pvc-volume
4000	130	persistentVolumeClaim:
4009	132	
4010	133	- name: fault-unittest-front-end-replica
4011	134	templateType: Task deadline: 5m5c
4012	136	task:
4013	137	container:
4014	138	name: fault-unittest-front-end-replica-container
4015	139	imager Chaos-eater/Kosapi:1.0
4016	141	command: ["/bin/bash", "-c"]
4010	142	args: ["python /chaos-eater/sandbox/cycle_20241127_043136/unittest_front-end-
4017	143	volumeMounts:
4018	144	- name: pvc-volume
4019	145	mountPath: /chaos-eater
4020	140	- name: pvc-volume
4021	148	persistentVolumeClaim:
4022	149	claimName: pvc
/023	150	# fault injections
4023	152	- name: fault-stresschaos
4024	153	templateType: StressChaos
4025	154	stressChaos:
4026	156	containerNames:
4027	157	- carts-db
4028	158	selector:
4029	160	labelSelectors:
4030	161	name: carts-db
4031	163	- sock-shop
1022	164	stressors:
4032	165	cpu:
4033	167	workers: 2
4034	168	
4035	169	name. fault nedebace
4036	170	templateType: PodChaos
4037	172	deadline: 5s
4038	173	podChaos:
/030	174	mode: one
4040	176	selector:
4040	177	labelSelectors:
4041	178 179	name: Iront-end namespaces:
4042	180	- sock-shop
4043	181	value: '1'
4044	182 183	
4045	184	#
4046	185	# Entry point of post-validation phase
/0/7	180	# name: post-validation-phase
4040	188	templateType: Serial
4048	189	deadline: 10m10s
4049		

190 191	children:
192	post variation parallel worklows
193	- name: post-validation-parallel-workflows
194	templateType: Parallel
195	deadline: 5m10s
196	children:
197	- post-unittest-carts-db-replicas
198	- post-unittest-front-end-replica
200	# Definitions of children of pre-validation-phase
201	- name: post-unittest-carts-db-replicas
202	templateType: Task
203	deadline: 5m10s
204	task:
205	container:
206	name: post-unittest-carts-db-replicas-container
207	image: chaos-eater/k8sapi:1.0
208	imagePullPolicy: IfNotPresent
209	command: ["/bin/bash", "-c"]
210	args: ["python /chaos-eater/sandbox/cycle_20241127_043136/unittest_carts-d
	replicas_mod0.pyduration 10"]
211	volumeMounts:
212	- name: pvc-volume
213	mountPath: /chaos-eater
214	volumes:
215	- name: pvc-volume
216	persistentVolumeClaim:
217	claimName: pvc
218	
219	- name: post-unittest-front-end-replica
220	templateType: Task
221	deadine: smius
222	task:
225	container:
224	name: post-unittest-front-end-replica-container
223	image: cnaos-eater/k0sap1:1.0
220	IMagerullrollCY: IINOUPresent
228	args ("nuthon /chaos-pater/sandboy/cycle 20241127 043136/unittest front-e
220	replica modo pyduration 10"]
229	volumeMounts.
230	- name: hydroxics.
231	mountPath. /chaos-eater
232	volumes.
233	- name: pyc-yolume
234	persistentVolumeClaim:
235	claimName: pvc
1	

## Reconfigured 09-front-end-dep.yaml

4085

4086

	apiVersion: apps/vl kind: Deployment
3	metadata:
4	name: front-end
5	namespace: sock-shop
6	spec:
7	replicas: 2
8	selector:
9	matchLabels:
10	name: front-end
11	template:
12	metadata:
13	labels:
14	name: front-end
15	spec:
16	containers:
17	- name: front-end
18	image: weaveworksdemos/front-end:0.3.12
19	resources:
20	limits:
21	cpu: 300m
22	memory: 1000Mi
23	requests:
24	cpu: 100m
	1 2 3 3 4 4 5 6 6 7 7 8 9 9 100 11 11 21 3 14 4 15 16 6 17 7 18 8 9 20 21 22 23 24