MORPHEUS: A Persistent Enterprise Benchmark for Continual Reinforcement Learning in the Big World

Pavan Seshadri[1,†], **Ankit Kumar**[1], **Anish Chhabra**[1], **Shashank Tonpekar**[1], **Sriram Ganapathi Subramanian**[1], **Kaheer Suleman**[1], **Sam Pasupalak**[1], **Sumit Pasupalak**[1]

[1]**Skyfall AI**
[†] Correspondence to pavan.seshadri@skyfall.ai

Abstract

We introduce MORPHEUS, a persistent enterprise simulation platform for continual reinforcement learning (CRL) research. Existing reinforcement learning (RL) benchmarks are episodic, low-dimensional, and stationary by design, properties antithetical to the challenges of real-world deployed systems. Grounded in the Big World Hypothesis, MORPHEUS provides environments in which the world never resets to an initial state, objectives shift over time, and past decisions have compounding consequences. MORPHEUS comprises five enterprise simulation environments; we evaluate two in this paper, drawn from outbound logistics and inbound warehouse operations, each exhibiting structured non-stationarity through a parameterisable failure injection engine and an asynchronous configuration shift controller. Policies are initialised via supervised fine-tuning on API-collected trajectories and subsequently trained with PPO-based reinforcement learning. Rewards are computed from operational verifiers embedded in the platform: structured failure event signals, financial ledger status, and resource throughput. We define a formal benchmark specification, propose a six-metric evaluation protocol covering per-configuration reward, adaptation speed, forgetting, recovery time, stability, and performance gap relative to a configuration-specific theoretical upper bound, and establish baseline results across four algorithm families: standard RL, replay-based, regularisation-based, and latent context modelling.

1 Introduction

Standard reinforcement learning (RL) benchmarks assume a Markov decision process (MDP) with a fixed transition function, a stationary reward signal, and an episode structure that resets the world to a known initial state distribution at the end of each episode. This paradigm has produced remarkable progress (Mnih et al., 2015; Schulman et al., 2017; Haarnoja et al., 2018) but studies a class of problems that excludes the most important challenges facing deployed intelligent systems.

Real decision-making systems operate continuously, without resets. Demand distributions shift, suppliers fail, regulatory requirements evolve, and the consequences of past decisions compound into current constraints. The Big World Hypothesis (Javed & Sutton, 2024) captures this precisely: intelligence learned in small, closed environments does not transfer to large, open, and evolving worlds. Here "large" and "evolving" mean that the world's complexity exceeds what the agent can represent, making it appear non-stationary even when the underlying dynamics are fixed. Elelimy et al. (2025) formalise this argument and identify four foundations of traditional RL as antithetical to continual learning: the MDP formalism, a focus on optimal policies, expected cumulative reward as the evaluation metric, and episodic benchmarks.

In this work, our central question is: what would a benchmark look like that makes continual learning necessary? Drawing on these arguments, we identify three required properties. First, **persistence**: the world must not reset to an initial state; past decisions must compound into future dynamics.

This follows directly from the Big World Hypothesis and from Elelimy et al. (2025)'s argument that continuing tasks are the appropriate setting for continual RL. Second, **non-stationarity**: the environment must change over time in ways that render any fixed policy eventually suboptimal. This follows from the Big World Hypothesis, where an environment larger than the agent will appear to change from the agent's perspective. We additionally require that this non-stationarity be *parameterisable* as a design choice for benchmark reproducibility, so that different algorithms can be evaluated under identical conditions and results compared across papers. Third, **operational complexity**: the environment must be rich enough that no fixed optimal policy exists and adaptation requires genuine structural understanding. This follows from the Big World claim that the environment exceeds the agent's representational capacity. We ground this in enterprise operational processes where the action space, delayed consequences, and interacting domain systems collectively prevent any policy from remaining optimal across all conditions.

We present MORPHEUS, a persistent enterprise simulation platform satisfying all three requirements, with five environments (comprising of outbound order fulfilment and inbound warehouse receiving, amoung others). MORPHEUS provides five simulation environments in total and we evaluate two in this paper. Policies are initialised via supervised fine-tuning (SFT) on trajectories collected from an API-based LLM agent, then trained with PPO-based reinforcement learning. Configuration changes that implement regime shifts are triggered by an asynchronous controller running independently of the training loop.

We make the following contributions in this work.

1. We introduce MORPHEUS, a persistent enterprise simulation platform for continual RL research, based on the Big World Hypothesis and addressing the benchmark gaps identified by Elelimy et al. (2025) and Sutton et al. (2022).
2. We present a benchmark specification comprising two tasks over the `process-outbound` and `process-inbound` environments, stressing adaptation under structured drift and long-horizon credit assignment under delayed configuration shift effects.
3. We propose a six-metric evaluation protocol covering per-configuration reward, adaptation speed, forgetting, recovery time, stability, and a performance gap relative to a configuration-specific theoretical upper bound, with relative adaptation advantage and plasticity tracking as supplementary diagnostics.
4. We establish baseline results across four algorithm families, providing an empirical study of algorithm behaviour under structured non-stationarity in persistent enterprise environments.

