DevOps S25-5

- Jonathan Bezeau
- Anzhelika Kostyuk
- Arsh Chauhan
- Jindo Kim

Feasibility Analysis of Cloud-Native Unity CI/CD: Cost, Security, and Flexibility

An analysis based on 8 weeks of Unity CI/CD pipeline activity, evaluating how cloud-native architecture could reduce costs while enhancing security and flexibility.

Legacy Setup: Unified CI/CD on Persistent VMs

Current Architecture: Coupled Logic and Execution in VMs

Our Unity CI/CD pipeline has traditionally centralized both orchestration logic and high-resource build tasks within persistent virtual machines.

- Workflow control (orchestration logic)
- All compute-intensive stages (e.g., test and build)

Baseline Analysis

Analysis Parameters and Pipeline Activity Overview

Analysis Parameters

- **Total Analysis Period:** 56 days
  - May 5th - June 29
- **Actual Usage Days:** 37 days

Daily Cost

- USD $11.04/day per VM (Standard D8ads v5 VMs)
- Excludes disk and Public IP costs

Project Breakdown

Metric	**Respiratory Therapy**	**Tattoo & Piercing Studio**
CI Jobs during the period	195	660
CD Jobs during the period	52	114
Average Jobs per Active Day	**6.85**	**20.92**
Estimated Daily Runtime	1h 41m 18s (**6080s/day**)	4h 18m 45s (**15,524s/day**)

VM Cost Optimization: Deallocation Strategy

Start/Deallocate Strategy

- VMs **start only when jobs are triggered**
- Automatically **deallocate when idle**
- Cost aligns with **actual usage**, not total calendar time

Estimated Cost: Full-Time VM vs Actual Usage

Scenario	Cost Formula	Total Cost
Always-On VM (24/7 for 56 days)	$11.04 × 56 days	**USD $618.24**
Deallocated VM (based on job activity)	$3.908 × 37 days	**USD $144.60**

Average daily cost of USD $3.908 calculated from job frequencies across RT and TPI projects.

~76.6% cost reduction just by changing how VM uptime is managed.

This cost estimate is based on running two WebGL builds per job. In reality, each job only runs one WebGL build. The extra time was added to safely cover testing and VM startup/shutdown time. So the actual cost will likely be even lower.

Cloud-Native Optimization: Azure Batch + Spot VMs

On-Demand Execution Model

- Jobs run in ephemeral Spot VM containers
- Resources exist only when needed
- Kubernetes option under testing (not yet validated)

Azure Batch can provides cloud-native execution flexibility without requiring persistent infrastructure.

Compared To	Reduction vs Batch
Always-On VM	**~94.88%**
Deallocated VM	**~78.11%**

VM Strategy Cost Comparison

Scenario	Cost Formula	Total Cost
Always-On VM	$11.04 × 56 days	**USD $618.24**
Deallocated VM (based on job activity)	$3.908 × 37 days	**USD $144.60**
Azure Batch + Spot VMs (based on job activity)	$0.855 USD x 37 days	**USD $31.635**

Batch Execution Summary: Cost and Runtime for 100 Parallel Jobs

Total Cost: 4.62 USD, 37 hours 36 minutes usage, 100 runs of RT and TPI WebGL builds in parallel — more than 6 times the typical daily VM usage.

2025-07-02	unitybatchci / rt-webgl-100-lowpriority-a	Batch account	ca central	batch-test		$0.78
2025-07-03	unitybatchci / rt-webgl-100-lowpriority-a	Batch account	ca central	batch-test		$0.53
2025-07-02	unitybatchci / rt-webgl-100-lowpriority-b	Batch account	ca central	batch-test		$0.68
2025-07-03	unitybatchci / rt-webgl-100-lowpriority-b	Batch account	ca central	batch-test		$0.47
2025-07-07	unitybatchci / tpi-webgl-100-lowpriority-a	Batch account	ca central	batch-test		$0.51
2025-07-08	unitybatchci / tpi-webgl-100-lowpriority-a	Batch account	ca central	batch-test		$0.57
2025-07-07	unitybatchci / tpi-webgl-100-lowpriority-b	Batch account	ca central	batch-test		$0.53
2025-07-08	unitybatchci / tpi-webgl-100-lowpriority-b	Batch account	ca central	batch-test		$0.55

Quality and Security Enhancements

🔒 Isolated Build Environments & Parallel Execution

- In the past, multiple jobs ran in the same environment, which could cause interference.
- Now, each job runs in a **clean, separate environment**, so they don't affect each other.
- **Tests and WebGL builds can run in parallel**, in isolated environments—this also helps reduce total execution time.

✅ More Reliable Results

- Builds and tests now run under **the same clean conditions every time**, so the results are more accurate and consistent.
- Since each test runs in a **fresh environment**, it isn't affected by previous jobs or leftover data.

🔐 Stronger Security

- Pipeline control servers like Jenkins **no longer need direct access to project files**.
- Tests and builds run in **secure, isolated containers** that can't be accessed from outside.
- Additional **high-security settings** can be applied as needed.

Flexible CI/CD Architecture Options

Option 1: Cloud-Native Serverless CI/CD

- Example: Azure Batch + Spot VMs, or AKS (Kubernetes)
- No always-on VM or Jenkins controller
- Jobs triggered via REST or job scheduler (e.g., Azure Batch, k8s cronjob)
- ✅ Cost-optimized (e.g., spot pricing)
- ⚠️ High operational complexity (infra-as-code, logging, orchestration)

Option 2: Always-On Jenkins with Deallocated Executors

- Jenkins controller stays on (small VM, low cost)
- Executors run on-demand (VMs or containers shut down after job)
- ✅ Balanced: lower cost with some elasticity
- ⚠️ Requires executor management

Option 3: Full Jenkins-on-Cloud with Cloud Executors

Jenkins controller must be self-hosted in the cloud

Executors run on cloud VMs or containers (manual scale-up/down or automation needed)

⚠️ High operational complexity, especially managing agent lifecycle

Each option offers a different balance of quality, efficiency, and cost, allowing teams to choose the most suitable model based on project size and goals.