Unveiling the Cloud Stack Behind the Crypto Exchange

30. Oktober 2025

Bitstamp

Alen Krmelj
Cloud Infrastructure Team Lead

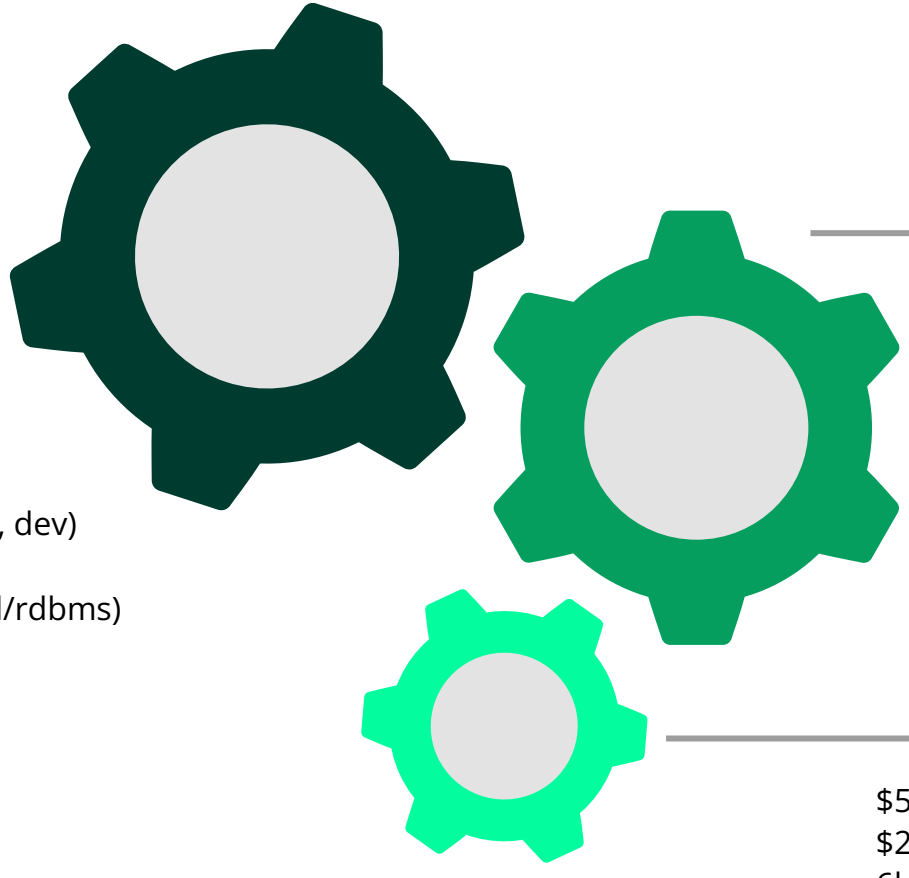
Introduction

01

Introduction

AWS Footprint

2 AWS regions (3 AZ/region)
~40 AWS accounts
~110 Applications
2 VPCs (prod)
~150 subnets
3 Environments (prod, staging, dev)
~2k EC2/ECS instances
~50 databases (in-mem/no-sql/rdbms)



People

Medium-to-large size
200+ Engineers
15 Cloud Ops
3 Cloud Infra Ops

Outcome

$500mil daily trading volume (avg)
$2.1bil daily trading volume (peak)
6k rps on API endpoint (avg)
61k rps on API endpoint (peak)

Organisational Structure & Governance

02

Organisational Structure & Governance

AWS Organization Setup

⚙️ Setup & Automation

- Use **Multi-account** setup from the start
- Automate account creation with **CloudFormation/Terraform** (StepFunctions/EventBridge)
- Request an **AWS Sandbox Org** for safe automation testing

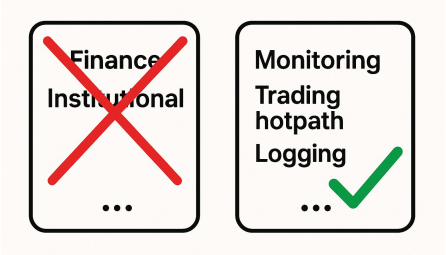
🛡️ Governance & Control

- Apply **Consolidated Billing & Spend** across accounts
- Use **SCPs** to:
  - **Disable services** not allowed to be used by developers
  - **Deny non-compliant configurations** (e.g., unencrypted EBS, S3 without versioning)
- **AWS Config** alerting for other non-critical violations

