MODUSBOX

Mojaloop - Recommended Architectural Practice for Infrastructure Deployment

An Open Source Toolset

Your Presenters

Steve Bristow

Infrastructure Engineer

MODUSBOX

- Global Network Engineering with UK Ministry of Defence
- 10yrs Banking experience at Barclays, Citi, delivering virtualization.
- 3yrs Pre-Sales Solution Architecture
- 5yrs Building On-Prem and Hybrid Clouds
- Believes technology should change the whole world.
- Loves repairing Synthesizers. Parent of 3, (only one of whom likes Synthesizers.)
- Rumoured to have Private Cloud in garage.

Jose Moreno

FastTrack Engineer

Microsoft

- 6-year experience in one of the largest civil data centers in Europe
- After that 10 years working in Cisco around many different data center technologies
- Since 2016 focusing on public cloud in Microsoft
- Married, father of 2, living in Munich, Germany.
- I describe myself as a tinkerer

Monolithic Application

- Nobody can see what's inside.
- Everyone is watching it.
- Nobody is sure how it stays up.
- Everyone is scared to touch it.
  It might fall/fail.
- Upgrades need a big project,
  and possibly a fork-lift truck.

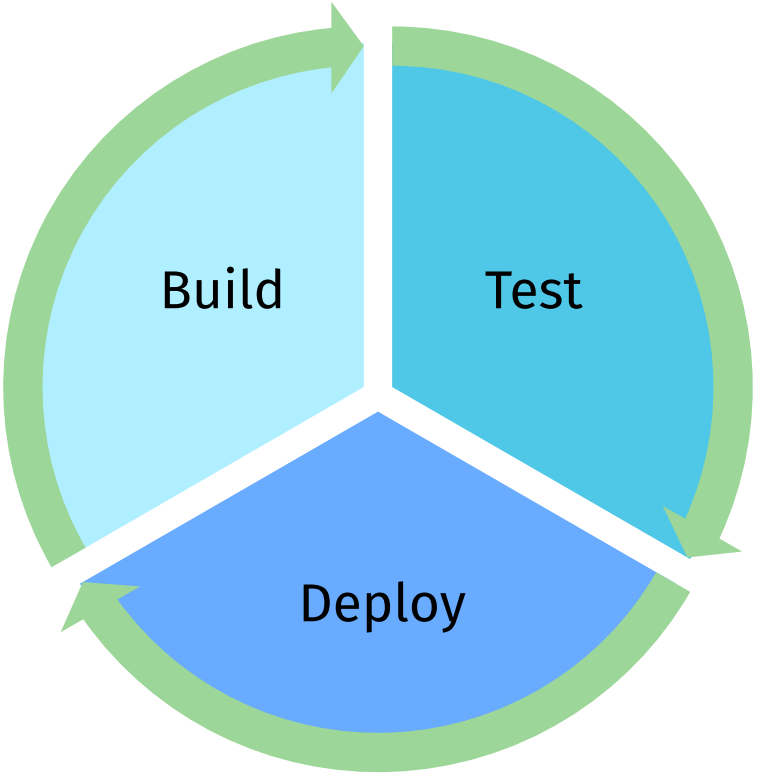
Containerized Application (Microservices)

- Each Container can be removed and replaced (upgraded) without disrupting the others
- Relatively easy to inspect the contents of each container
- Relatively simple to understand and operate
- Unused containers can be removed to save resources.
- Far more portable than monolithic – no forklift required.

What is CI/CD?

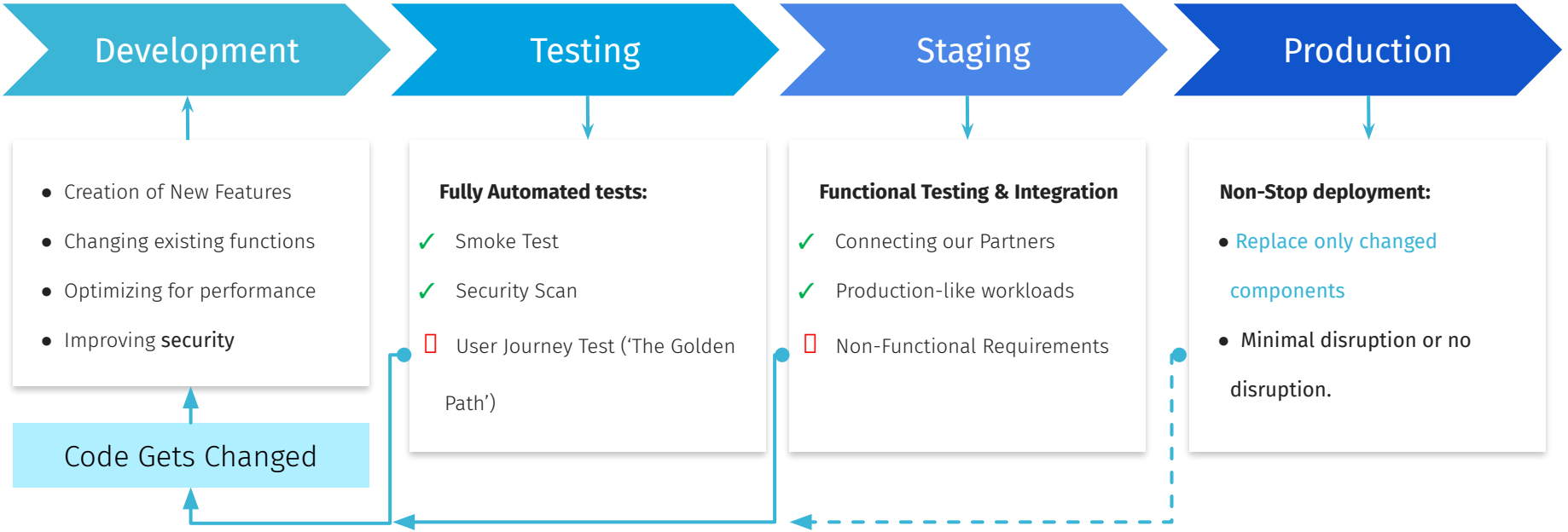
Continuous Integration
Continuous Deployment