2 Background and Related Work

Continual Reinforcement Learning. Continual RL concerns agents that must learn and adapt over an indefinitely long horizon without treating each regime as an independent problem (Khetarpal et al., 2022). Core challenges include catastrophic forgetting (Kirkpatrick et al., 2017), loss of plasticity (Dohare et al., 2024), and regime detection without task boundary signals (Chandak et al., 2020). Abel et al. (2023) distinguish continual RL from multi-task RL and meta-learning by the absence of task labels and the requirement to handle non-stationarity arising from the world itself. Elelimy et al. (2025) argue that the MDP formalism fails in this setting and propose history processes and deviation regret as replacements. MORPHEUS is aligned with both: its environments are non-Markovian and we adopt a theoretically-grounded performance gap as a primary metric.

The Big World Hypothesis and Continual RL. The Big World Hypothesis (Javed & Sutton, 2024) asserts that the real world's complexity exceeds any agent's representational capacity ($C(\mathcal{E}) \gg C(\mathcal{A})$), causing the world to *appear* non-stationary even under fixed dynamics because the agent's state representation omits causally relevant variables. We define a *regime* as a contiguous interval during which the dynamics, as observed through a given agent's representation, are approximately stationary, and a *regime shift* as a transition inducing a meaningful change in the agent's approximately optimal policy. Regime boundaries are therefore properties of the agent-environment

pair, not the environment alone. In MORPHEUS, configuration changes serve as a reproducible experimental proxy; whether they constitute true regime shifts for a given agent is itself an empirical question. The Alberta Plan (Sutton et al., 2022) operationalises the Big World Hypothesis as a twelve-step programme, where Step 6 calls for a suite of continuing, non-episodic environments analogous to OpenAI Gym. MORPHEUS addresses this gap directly.

The Big World Hypothesis implies five CRL challenges MORPHEUS is designed to expose: **reward non-stationarity** (the reward mapping shifts across configurations); **catastrophic forgetting** (adapting to a new configuration risks overwriting prior policies, measurable via cyclic configuration sequences); **task-boundary-free detection** (no boundary signal is emitted and shifts must be inferred from experience); **non-stationary credit assignment** (value functions trained under one configuration produce biased targets after a shift, compounded by delayed outcomes); and **forward transfer** (shared domain structure across environments provides a falsifiable positive-transfer hypothesis).

Existing Benchmarks. ALE (Bellemare et al., 2013), MuJoCo (Todorov et al., 2012), and OpenAI Gym (Brockman et al., 2016) are episodic by design. Procgen (Cobbe et al., 2020) introduces procedural variation but remains episodic. ContinualWorld (Wołczyk et al., 2021) uses explicit task boundaries and episodic resets. None tests the core CRL property: detecting, adapting to, and retaining competence across unlabelled regime shifts in a world that never resets. AgarCL (Mohamed et al., 2025), based on Agar.io, provides emergent non-stationarity via multi-agent interactions well suited for open-ended robustness. MORPHEUS is complementary: it provides structured, reproducible configuration shifts enabling precise analysis of adaptation speed, forgetting, and recovery. Non-stationary MDP methods (Padakandla, 2021) largely preserve episodic structure, however, the MORPHEUS setting is harder because regime shifts are unannounced and delayed reward produces biased temporal difference (TD) targets across configuration boundaries (Hung et al., 2019).

3 The MORPHEUS Platform

3.1 Architecture

Each environment in MORPHEUS is a self-contained *world plugin*: a TypeScript package exporting *Operational Descriptors* (ODs), a simulation scheduler, seed data, and documentation. An OD defines the step-by-step execution plan for a capability (consisting of the sequence of available actions, tools called at each step, and data dependencies) and is the closest analogue to a transition structure in MORPHEUS. Agents interact via a capability API, where each call triggers an OD execution, with the failure injection engine (see Appendix B) introducing typed disruptions between steps. All results, failure events, state transitions, and ledger entries are logged to persistent, queryable collections. World instances are persistent (there are no episode resets). A world created at t_0 continues to evolve via the simulation scheduler independently of agent activity. An agent that fails to act faces real consequences: inventory depletes, orders go unfulfilled, and production schedules slip.

3.2 Simulation Environments

MORPHEUS provides five simulation environments, summarised in Table 1. The current study provides the performances of algorithms on two environments for which experiments have been completed. Evaluation on others is deferred to follow-on work.

Here, **Process Inbound** simulates the inbound logistics lifecycle from supplier arrival through quality control, putaway, and inventory reconciliation. Errors at receiving propagate through downstream steps with consequences only becoming observable later. To give a concrete example: a missing supplier advance shipping notice (ASN) (i.e., the electronic manifest accompanying an inbound shipment) forces the agent to proceed with a physical count alone, introducing discrepancies that surface as inventory record errors several steps later. The agent's observation at any step does not reveal whether a prior discrepancy exists, making this a partially observable problem. **Process Outbound** simulates order fulfilment from creation through picking, dispatch, and payment settlement. The headline metric, On-Time In-Full (OTIF) delivery, is observable only at delivery, several simulated days after the dispatch decision. The agent must learn to use upstream process

Environment	World ID	Domains	Primary CRL Challenge
Inventory Optimisation	`inventory-optimizer`	WMS, ERP	Regime-dependent policy
Process Inbound†	`process-inbound`	WMS, ERP, TMS	Sequential process accuracy
Process Outbound†	`process-outbound`	WMS, ERP, TMS	Delayed reward (OTIF)
Production Planning	`production-planning`	ERP, WMS, MES	Multi-step credit assignment
Order-to-Cash	`order-to-cash`	EFM	Hidden feature extraction

Table 1: MORPHEUS environments. Environments marked † are evaluated in this paper. WMS: Warehouse Management System. ERP: Enterprise Resource Planning. TMS: Transportation Management System. MES: Manufacturing Execution System. EFM: Enterprise Financial Management.

signals as proxies for this delayed outcome. We define two tasks (Task 1 and Task 2) on these two environments. Details are in Appendix A.