📌 Best Practices

- Organize accounts by **domains**, not by org structure

Examples of domains:



Organisational Structure & Governance

AWS Accounts

⚙️ Account Setup & IaC

- Keep development **IaC**, separate **Scratch account** (AWS Innovation Sandbox)
- **StackSets** deploys of IAM roles, policies, and KMS keys
- Deploy **baseline IAM roles & policies** for every account to enable CI/CD pipelines

🔒 Access & Security

- Enforce **SSO only** — **no IAM users** (compliance issues)
- Allow **SSH access only via Session Manager** (supports Ansible)
- Provide **on-demand, temporary production access** for developers on-call

💰 Monitoring & Billing

- Enable **Forecasting** of spend (diff is important!)
- Report **billing differences** to domain account owners

Networking & Infrastructure

03

Networking & Infrastructure

Networking Strategy

🌐 VPC & Subnets

- Use **One VPC** with multiple subnets, managed via **IaC**
- Place **firewalls/IPS** in dedicated subnets
- **VPC & subnets** in a **Networking Account** → share via **RAM**
- Plan **CIDRs** for the maximum number of accounts

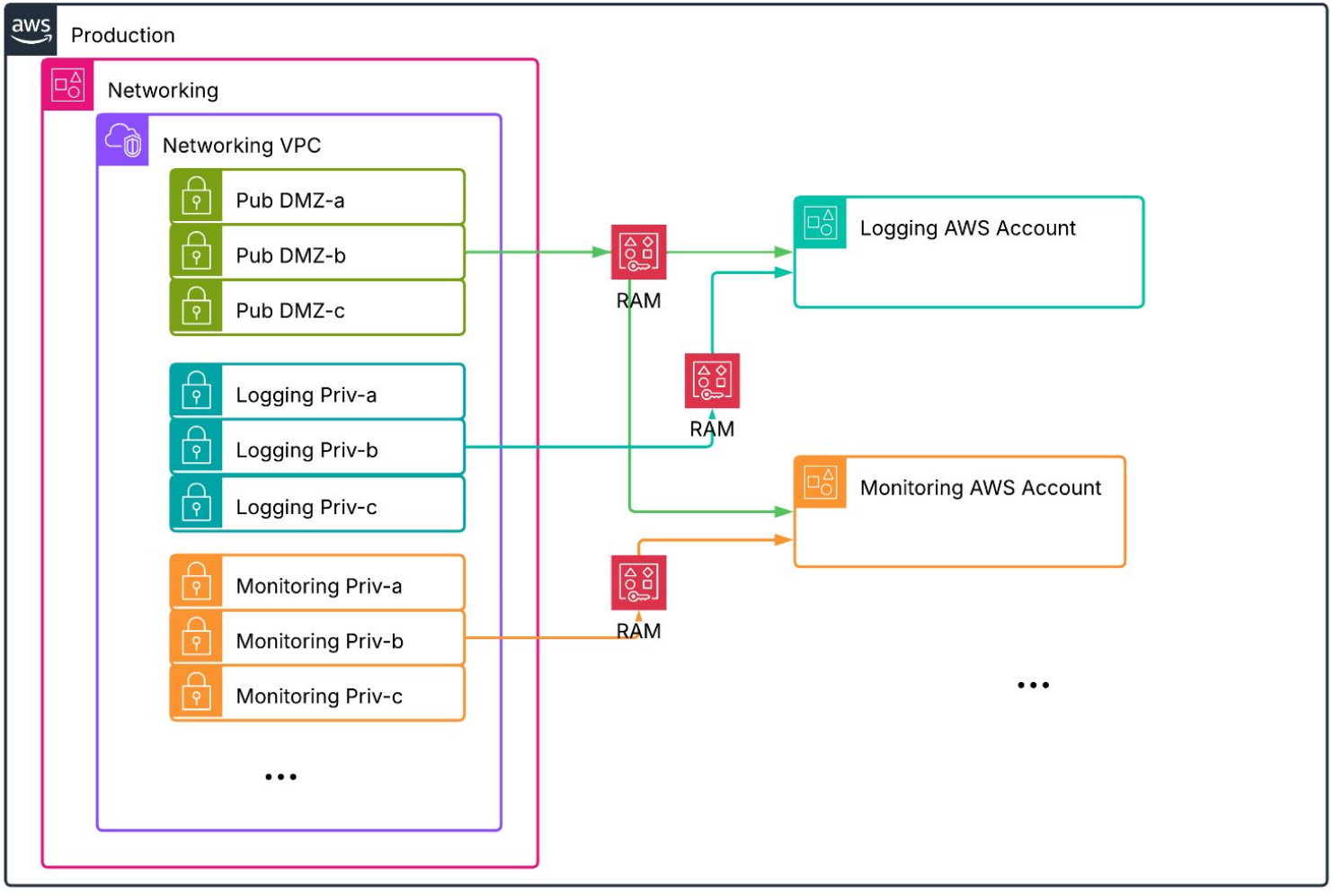
🔒 Isolation & Security

- Keep **infrastructure and networking separate** from the start
- *Optional:* Use **staging & dev isolation** (block egress, allow only via DNS/IP whitelist)

⚡ Best Practices

- Enforce consistent **IaC** network management

Networking & Infrastructure



Networking & Infrastructure

Logging infrastructure