A set of principles where improvements to a business application are introduced at a **regular interval** with the **minimum possible disruption.**



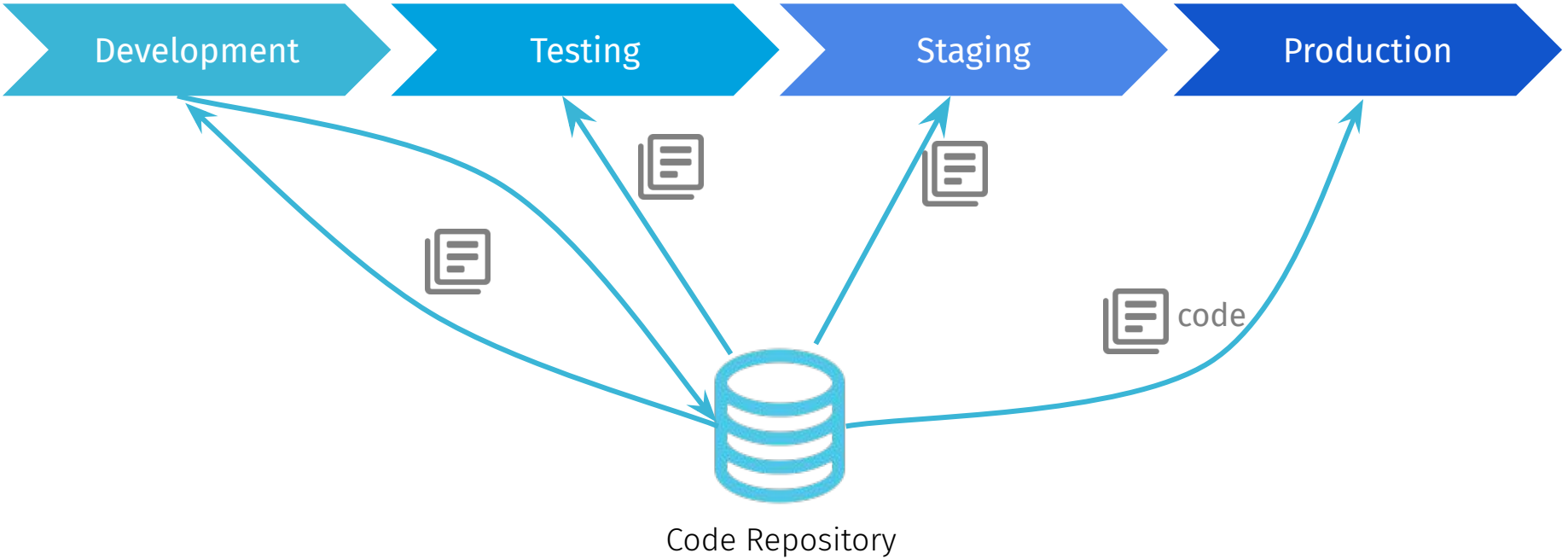
Four Steps – Four Stages

How does application code get to production?



Four Steps – Four Environments

How does application code get to production?