3.3 Configuration Shift Control

Non-stationarity between configurations is implemented by an **asynchronous configuration shift controller**, distinct from the simulation scheduler. The controller modifies the failure preset and demand parameters at pre-defined simulation timestamps, running independently of the agent's training loop so that configuration transitions are not synchronised to gradient updates. This prevents the agent from using its own update periodicity as a proxy clock for predicting shifts. Fixed configuration timestamps ensure all algorithms face identical shift points, prioritising reproducibility over endogenous non-stationarity (where shifts would emerge from compounding operational decisions). Experimental features for endogenous non-stationarity is a compelling direction for future work.

3.4 Reward Signal

Reward is computed from three categories of operational verifier logged natively by the platform, requiring no external annotation: (1) **failure event signals**, penalising incident ticket generation with severity weights that vary by failure type; (2) **financial ledger status**, incorporating cost variance, payment timeliness, and freight efficiency; and (3) **resource throughput**, measuring processed units and fulfilled orders relative to capacity. The composite reward is a weighted sum $R(t) = w_f r_f(t) + w_l r_l(t) + w_p r_p(t)$ with default weights $w_f = 0.5$, $w_l = w_p = 0.25$. Full formulation and theoretical upper bound derivation are in Appendix C.

3.5 Policy Initialisation and Training

The richness of the MORPHEUS action space makes pure RL from random initialisation impractical. We address this with a two-stage pipeline. An API collection stage uses an LLM agent to produce observation-action-reasoning trajectories across both environments and all configurations. These trajectories then fine-tune a base language model via imitation learning, producing a shared SFT checkpoint. All RL training runs are initialised from this checkpoint, ensuring performance differences reflect continual learning behaviour rather than differences in basic operational competence.

4 Evaluation Metrics

We consider a continual RL setting (S, A, P_t, r_t) with latent configurations $\{z_t\}_{t=1}^{T}$. Cumulative reward $R_{\text{total}} = \mathbb{E}[\sum_t \gamma^{t-1} r_t(s_t, a_t)]$ alone is insufficient: a scalar sum over a non-stationary horizon conflates performance across qualitatively different operating conditions. We propose a six-metric protocol, each targeting a distinct aspect of CRL. We also report R_{total} in our experiments for completeness. Furthermore, we also report two supplementary metrics in addition to the six-metrics.

Let t_k denote the onset of configuration k and $\mathcal{T}_k$ its interval. R_k^{ub} is the *theoretical upper bound*: the maximum expected reward achievable with full knowledge of the current configuration parameters, computed analytically (Appendix C). All metrics are computed post-hoc from experimental logs. An oracle agent receiving the true label z_t is not included, with R_k^{ub} serving as a principled agent-independent reference.

1. Per-Configuration Reward (all tasks): Mean reward reported separately for each operating configuration: $J_k = \frac{1}{|\mathcal{T}_k|}\sum_{t\in\mathcal{T}_k} r_t(s_t, a_t)$.

2. Adaptation Speed (all tasks): Our headline metric. Adaptation speed is the minimum number of steps from t_k for the running average reward to reach αR_k^{ub}. We use $\alpha = 0.5$.:

$$\tau_{\text{adapt}}^{(k)} = \min\left\{\tau > 0 : \frac{1}{\tau}\sum_{j=0}^{\tau-1} r_{t_k+j} \geq \alpha R_k^{\text{ub}}\right\} \tag{1}$$

Using a running average rather than a single-step threshold makes the metric robust to reward noise. The threshold αR_k^{ub} is now anchored to the theoretical maximum for each configuration, making it comparable across algorithms and configurations. If the threshold is never reached within $\mathcal{T}_k$, $\tau_{\text{adapt}}^{(k)}$ is recorded as $|\mathcal{T}_k|$.

3. Forgetting (extended recurrent runs only): Applicable when a configuration recurs. Forgetting (F_k^{policy}) is the gap between interval mean rewards on first and second encounters, where negative values indicate positive backward transfer.

4. Recovery Time (all tasks): The number of steps for the running average reward to stabilise within $\epsilon = 0.5 \cdot R_k^{\text{ub}}$ of the upper bound (Eq. 2). Recovery time is distinct from adaptation speed, where adaptation speed measures steps to first cross the threshold, and recovery time measures steps to remain within ϵ of the upper bound.

$$\tau_{\text{rec}}^{(k)} = \min\left\{\tau > 0 : \left|\frac{1}{\tau}\sum_{j=0}^{\tau-1} r_{t_k+j} - R_k^{\text{ub}}\right| \leq \epsilon\right\} \tag{2}$$

5. Stability (all tasks): Within-configuration reward variance σ_k^2: high stability with low per-configuration reward indicates a converged but suboptimal policy; high stability with high reward indicates genuine competence.

6. Performance Gap (all tasks): $\Delta_k^{(\pi)} = R_k^{\text{ub}} - \bar{r}_k^{(\pi)}$, where $\bar{r}_k^{(\pi)}$ is mean reward over the final 20% of $\mathcal{T}_k$. A large gap with low adaptation speed indicates failure to adapt; a large gap with high adaptation speed indicates convergence to a locally optimal but globally suboptimal policy. Reported values are normalized.