- Centralize **CloudTrail & VPC flow logs** → Panther + Panther AI
- Stream app logs via **Vector** → Kafka → Data Lake
  - Store in **S3 (long term)**
  - Index in **OpenSearch + Kibana (short term)**

Monitoring and Tracing

- Use **CloudWatch** for infrastructure metrics (EC2, Load Balancer, ECS, credits)
- Deploy **Telegraf agents** near apps → forward to **Victoria Metrics database**
- Visualize with **Grafana** (CloudWatch + Telegraf metrics)
- Add **Zabbix** for secondary monitoring & cross-checks
- Collect **Open Telemetry + eBPF** (coroot/skywalking) data → **Jaeger collector** → **Jaeger UI (short term)**
- Build automated **event-driven actions** from alerts → Self healing infrastructure (RH Automation Controller)

Networking & Infrastructure

Compute infrastructure

- Use **Placement Groups** for latency-sensitive workloads (shared via RAM)
- Prefer **ECS over EKS** for workloads
- Prohibit use of **burst instance types**

Automation & Microservices

- Manage deployments with **Jenkins (OIDC)** → migrating to **GitHub Actions**
- Standardize on **MSK** as the **main message bus**
- Run **Microservices on ECS** (Need to have good reason to consider EKS)
- Adopt our own **BAF CDKTF framework** for secure, automated microservices deployments

Networking & Infrastructure

Database Infrastructure

🗄️ Database Engine Choices

- Prefer **Aurora over RDS** (lower replication latencies)
- Separate **databases by workload** (latency-sensitive vs non-latency-sensitive)

