Accelerated Kubernetes Adoption via ECS to EKS Automation

An automated migration framework to move 100+ applications from Amazon ECS to Amazon EKS, enabling scalability and reliability for a global retail leader and reducing manual effort by 70%.

Overview

A global fashion retail group needed to migrate over 100 applications from Amazon ECS to Amazon EKS to achieve greater scalability and peak-season readiness, but was bottlenecked by manual configuration conversion.

- Engineered an automated ECS-to-EKS migration framework using a Python script orchestrated by two Jenkins pipelines for configuration generation and validation.
- Achieved a 70% reduction in manual effort by automatically extracting scattered ECS data (tasks, services, etc.) via AWS APIs and generating standardized Helm charts and ArgoCD values files.
- Accelerated Kubernetes adoption by up to 3x and ensured high accuracy through an automated verification job that compared generated Kubernetes configurations against live ECS data.
- Outcome: Enabled faster global e-commerce rollouts, significant cost savings, and near-zero post-migration issues.

Client Profile

Based in Japan, the client is a leading retail group with a strong multi-brand presence across Asia, Europe, and North America.

Challenges: Manual Conversion and Migration Bottleneck

The sheer volume of applications combined with the complexity of manual configuration translation created critical challenges

- **Massive Bottleneck:** Migrating over 100 applications was a highly repetitive, error-prone, and manual process, causing significant delays and high labor costs.

- **Configuration Complexity:** Manually converting existing ECS configurations (Task Definitions, Service details, CloudWatch events) into Helm charts and region-specific values files was time-consuming and inconsistent.
- **Risk of Disruption:** A critical need for thorough verification against live ECS services was necessary to avoid operational disruptions and post-migration errors.
- **Slow Adoption:** The migration pace was limiting the client's ability to fully leverage Kubernetes for global rollouts and scalability.

QBurst Solution: Python-Driven Automation Framework

QBurst developed a unique, automated framework that utilizes a custom Python script orchestrated by two dedicated Jenkins pipelines to manage the end-to-end ECS-to-EKS conversion and validation process.

- **Migration Job (Configuration Generation):** The first Jenkins pipeline executed a Python script which used AWS APIs to extract all necessary configuration details from the live ECS environment. This intelligent engine automatically generated accurate, standardized Helm charts and region-specific values files, consolidating them into a ready-to-use zip package.
- **Verification Job (Automated Validation):** A second Jenkins pipeline ran an automated validation script against the generated files. This script simulated the EKS environment and performed necessary checks (linting, sanity checks) before comparing the output directly with the live ECS service data, ensuring accuracy and consistency.
- **Decoupled Workflow:** By automatically generating and validating the configurations, the solution empowered DevOps teams to integrate the files seamlessly into their Git repositories (ArgoCD deployment model) without manual configuration mapping.

Technical Highlights

The framework's power lies in its automated intelligence and process control:

- **Intelligent Python Engine:** Custom scripting drove the entire process, translating scattered ECS-specific data into EKS-compatible Kubernetes manifest formats.
- **CI/CD Orchestration:** Jenkins pipelines provided the control, sequencing, and logging for the automated generation and validation stages.
- **Configuration as Code:** Automatically generated Helm charts and ArgoCD values files standardized deployments and simplified EKS management.
- **Automated Validation:** Self-testing features and the crucial step of comparing generated configurations with live AWS service data ensured near-zero post-migration issues.

Impact

- **70% Reduction in Manual Effort:** Eliminated repetitive configuration work, freeing DevOps resources for strategic EKS operations and development.
- **Accelerated Adoption:** Enabled up to 3x faster Kubernetes adoption, removing the migration bottleneck of over 100+ applications.
- **Increased Reliability:** Automated validation and error-free generation ensured high consistency and led to near-zero post-migration issues.
- **Peak-Season Readiness:** Unlocked the scalability and resilience of Amazon EKS, positioning the client's e-commerce platforms for seamless global rollouts and successful peak-season operations.