Code Repository

- Holds the assets that create components of the apps
- Controls “versioning” of each asset, allowing multiple iterations of an object to co-exist.
- Uses “releases” to match specific object versions to specific versions of the overall switch
- Drives the deployment pipeline

Code Repository

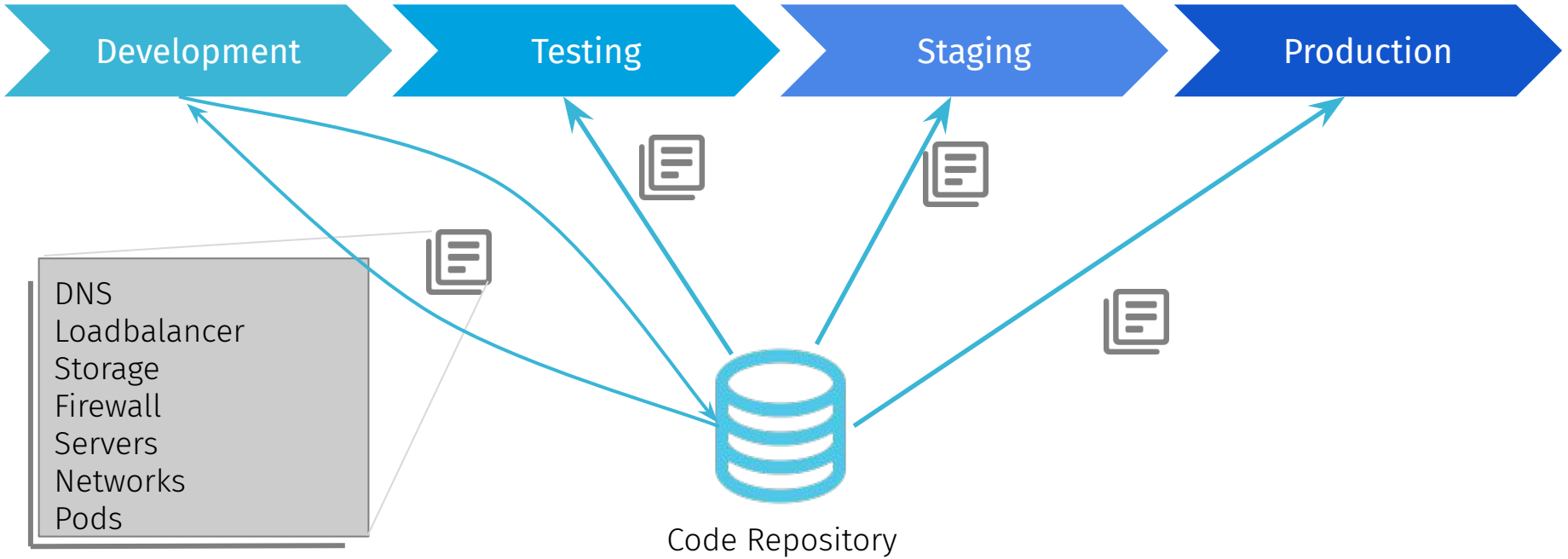
- Approvals protect production from unplanned releases
- Gated tests prevent a ‘release’ from progressing past an environment without passing tests.
- Will apply the same controls to **any asset.**

What about Servers and Networks?

- “The servers weren’t setup the same.”
- “We can’t test the new version until they find some servers to build a new test environment”
- “It worked in test, but the firewall was different in production.”
- “Just let me login to see how it’s setup”
- “The auditor says they need evidence they’re all the same?”

Infrastructure as Code

Describing the fabric of each environment idempotently:



IaC Outcomes

Repeatability

Deploy the same environment again, and again and again, either re-deploying to a specific state, or multiple different environments simultaneously.

Code tested in one environment should work seamlessly in another.

Idempotency

Select an appropriate release number and press apply - unneeded parts disappear, required servers, firewall rules appear. Re-Apply an existing release to remove any ad-hoc changes.

Reliability /Auditability

Anyone can see how things are setup - just check the code. By restricting access to vital system components, and instead configuring them via IaC, changes are always tested before production.

Accelerated Delivery

Features, Changes, Enhancements and patches delivered as fast as your business can test them, with a culture of end-to-end reliability ownership.

Recommended Infrastructure Practice

Defining an Architecture for a successful mojaloop implementation

Design Objectives

Simplicity	Minimise layered complexity - define as much as possible within helm/values files - policy driven deployment, rather than building complex infrastructure.
Security	Separating Execution spaces to minimize Spectre/Meltdown style vulnerabilities Tightly controlled network access footprint 'Shift-Left' model: secure from the start of the pipeline
Efficiency	Reviewing node sizes for production Consolidate workloads onto fewer VMs. Minimise "outside K8S functionality" Ultra-idempotent, Ultra-declarative = Ultra-Re-useable
Community	Open-Source (and Open-Document) sharing MBX market experience with the community. Integrated monitoring and logging solutions

Component Choices

 AMBASSADOR	Ingress Control
 Consul by HashiCorp	Service Mesh
 GitLab	Source Control & CICD
 WireGuard	VPN Solution
 kubernetes	Containerization

- OpenSource with Enterprise options
- Horizontally Scalable

- OpenSource with Enterprise options
- Seamless Terraform Integration
- Supports a variety of underlying network technologies

