FEATURE STORE SUMMIT 2025

From EC2 to K8s: Zalando's Journey to Large-Scale, Real-Time Feature Serving.

Morteza Ghasempour, Senior Platform Engineer, Zalando SE

Organized by HOPSWORKS

Zalando Central Feature Store

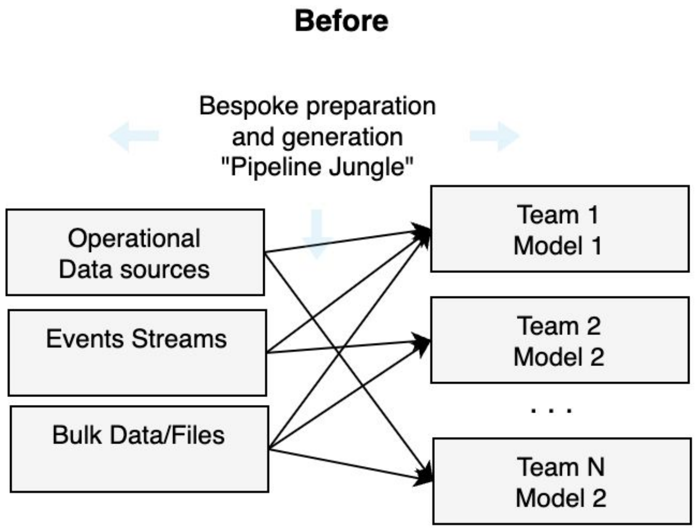
FEATURE STORE SUMMIT 2025

Organized by HOPSWORKS

Zalando Central Feature Store

Why Centralized?

- **Centralized Platform**, replacing siloed data infra.
- **Consistency** across training and production
- **Collaboration** Across Teams
- **Feature Versioning** and Governance
- **Feature Discovery**
- **Real-Time** and Batch Serving
- **Faster, more scalable** model development