The two additional supplementary metrics are defined below.

Relative Adaptation Advantage: $\text{RAA}_k^{(\pi)} = \tau_{\text{adapt,PPO}}^{(k)} - \tau_{\text{adapt}}^{(k),(\pi)}$ for $k > 1$. Positive values indicate the continual learning (CL) mechanism confers faster adaptation beyond incidental experience accumulation.

Plasticity Tracking (diagnostic): Effective rank (ER) of the policy network's penultimate layer (Dohare et al., 2024); declining rank without performance recovery signals plasticity collapse.

Aggregation and reporting. Each metric above is defined for a single configuration k and a single seed. We first average over K configurations of the schedule, written with an overbar: for a per-configuration quantity X_k, $\bar{X} = \frac{1}{K}\sum_{k=1}^{K} X_k$ (e.g. $\bar{J}_k$, $\bar{\tau}^{\text{adapt}}$, $\bar{\sigma}^2$, $\bar{\Delta}_k$, $\bar{F}_k^{\text{policy}}$; ER is averaged analogously and RAA over $k > 1$). All results tables then report the mean $\pm$ standard deviation of these configuration-averaged quantities *across seeds* (n per task as stated in section 5).

5 Experimental Protocol and Algorithm Baselines

Experimental Protocol: Configuration sequences are fixed and identical across all algorithms and seeds. Five random seeds per algorithm per task is used, with results reported as mean $\pm$ standard deviation. Upper bounds R_k^{ub} are computed analytically per configuration prior to experiments. Per-timestep logs include: reward r_t, component reward vector, configuration ID z_t, shift timestamps t_k, and effective rank of the penultimate layer.

Family	Cum. Reward	$\bar{J}_k$	$\bar{\tau}^{\text{adapt}}$	$\bar{\tau}^{\text{rec}}$	$\bar{\sigma}^2$	$\bar{\Delta}_k$	RAA	ER	$\bar{F}_k^{\text{policy}}$
PPO	$0.5040_{\pm 0.0213}$	$0.00210_{\pm 0.00007}$	$45.2_{\pm 9.5}$	$12.3_{\pm 0.0}$	$\mathbf{0.000525}_{\pm 0.00002}$	$0.9931_{\pm 0.0005}$	–	$35.77_{\pm 1.84}$	$0.00302_{\pm 0.00038}$
HER	$0.5347_{\pm 0.0486}$	$0.00223_{\pm 0.00021}$	$45.2_{\pm 11.2}$	$14.3_{\pm 0.0}$	$0.000597_{\pm 0.00002}$	$0.9931_{\pm 0.0020}$	$0.0_{\pm 9.3}$	$\mathbf{36.30}_{\pm 0.86}$	$0.00328_{\pm 0.00039}$
EWC	$\mathbf{0.5741}_{\pm 0.0278}$	$\mathbf{0.00239}_{\pm 0.00007}$	$32.5_{\pm 9.2}$	$\mathbf{1.0}_{\pm 0.0}$	$0.000623_{\pm 0.00002}$	$0.9927_{\pm 0.0010}$	$18.3_{\pm 12.0}$	$35.52_{\pm 1.87}$	$0.00334_{\pm 0.00045}$
LCM	$0.5210_{\pm 0.0207}$	$0.00217_{\pm 0.00003}$	$\mathbf{21.8}_{\pm 9.6}$	$\mathbf{1.0}_{\pm 0.0}$	$0.000573_{\pm 0.00002}$	$0.9929_{\pm 0.0000}$	$\mathbf{32.3}_{\pm 15.4}$	$35.65_{\pm 1.59}$	$0.00359_{\pm 0.00045}$

Table 2: Task 1 regime-aggregated metrics for the `process-outbound` Environment. Cumulative reward is reported as undiscounted return. Adaptation and recovery times use data-relative thresholds. **RAA** is averaged over configurations $k > 1$ relative to PPO. $\bar{\mathbf{F}}_{\mathbf{k}}^{\text{policy}}$ is policy-attributable forgetting (terminal bonus excluded). Lower is better for $\bar{\tau}^{\text{adapt}}$, $\bar{\tau}^{\text{rec}}$, $\bar{\sigma}^{\mathbf{2}}$, $\bar{\mathbf{\Delta}}_{\mathbf{k}}$, and $\bar{\mathbf{F}}_{\mathbf{k}}^{\text{policy}}$; higher is better for **Cum. reward**, $\bar{\mathbf{J}}_{\mathbf{k}}$, **RAA**, and **ER**.

Baselines: We evaluate four algorithm families, all initialised from a shared SFT checkpoint using online post-training in MORPHEUS via PPO as the base optimizer. We first collect offline trajectories in MORPHEUS using a frontier model (Gemini 3.1 pro) using the ReAct framework (Yao et al. (2022)) to collect (state, action, reasoning) traces of solving tickets in the online MORPHEUS environment across both layouts. We use these to initialize a policy (Qwen3-14B) via SFT for RL post-training in each subsequent experiment.