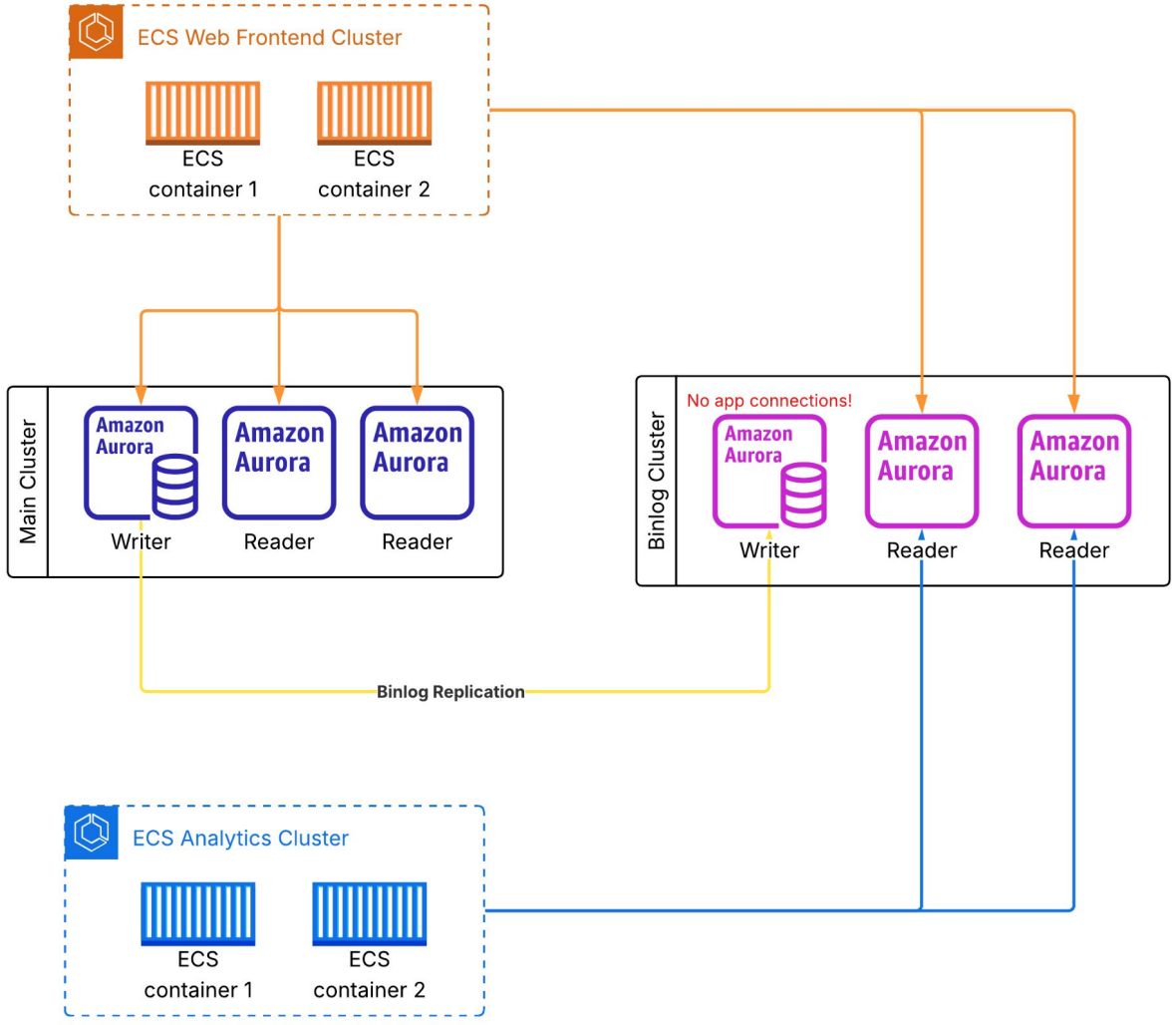
⚡ Caching Strategy

- Migrate **Redis** → **Valkey** (faster parallel execution)
- Decommission **ElastiCache Memcache** → use **Mcrouter only when necessary**

📊 Replication & Analytics

- Utilize **binlog replicas** (Aurora → Aurora cluster to cluster replication with native binlog)
- Run **heavy queries & analytics** on **binlog clusters**

Networking & Infrastructure



Networking & Infrastructure

CI/CD

⚙️ Build & Deployment

- Use **Image Builder** for EC2 metal instances (latency-sensitive workloads)