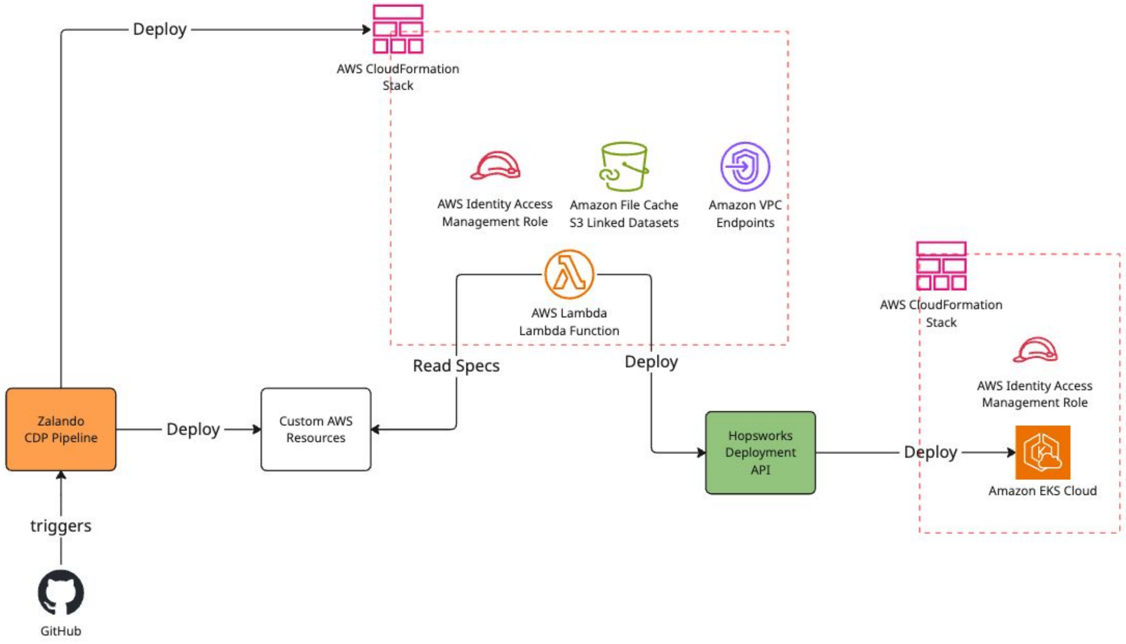
Operational Requirements

- **SLOs:**
  - **Low latency** online store (**< 10ms** for some use-cases) ✅
  - **High availability** (99.99%)
    - Replicated data across AZs ✅
    - Reduced blast radius for projects
      - Supports multi cluster setup for project isolation ✅
      - Federated FSs ✅
- **Auto scalability** of Database and event Systems
  - Database:
    - RonDB Rest API Server (RDRS)✅
    - MySQL✅
  - Event Bus:
    - Kafka (❌ in v3.9, limited to 3 brokers per cluster)
- **Compatibility** with Zalando Networking configuration and Tech stack ✅
- Comprehensive **Managed Clusters API** to Implement **IaC** ✅
- **High Feature Freshness**
  - Higher priority on updating Online Store than Offline Store ✅
- **Integrated Vector Database**✅
- Inbuilt feature **monitoring and versioning** ✅