Specifically, we test **(1) Standard RL (PPO)** (Schulman et al., 2017): no CL mechanism; establishes the failure baseline. **(2) Replay-Based (HER)**: PPO with hindsight experience replay (Andrychowicz et al., 2017) retaining transitions across configuration boundaries; tests whether explicit retention prevents forgetting. **(3) Regularisation-Based (EWC)**: PPO with Elastic Weight Consolidation (Kirkpatrick et al., 2017); tests the stability side of the plasticity-stability trade-off. **(4) Latent Context Modelling (LCM)**: PPO with a latent context model (Rakelly et al., 2019) that infers the configuration from experience history; tests whether implicit detection enables fast adaptation. All the source code for the experiments have been open sourced (Seshadri et al., 2026).

6 Experimental Results

6.1 Process-Outbound Environment

Task 1: Dynamic Resource Allocation Under Structured Drift (see Table 2). EWC obtains the highest per-configuration reward ($\bar{J}_k = 0.00239$) and cumulative reward (0.5741), while LCM achieves the fastest adaptation ($\bar{\tau}^{\text{adapt}} = 21.8$ steps vs. 32.5 for EWC and 45.2 for PPO and HER), suggesting latent context inference is more effective at detecting structured drift. EWC and LCM both recover within 1.0 step on average; PPO and HER require 12.3 and 14.3 steps respectively. Task 1 thus exhibits a split: EWC is the best reward maximiser and LCM is the strongest adaptor.

Task 2: Scheduling Under Drift with Delayed Effects (see Table 3). Table 3 reports Task 2 results for `process-outbound`. The winner pattern differs from Task 1. HER achieves the highest cumulative reward (0.5848) and per-configuration reward ($\bar{J}_k = 0.00127$), narrowly ahead of EWC (0.5753). Adaptation speed is slower across all methods relative to Task 1, with EWC achieving the fastest adaptation ($\bar{\tau}_{\text{adapt}} = 40.5$ steps) and a positive RAA of 9.3, while LCM is the slowest adaptor ($\bar{\tau}_{\text{adapt}} = 51.1$, RAA $= -1.4$), suggesting that latent context inference is less effective under delayed reward feedback. Recovery is fast across all methods (1.0–1.2 steps). Effective rank is broadly stable across families, with EWC maintaining the highest rank (36.80). Forgetting scores are small and similar across methods, ranging from 0.00330 (PPO) to 0.00370 (HER). Task 2 in `process-outbound` therefore does not show the clean EWC/LCM split observed in Task 1: HER leads on reward, EWC leads on adaptation, and LCM offers no adaptation advantage under the delayed credit-assignment structure of this task.

6.2 Process-Inbound environment

Task 1: Dynamic Resource Allocation Under Structured Drift. Table 4 reports Task 1 results. HER obtains the highest cumulative reward (0.2474) and per-configuration reward ($\bar{J}_k = 0.00082$), while also achieving the highest effective rank (37.74). PPO is the second-best reward maximiser,

Family	Cum. Reward	$\bar{J}_k$	$\bar{\tau}^{\text{adapt}}$	$\bar{\tau}^{\text{rec}}$	$\bar{\sigma}^2$	$\bar{\Delta}_k$	RAA	ER	$\bar{F}_k^{\text{policy}}$
PPO	$0.5454_{\pm 0.0093}$	$0.00117_{\pm 0.00002}$	$49.7_{\pm 14.5}$	$1.0_{\pm 0.0}$	$0.000300_{\pm 0.00002}$	$0.9996_{\pm 0.0009}$	–	$36.04_{\pm 1.82}$	$0.00330_{\pm 0.00038}$
HER	$\mathbf{0.5848}_{\pm 0.0174}$	$\mathbf{0.00127}_{\pm 0.00006}$	$35.2_{\pm 19.0}$	$1.2_{\pm 0.4}$	$0.000300_{\pm 0.00002}$	$0.9990_{\pm 0.0014}$	$\mathbf{14.5}_{\pm 20.8}$	$35.76_{\pm 1.99}$	$0.00370_{\pm 0.00035}$
EWC	$0.5753_{\pm 0.0091}$	$0.00123_{\pm 0.00003}$	$\mathbf{40.5}_{\pm 9.3}$	$\mathbf{1.0}_{\pm 0.0}$	$\mathbf{0.000290}_{\pm 0.00002}$	$0.9991_{\pm 0.0009}$	$9.3_{\pm 10.6}$	$\mathbf{36.80}_{\pm 1.74}$	$0.00365_{\pm 0.00040}$
LCM	$0.5438_{\pm 0.0179}$	$0.00114_{\pm 0.00005}$	$51.1_{\pm 14.1}$	$1.0_{\pm 0.0}$	$0.000300_{\pm 0.00002}$	$\mathbf{0.9999}_{\pm 0.0001}$	$-1.4_{\pm 10.1}$	$35.83_{\pm 1.07}$	$0.00347_{\pm 0.00023}$

Table 3: Task 2 regime-aggregated metrics for `process-outbound` Environment.