- OpenSource with Enterprise options
- Git Based
- Easy to learn - highly powerful when scaled

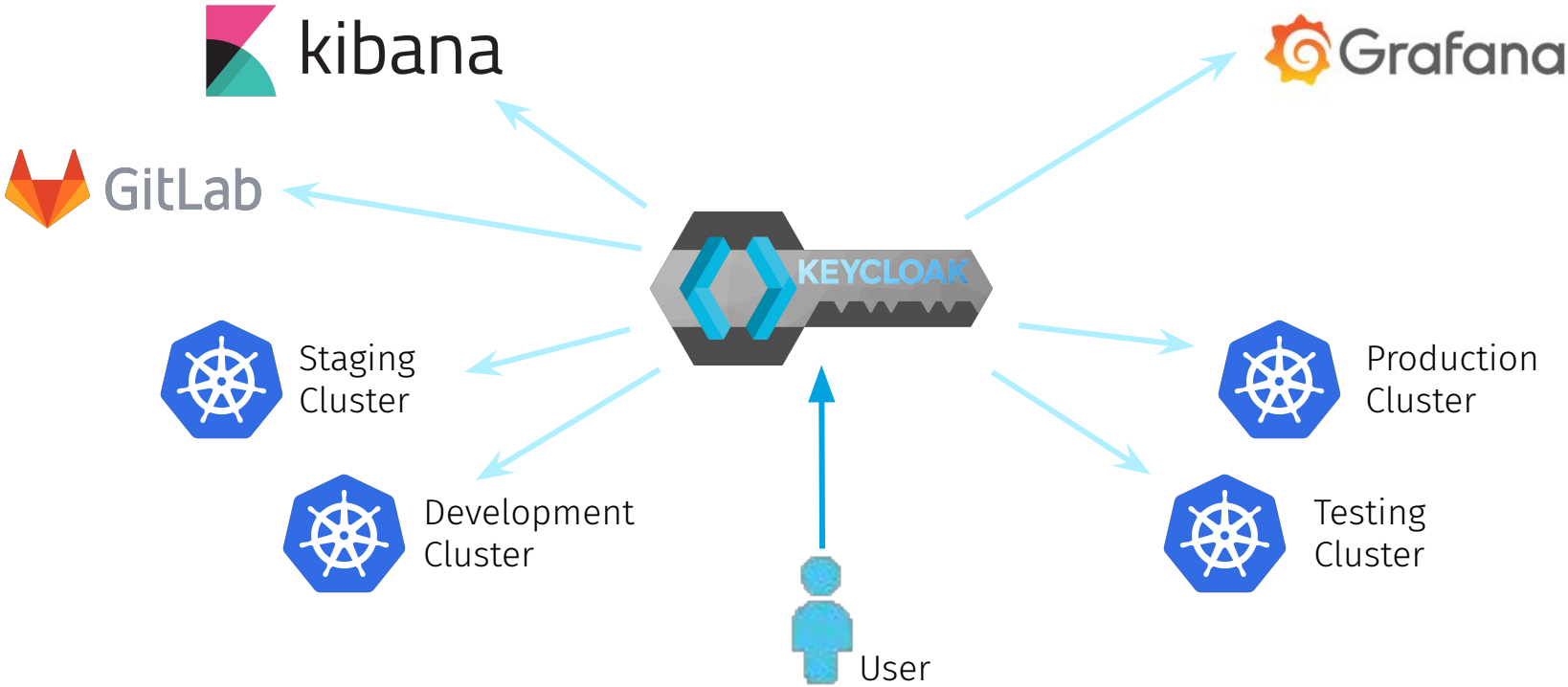
- Built into Linux Kernel (OpenSource) and widely adopted.
- Ultra-High Performance
- Programmatic configuration - Comparatively simple to implement.

- Best-of-breed programmatic software configurability
- Highly Modular / Customizable (see above)
- OpenSource but Available as a managed service