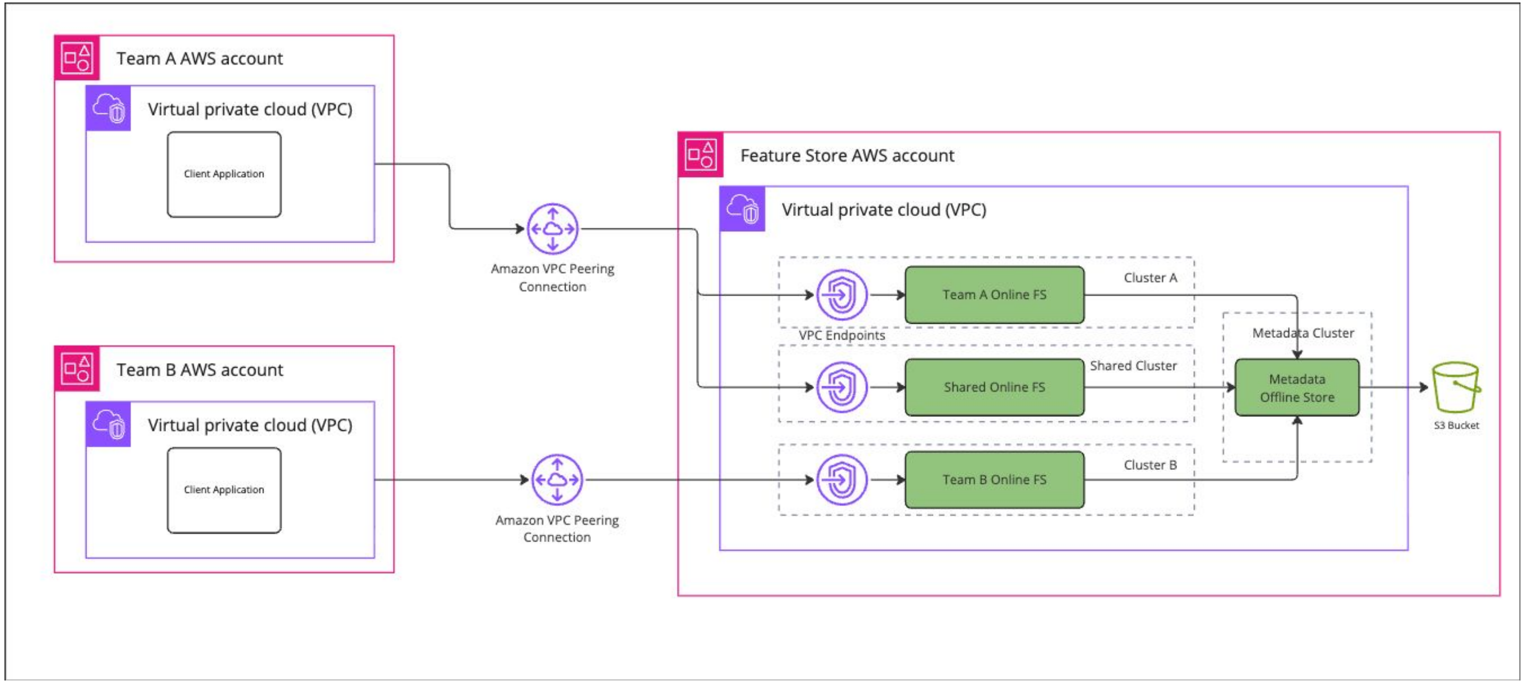
Zalando Central Feature Store

IaC Deployment pipeline

- **Consistency** and Reproducibility
- **Version Control** and Auditability
- **Scalability**
- Improved **Security** and Compliance
- **Collaboration** and Knowledge Sharing



Federated Online Feature Store



Feature Store v3.9 (EC2)

FEATURE STORE SUMMIT 2025

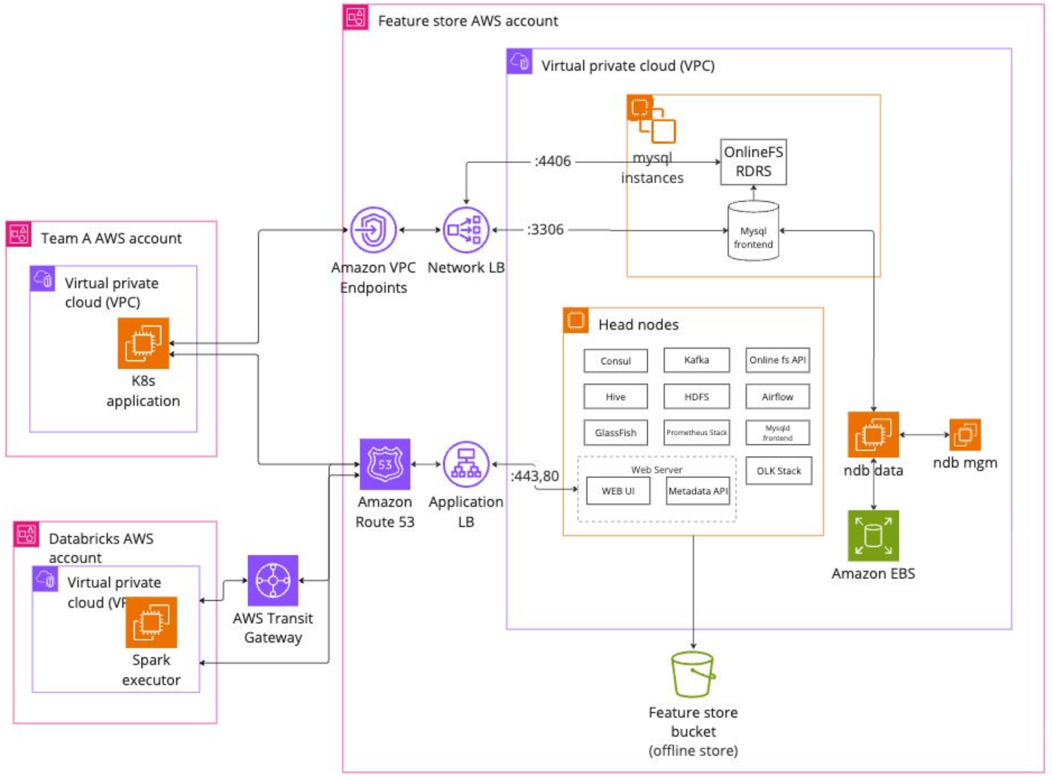
Organized by HOPSWORKS

AWS EC2 Deployment

Pros

- Instance Level Service Isolation for DB nodes
- Replicated data across availability zones

Simplified Architecture Diagram



AWS EC2 Deployment

Cons (issues)

- **Availability and reliability**
  - Resource Contention and "**Noisy Neighbor**" Syndrome
    - Numerous processes lacking resource isolation on a single kernel/OS instance Often causing **OOM**, triggering switch to the Secondary node.
- **Redundancy**
  - Slow **Blue Green** Deployment. strategy instead of **rolling update**
  - **Not** all the services where could **failover** to the Secondary Nodes.
  - **Sluggish failover**:
    - Often required manual intervention such as rebooting the stuck master node