Family	Cumulative Reward	$\bar{J}_k$	$\bar{\tau}^{\text{adapt}}$	$\bar{\tau}^{\text{rec}}$	RAA	ER	$\bar{F}_k^{\text{policy}}$
PPO	0.2366 ± 0.0176	0.00079 ± 0.00006	60.0 ± 0.0	1.0 ± 0.0	–	35.06 ± 0.71	0.00397 ± 0.00027
HER	$\mathbf{0.2474 \pm 0.0302}$	$\mathbf{0.00082 \pm 0.00010}$	60.0 ± 0.0	1.0 ± 0.0	0.0	$\mathbf{37.74 \pm 1.02}$	0.00401 ± 0.00047
EWC	0.2278 ± 0.0125	0.00076 ± 0.00004	60.0 ± 0.0	1.0 ± 0.0	0.0	35.47 ± 0.31	0.00400 ± 0.00035
LCM	0.2275 ± 0.0215	0.00076 ± 0.00007	60.0 ± 0.0	1.0 ± 0.0	0.0	35.81 ± 0.39	$\mathbf{0.00366 \pm 0.00039}$

Table 4: Task 1 regime-aggregated metrics for the `process-inbound` Environment. Stability and performance gap are omitted as near-constant low per-configuration reward renders both metrics uninformative in `process-inbound`.

with cumulative reward 0.2366 and $\bar{J}_k = 0.00079$. EWC and LCM obtain similar cumulative rewards (0.2278 and 0.2275 respectively). Adaptation speed saturates at 60.0 steps for all methods, and recovery time is 1.0 step for all methods due to near-constant low per-configuration reward across regimes; consequently, RAA is 0.0 for HER, EWC, and LCM. Forgetting is small across all methods, with LCM obtaining the lowest forgetting score (0.00366). Task 1 therefore shows a clear reward and representation advantage for HER, while adaptation and recovery metrics do not distinguish the algorithm families.

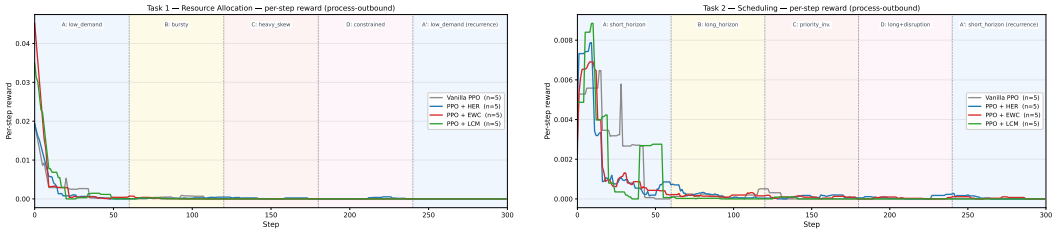
Task 2: Scheduling Under Drift with Delayed Effects. Table 5 reports Task 2 results. HER again obtains the highest cumulative reward (0.2499) and per-configuration reward ($\bar{J}_k = 0.00083$), with LCM close behind on reward (0.2440, $\bar{J}_k = 0.00081$). LCM is the clear adaptation leader, achieving the fastest adaptation speed ($\bar{\tau}_{adapt} = 51.0$) and the only positive RAA (+1.56). HER has the highest effective rank (36.87), suggesting stronger representation preservation, but adapts more slowly than PPO and LCM. EWC performs worst on cumulative reward (0.1861) and adaptation speed (59.0), although it has the lowest forgetting score (0.00315). Task 2 thus separates reward maximisation from adaptation: HER is the strongest reward maximiser, while LCM is the strongest adaptor.

7 Discussion

We make the following observations from the performance of the algorithms.

7.1 Process-Outbound Environment

All algorithms remain far below the theoretical upper bound. Mean performance gaps $\bar{\Delta}_k$ sit near 1.0 for every method on both tasks, indicating a large settled-state deficit rather than a minor tuning gap. This is not attributable to isolated episodes or reset artefacts. MORPHEUS worlds



(a) Task 1: dynamic resource allocation.

(b) Task 2: delayed-effect scheduling.

Figure 1: Per-configuration (or Per-step) reward over both tasks across circular regime shifts in process-outbound.

Family	Cumulative Reward	$\bar{J}_k$	$\bar{\tau}^{\text{adapt}}$	$\bar{\tau}^{\text{rec}}$	RAA	ER	$\bar{F}_k^{\text{policy}}$
PPO	0.2376 ± 0.0465	0.00079 ± 0.00016	52.2 ± 13.6	1.0 ± 0.0	–	34.88 ± 1.06	0.00399 ± 0.00085
HER	$\mathbf{0.2499 \pm 0.0191}$	$\mathbf{0.00083 \pm 0.00006}$	58.6 ± 2.5	1.0 ± 0.0	-8.56 ± 14.82	$\mathbf{36.87 \pm 1.07}$	0.00419 ± 0.00038
EWC	0.1861 ± 0.0807	0.00062 ± 0.00027	59.0 ± 1.7	1.0 ± 0.0	-9.11 ± 15.78	34.94 ± 0.50	$\mathbf{0.00315 \pm 0.00162}$
LCM	0.2440 ± 0.0348	0.00081 ± 0.00012	$\mathbf{51.0 \pm 7.8}$	1.0 ± 0.0	$\mathbf{+1.56 \pm 15.72}$	35.58 ± 0.84	0.00396 ± 0.00054

Table 5: Task 2 regime-aggregated metrics for the `process-inbound` Environment.

are persistent with hidden shifts and delayed consequences, so the gap reflects genuine difficulty maintaining competence as operational conditions drift.

Figure 1 shows the per-configuration reward of each task from one seed run in process-outbound. We find that the bulk of variance in each algorithm family occurs on the first regime (regime A), when there is both a low amount of tickets (and therefore, a higher chance of selecting the most impactful tasks to execute), with easier reward-wins via ticket completion in the beginning of ticket completion compared to the tail end. During the more difficult regimes, this ability falters, both along securing completion for previous tasks, as well as in selecting the most reward-maximizing tickets in a larger, more difficult set, given the environment changes up until that point. When returning to regime A at the end, the agent is not able to return to previous performance, hindered by the workload of the intermediate regimes.