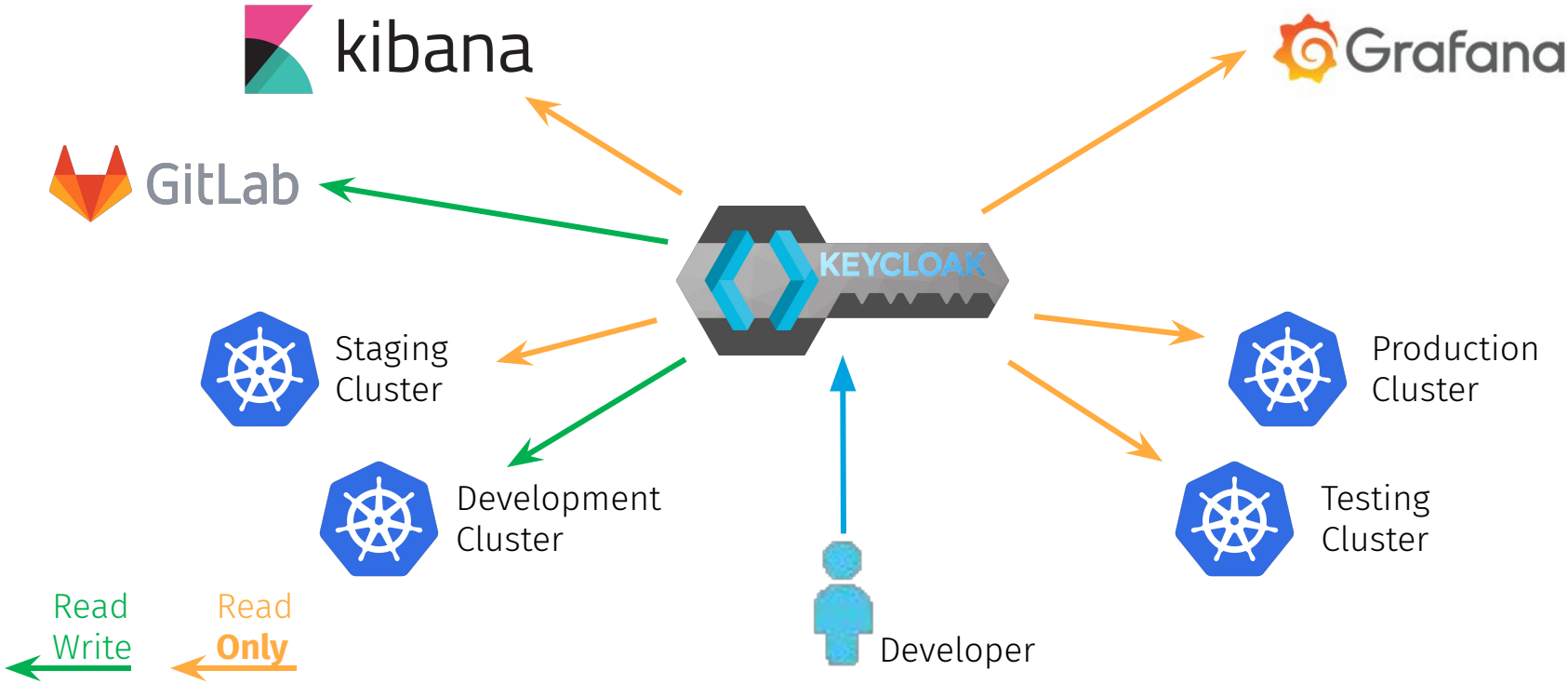
Component Choices

	**Identity Management** • OpenSource with Enterprise options • Programmatically Configurable
	**Secrets Management** • OpenSource with Enterprise options • Seamless Terraform Integration • Enterprise Version Integrates with HSMs
	**Log Event Management** • OpenSource with Enterprise options • Widely used for SIEM Analytics and Alerts • Integrations with most major systems
	**Metrics Management** • OpenSource Project with huge following • Prometheus designed for Kubernetes • Highly extensible
	**Message Streaming** • OpenSource with Enterprise Options • Highly Performant - can be used for logs or for transactions • Massively Scalable and resilient