EC2 Deployment (v3.9)

Administration Issues

- **No easy** support for service **auto-scaling** of services
- **High Possibility** of **Configuration Drift**
- **Linux Administration overhead** for the team
- **Storage Management** and Data Persistence
    - **EBS Storages** on EC2 instances
        - In-place resizing of the RonDB nodes and resizing the EBS Storages were faster than triggering a cluster scale up using cluster management api, or webui.
- **Unfamiliar non-k8s Stack**

Feature Store v4.5 on EKS

FEATURE STORE SUMMIT 2025

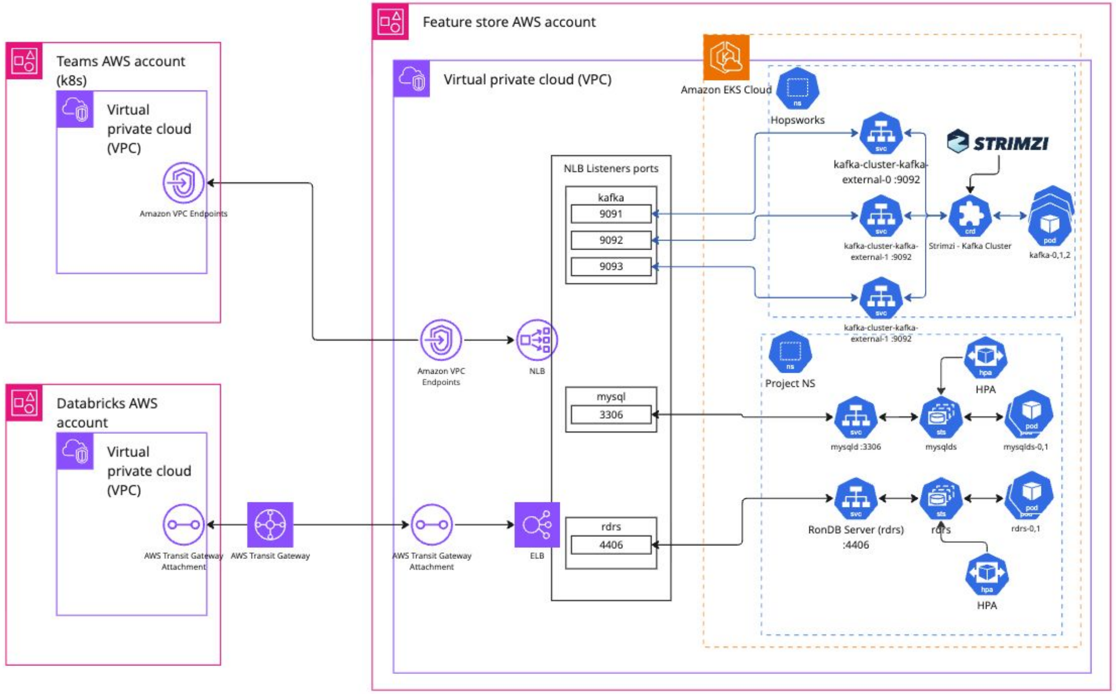
Organized by HOPSWORKS

Simplified Architecture Diagram

AWS EKS Deployment

Fixes the reliability and administration issues with **EC2** setup with:

- **Self-Healing** Services using k8s
- **Auto-Scaling** using HPA and Strimzi
- **Less risk of Configuration drift**
- **Container Level Isolation** of Services
- **Dedicated Node Groups** per projects.
  - **Easier resource allocation** administration
  - **Faster updates and rollout**
- Familiarity of the team with **K8s stacks**



Auto-Scaling

For Database and Event bus components

- via K8s HPA for
  - RonDB Rest API Server (RDRS)
    - Rdrs endpoint response time
    - Requests per seconds per Pods
  - MySQL Server instances
    - CPU usage
  - Arrow Flight
    - Request Queue size and time
- Via Strimzi Kafka Operator for Apache Kafka (not implemented yet)
  - Cruise Control
    - Automatic rebalancing

Questions?

Thank You!
Please feel free to ask.