No single algorithm family dominates. EWC leads on cumulative reward and per-configuration reward on Task 1, while HER leads on Task 2. LCM leads on adaptation speeds in Task 1 and lags behind in Task 2. This divergence validates the six-metric protocol: aggregate reward alone would crown EWC on Task 1 and HER on Task 2, obscuring the cross-task role reversal. The six metrics make visible that reward accumulation and adaptation speed identify different winners, and that the identity of each winner changes with the task's CRL structure.

Regime detection is hard even in controlled settings. PPO and HER generally adapt only in the first configuration and fail to adapt in later regimes, even in a controlled four-regime sequence without label signals. Per-configuration reward decays toward zero despite continued experience, indicating agents cannot capture the regime changes.

7.2 Process-Inbound Environment (additional observations to that in Process-Outbound)

Forgetting remains small across all methods. LCM has the lowest forgetting score on Task 1, while EWC has the lowest forgetting score on Task 2. However, the forgetting values are close in magnitude and should be read together with reward and effective rank. A low forgetting score without high reward does not necessarily imply strong continual-learning performance. It may instead be an artifact of uniformly low performance across the initial and recurrent configurations.

The inbound results show no evidence of plasticity collapse. Effective rank remains stable across all families, with HER maintaining the highest rank in both tasks. The main difficulty is therefore not representational collapse, but the ability to convert persistent representations into fast and high-reward adaptation under hidden configuration shifts. Overall, `process-inbound` favours HER for reward maximisation and representation preservation, while Task 2 reveals a distinct adaptation advantage for LCM.

8 Conclusion

We presented MORPHEUS, a persistent enterprise simulation platform that makes continual RL necessary. Grounded in the Big World Hypothesis, MORPHEUS provides environments that are persistent, non-stationary by design, and operationally complex enough that no fixed policy remains optimal. We defined a two-task benchmark, a six-metric evaluation protocol anchored to configuration-specific theoretical upper bounds, and four algorithm baselines covering the major CRL paradigms. Baseline results reveal large settled-state deficits and task-dependent algorithm specialisation. This demonstrates that no existing approach solves the benchmark and that multi-metric evaluation is necessary to characterise CRL behaviour. Limitations and future work are discussed in Appendix D.

References

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A Definition of Continual Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip S. Thomas. Optimizing for the Future in Non-Stationary MDPs. In *International Conference on Machine Learning (ICML)*, 2020.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2020.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of Plasticity in Deep Continual Learning. *Nature*, 632(8026): 768–774, 2024.

Esraa Elelimy, David Szepesvari, Martha White, and Michael Bowling. Rethinking the Foundations for Continual Reinforcement Learning. *arXiv preprint arXiv:2504.08161*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning (ICML)*, 2018.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing Agent Behavior over Long Time Scales by Transporting Value. *Nature communications*, 10(1):5223, 2019.

Khurram Javed and Richard S. Sutton. The Big World Hypothesis and its Ramifications for Artificial Intelligence. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards Continual Reinforcement Learning: A Review and Perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Mohamed A Mohamed, Kateryna Nekhomiazh, Vedant Vyas, Marcos M. Jose, Andrew Patterson, and Marlos C. Machado. The Cell Must Go On: Agar.io for Continual Reinforcement Learning. *arXiv preprint arXiv:2505.18347*, 2025.

Sindhu Padakandla. A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. In *International Conference on Machine Learning (ICML)*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.

Pavan Seshadri, Ankit Kumar, Anish Chhabra, Shashank Tonpekar, Sriram Ganapathi Subramanian, Kaheer Suleman, Sam Pasupalak, and Sumit Pasupalak. Morpheus evals. https://github.com/Skyfall-Research/morpheus-evals, 2026. Accessed: 2026-06-27.

Richard S. Sutton, Michael Bowling, and Patrick M. Pilarski. The Alberta Plan for AI Research. *arXiv preprint arXiv:2208.11173*, 2022.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A Physics Engine for Model-Based Control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual World: A Robotic Benchmark For Continual Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Benchmark Specification

We define two benchmark tasks, each stressing a distinct CRL failure mode. Both are continuing tasks with no episode resets. Configuration shifts are applied by the asynchronous controller during training without any signal to the agent.

A.1 Task 1: Dynamic Resource Allocation Under Structured Drift

In `process-inbound`, an agent is tasked with allocating resources across incoming workloads w_t subject to capacity C_t, with configuration effects modulating job rate λ_t and capacity ceiling C_t. In `process-outbound`, an agent distributes fulfilment capacity across K order categories subject to the capacity ceiling. Four configurations are applied in sequence (Table 6). The **primary challenge** is the adaptation-stability trade-off, where fast adaptation to bursty demand risks overwriting the low-demand policy, while excessive stability prevents timely response to change.

A.2 Task 2: Scheduling Under Drift with Delayed Effects

In `process-inbound` an agent is tasked with scheduling incoming shipments and putaway jobs across M receiving lanes subject to regime-dependent arrival distributions and quality control constraints. In `process-outbound`, an agent is tasked with scheduling outbound jobs across M processing lanes. Reward (financial component) is delayed: OTIF is observable only at delivery, lagging dispatch decisions by several simulated days.