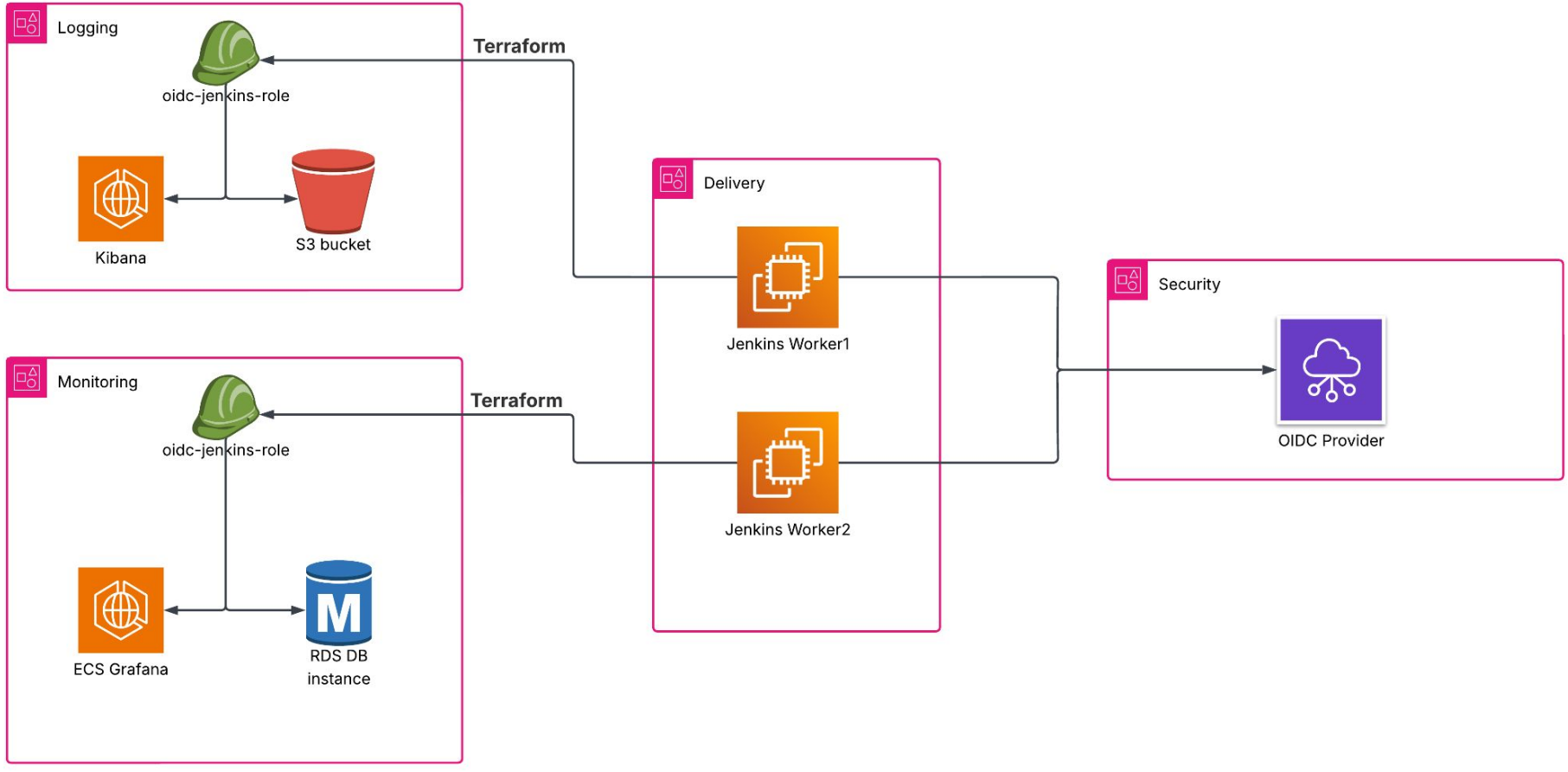
🔒 Authentication & Security

- Apply **OIDC authentication** with assume-role for cross-account deployments

📦 Package Management

- Store packages in **AWS CodeArtifact** (Decommissioned legacy Nexus)
- Store images in **ECR**

Networking & Infrastructure



Security & Compliance

04

Security & Compliance

Security strategy

🔒 Access & Isolation

- **Block all** by default → whitelist only what's needed
- **Isolate stacks** with blocked egress

📋 Compliance & Monitoring

- Use **AWS Config** for production compliance monitoring
- Use **CDN** for masking real infrastructure and DDOS protection

🛡️ Testing & Protection

- Run automated **vulnerability scans** with **Nessus** and other tools
- Perform regular **pentesting** (must schedule with AWS)

Thank You