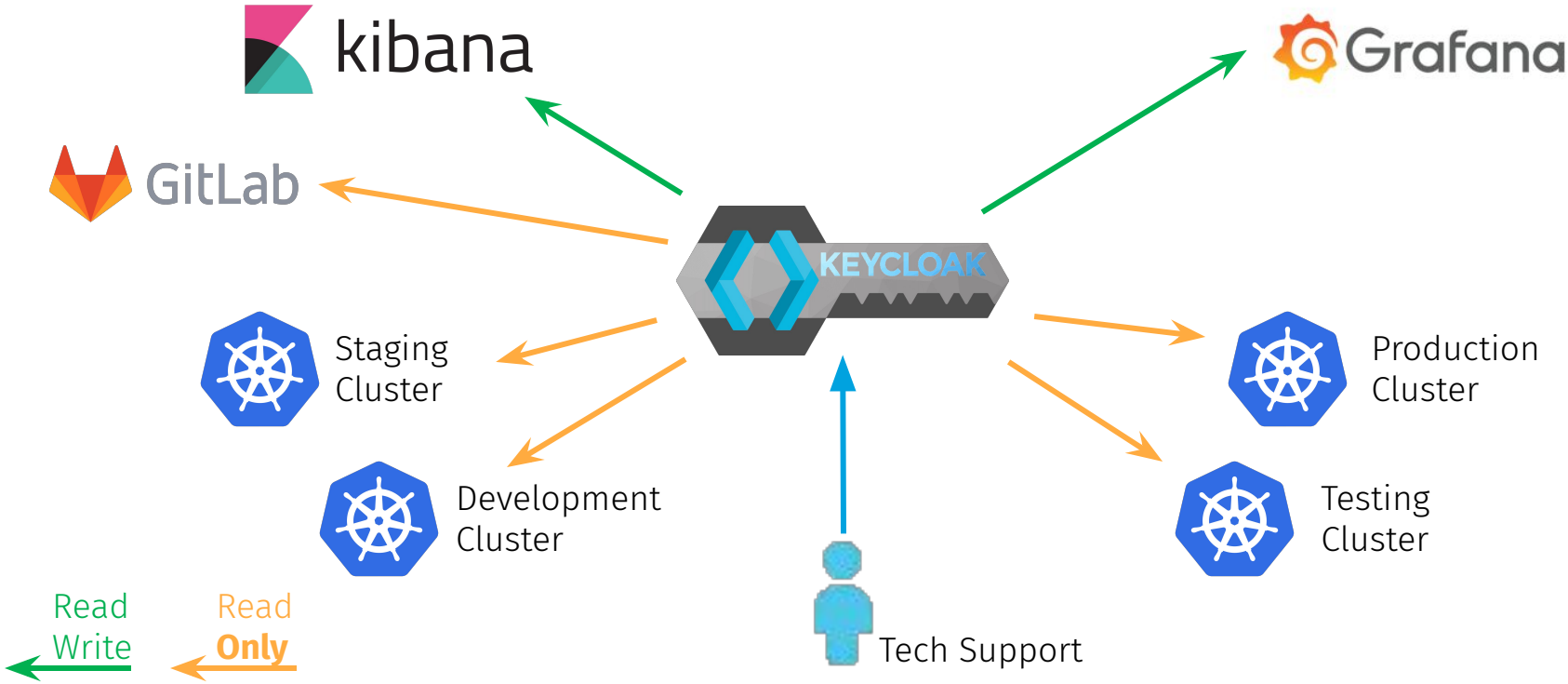
Authentication Architecture



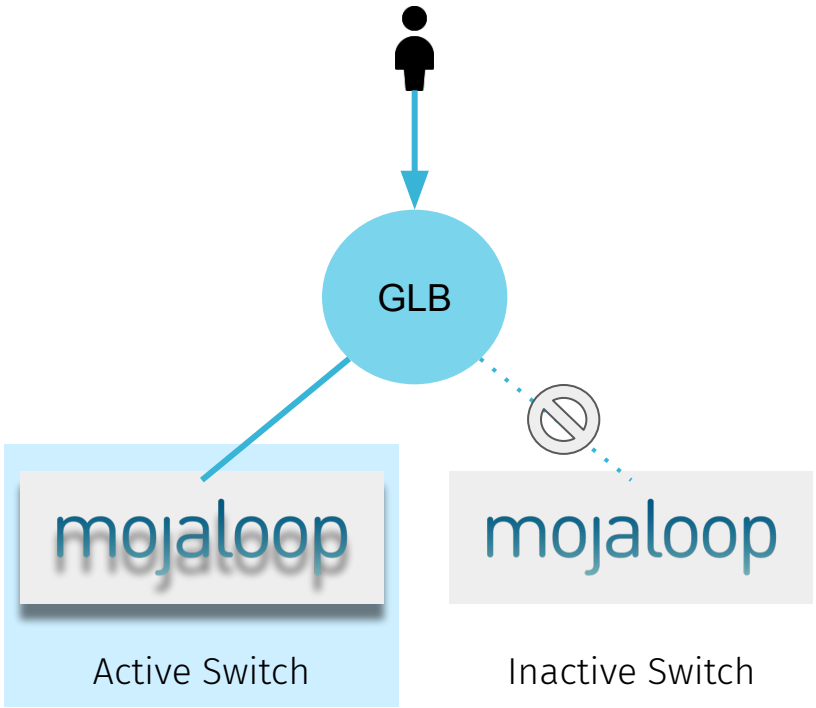
Authentication Architecture - Example



Authentication Architecture - Example



Ingress - Global Load Balancing



AnyCast

Static IP advertised at different internet locations - traffic routes to live target.

AWS Global Accelerator Azure FrontDoor OnPrem BGP

- Near-Instant response. Can be used for load-balancing and network ECMP.
- Requires co-operation of network providers

DNS Redirection

DNS entry directs clients to appropriate IP address

AWS Route53 Azure Traffic Manager

- Slower failover - TTL - clients may be unable to connect
- No availability On Premises