Four configurations are applied to both Tasks (Table 6). The **primary challenge** is long-horizon credit assignment under delayed configuration effects, where the value functions calibrated for short-horizon reward produce systematically biased estimates under long-horizon orders because the temporal structure of reward accumulation has changed. Together, Tasks 1 and 2 stress complementary failure modes of continual RL within the same environment: adaptation stability and long-horizon credit assignment respectively.

Config	Task 1	Task 2	Failure preset
1	Low demand	Short-horizon orders	`light`
2	Bursty demand	Long-horizon orders	`realistic` / `moderate`
3	Heavy SKU skew	Priority inversion	`realistic`
4	Constrained capacity	Long-horizon + disruption	`aggressive` / `moderate`

Table 6: Configuration sequences for Tasks 1 and 2.

A.3 Extended Run: Recurrent Configurations

Both tasks can be extended to include a return to the initial configuration ($A \to B \to C \to D \to A$). This enables direct measurement of forgetting: performance on the return to configuration A is compared to the original encounter, with any gap constituting the forgetting score.

B The Failure Injection Engine

Non-stationarity in MORPHEUS is implemented through a built-in failure injection engine that introduces typed disruptions into OD executions. Rather than injecting random noise, the engine draws from a structured taxonomy of eleven failure types with specific operational semantics (e.g., `missing_data`, `dependency_failure`, `data_corruption`, `rate_limit`), making the non-stationarity interpretable and algorithm-designable. The engine operates at four preset failure rates: `light` (5%), `realistic` (8%), `moderate` (15%), and `aggressive` (30%). Each failure generates a structured *incident ticket*, where the reward penalises ticket generation with severity weights (see Section 3.4 and Appendix C), directly linking operational failures to agent incentives.

Crucially, different presets constitute qualitatively different operating regimes, not merely different difficulty levels.

C Reward Formulation and Theoretical Upper Bound Derivation

C.1 Composite Reward

The composite reward is:

$$R(t) = w_f \cdot r_f(t) + w_l \cdot r_l(t) + w_p \cdot r_p(t) \tag{3}$$

where r_f is the failure event signal, r_l is the financial ledger signal, and r_p is the resource throughput signal.

Failure event signal. Let $\mathcal{T}_t$ denote the set of incident tickets generated at timestep t. Each ticket carries a severity weight $s(\tau)$ depending on its failure type:

$$r_f(t) = -\sum_{\tau \in \mathcal{T}_t} s(\tau) \tag{4}$$

Severity weights: `permission_denied`, `dependency_failure`: $s = 1.0$; `data_corruption`, `missing_data`, `invalid_state`: $s = 0.6$–0.8; `rate_limit`, `format_change`, `partial_data`: $s = 0.3$–0.4; `stale_data`, `duplicate_data`, `timing_issue`: $s = 0.1$–0.2.

Financial ledger signal.

$$r_l(t) = 1 - \frac{\text{actual_cost}(t)}{\text{planned_cost}(t)} \tag{5}$$

clipped to $[-1, 1]$, incorporating cost variance, payment timeliness, and freight efficiency as logged by the platform ledger.

Resource throughput signal.

$$r_p(t) = \frac{\text{units_processed}(t)}{\text{capacity}(t)} \tag{6}$$

clipped to $[0, 1]$.

Default weights: $w_f = 0.5$, $w_l = 0.25$, $w_p = 0.25$. These are research variables; researchers are encouraged to treat them as such.

C.2 Theoretical Upper Bound Derivation

The theoretical upper bound R_k^{ub} for configuration k is computed as the maximum expected reward achievable by a policy with perfect knowledge of the configuration parameters, under the assumption of zero failures and maximum resource utilisation:

$$R_k^{\text{ub}} = w_f \cdot 0 + w_l \cdot r_l^{\max} + w_p \cdot 1 \tag{7}$$

The failure signal contributes zero under the upper bound assumption (no tickets generated). The financial signal $r_l^{\max}$ is derived from the minimum achievable cost ratio under the configuration's demand parameters (planned cost equal to actual cost, giving $r_l^{\max} = 1$). The throughput signal is clipped at 1 by definition. This gives:

$$R_k^{\text{ub}} = w_l + w_p = 0.25 + 0.25 = 0.50 \tag{8}$$

under the default weight setting. The upper bound is therefore identical across configurations under the default weights, with configuration difficulty reflected in how far agents fall below it. Researchers using alternative weight vectors should recompute R_k^{ub} accordingly. When $w_l \cdot r_l^{\max} < 1$ due to structural cost constraints in a given configuration (e.g., freight costs that cannot be reduced below planned cost), the configuration-specific $r_l^{\max}$ should be used instead of 1.

D Limitations and Future Work

There are four important limitations of this work. 1) **Environment scope.** Evaluation covers two of five environments, with experiments on `inventory-optimizer`, `production-planning`, and `order-to-cash` deferred to future work. 2) **Upper bound tightness.** R_k^{ub} assumes zero failures and perfect utilisation. In high-failure configurations it is an optimistic ceiling rather than an attainable target. 3) **Endogenous non-stationarity.** The controller implements externally triggered shifts. A study of endogenous regime changes driven by compounding decisions are deferred to future work. 4) **Reward grounding.** Component weights are research variables, not validated industry objectives. Future work will evaluate the full suite, add an oracle baseline, establish a delayed-reward correction for zero-shot generalisation in `process-outbound`, and explore endogenous non-stationarity.