QBurst | A High AI-Q™ Company

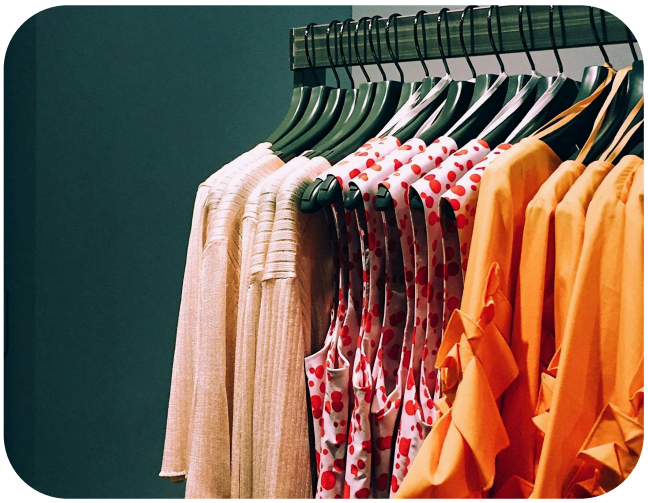
CI/CD Automation: Empowering Self-Service Microservice Deployment

Automating the deployment of over 40 microservices on AWS ECS for a leading fashion retailer, eliminating DevOps dependency and accelerating release time by 60% through a custom Jenkins pipeline.

Overview

A prominent fashion retailer, operating over 40 microservice platforms on AWS ECS, needed to resolve bottlenecks caused by manual deployments managed through support tickets and complex infrastructure code approvals.

- Developed a custom, self-service CI/CD solution using Jenkins that leverages Amazon ECR image tagging capabilities and Groovy scripting.
- Empowered application teams to independently manage their own deployments and rollbacks with zero DevOps intervention or infrastructure code (Terraform) modifications.
- Achieved a 60% reduction in deployment time by effectively skipping the time-consuming Terraform code update and approval steps.



Client

A major player in the fashion retail and e-commerce sector, the client manages several well-known brands. Their large, diverse infrastructure supports high-volume, global operations across multiple geographies.

Challenges: Manual Delays and Bottlenecks

The client's release process was slow and resource-intensive due to specific operational friction points

- **DevOps Dependency:** Developers were required to raise support tickets for every deployment, placing a significant and time-consuming operational burden on the small DevOps team.
- **Infrastructure Access Restrictions:** Application teams lacked the necessary infrastructure access to perform their own releases, creating a deployment bottleneck.

- **Slow Terraform Process:** Changes to the infrastructure (managed by Terraform) required multiple levels of approval, adding considerable complexity and time to the overall release cycle.
- **High Operational Friction:** The continuous manual intervention required by the DevOps team hindered scalability and operational agility.

QBurst Solution: Custom ECR-based CI/CD Pipeline

Our solution centered on implementing a custom, CI/CD-driven approach that utilized the client's existing AWS and Jenkins setup to facilitate self-service deployments without requiring infrastructure code (Terraform) changes.

The core mechanism was built around Amazon ECR's image tagging feature and a Jenkins pipeline.

- **Image Tagging Strategy:** A static tag was assigned to each application within the Amazon ECS task definition. Application container images are safely stored in ECR, which supports multiple tags per image.
- **Self-Service Deployment Pipeline:** We created a dedicated Jenkins job (built with Groovy/Shell scripting) that allowed application teams to select their desired image version from a list of available ECR tags.
- **Zero-Downtime Rollout:** Upon selection, the pipeline simply re-tagged the chosen image with the static tag referenced in the ECS definition and initiated an ECS service restart. This triggered an AWS rolling update, ensuring a smooth, zero-downtime transition to the new version.
- **Security & Control:** Jenkins job-level access restrictions were implemented to ensure application teams could only manage their own services, maintaining strict security and infrastructure integrity.

Technical Highlights

The solution focused on simplicity, security, and integration with existing tools.

- **Jenkins Pipeline (Groovy/Shell):** Custom logic to list ECR image tags, re-tag the selected image, and trigger the ECS service update.

- **AWS ECS & ECR:** Leveraged ECR's multi-tagging capability and ECS rolling updates for consistent, reliable deployments.
- **Decoupling from Terraform:** The strategy bypasses the need to update and approve infrastructure-as-code files for application releases.
- **Built-in Rollback:** The ability to select any previous ECR tag enables quick and easy rollback mechanisms managed directly by the application teams.

Impact

The streamlined CI/CD solution delivered significant improvements in efficiency and autonomy:

- **60% Reduction in Release Time:** Completely skipping the Terraform code update and multi-level approval process drastically reduced deployment lead time.
- **Eliminated DevOps Dependency:** Application teams gained the ability to manage their own releases and rollbacks, freeing up the DevOps team for strategic, high-value tasks.
- **Standardization by Design:** The automated Jenkins pipeline enforced a consistent, reliable deployment and rollback methodology across all 40+ microservices.
- **Enhanced Agility:** Application restarts, releases, and rollbacks are now simpler, faster, and easier to manage, accelerating the entire development lifecycle.