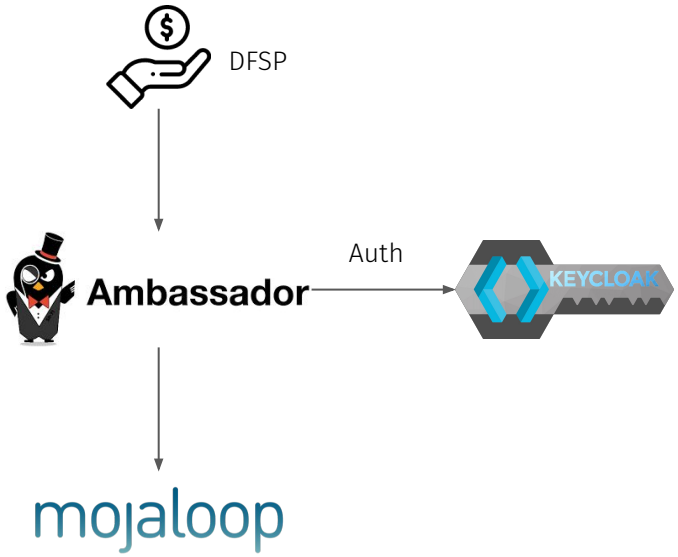
Ingress - Migrating from WSO2 to Ambassador API Gateway

What?

- Envoy Proxy based ingress controller/API gateway
- Cloud native
- Traffic routing to Mojaloop K8S services
- OAuth2 IAM using Keycloak

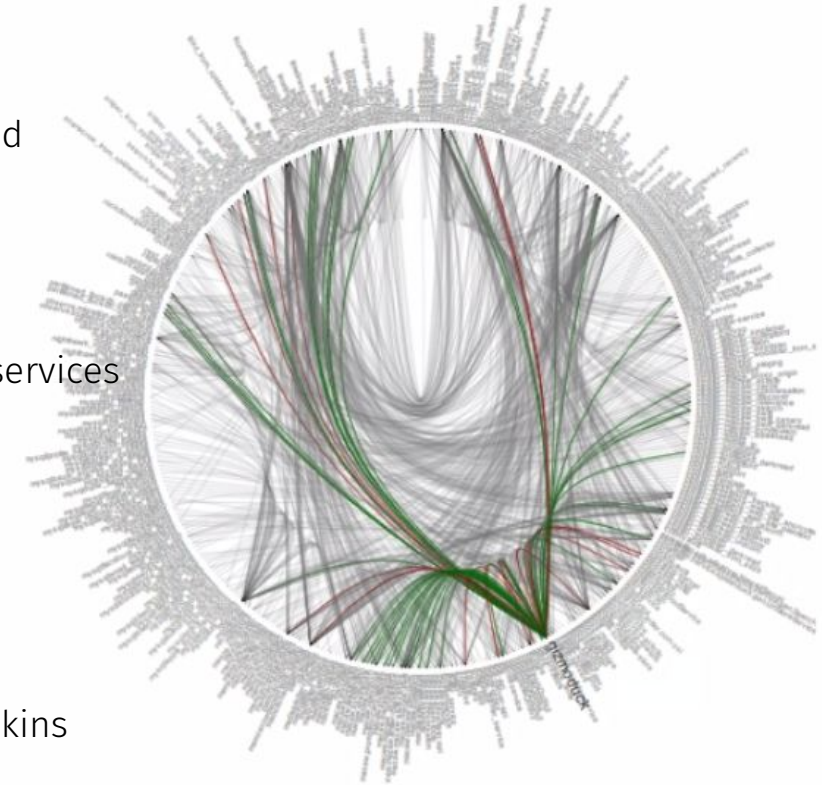
Why?

- Lightweight and fast
- Reduce deployment and maintenance complexity
- Simplification of platform integration architecture



Service Mesh

- Secure service to service communication
  - Transparent mutual TLS provides encryption and authentication between services
- Micro-segmentation
  - Policy based traffic management ensures that services cannot reach out-of-scope endpoints
- Improved observability and monitoring capabilities
  - Troubleshooting issues is easier and faster
  - Can be combined with tracing tools such as Zipkins

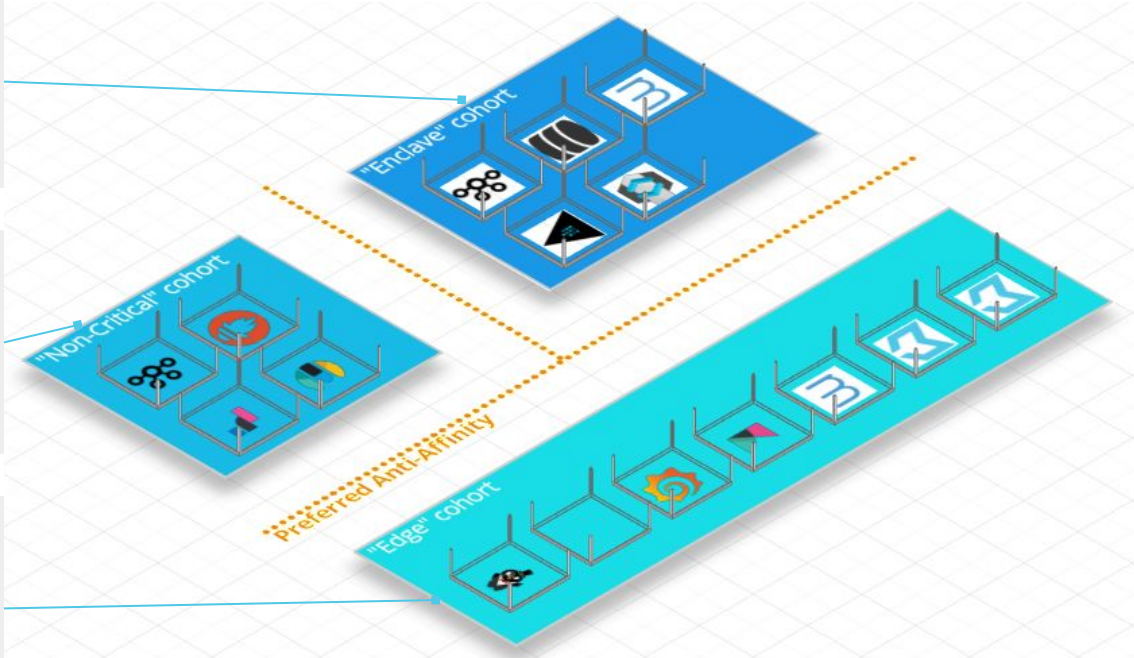


Kubernetes Workload Separation

Hub-Enclave Cohort
"Protected execution" workloads, where the contents of memory might be extremely sensitive : Secrets and user data.

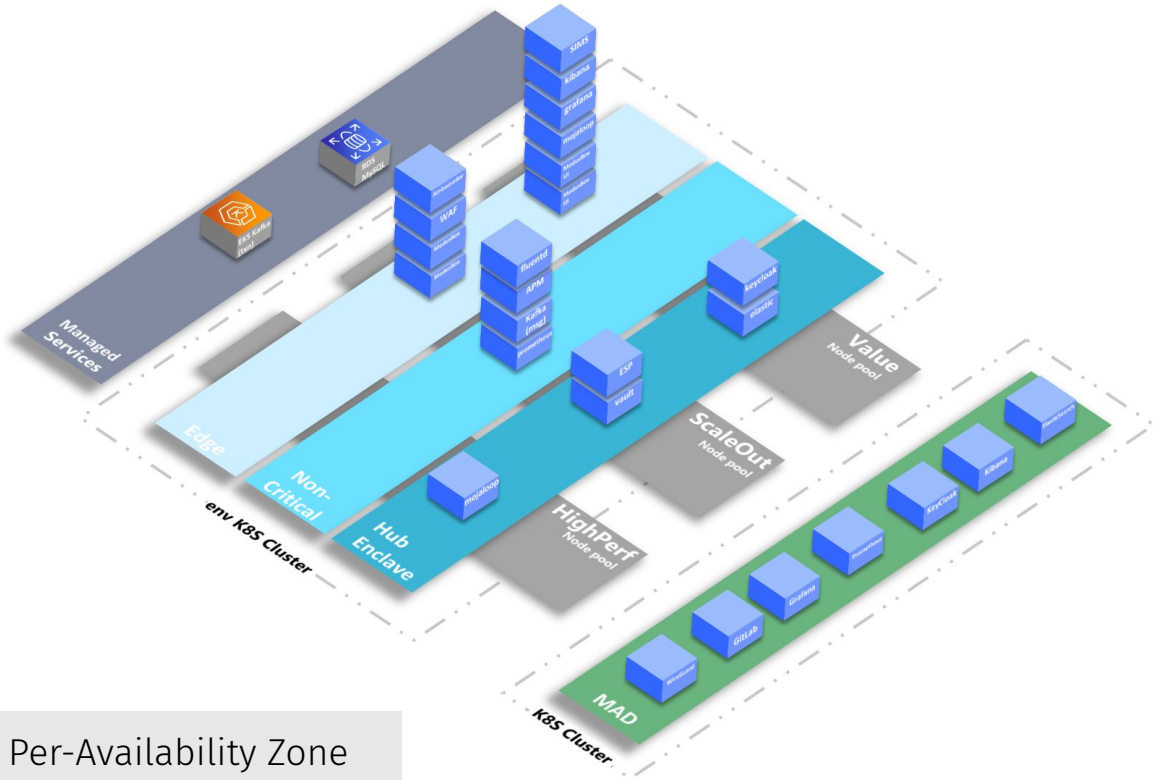
Non-Critical Cohort
Workloads that do not have the same SLA and/or security requirements as the other cohorts.

Edge Cohort
Workloads closest to the network edge - most at risk of compromise.



Scheduler will PREFER to keep pods separate as long as adequate resources are available to do so. Should no further nodes be available, or nodes lost, pods may be forced together.

Scheduling Workloads to Physical Nodes (NodePools)



Per-Availability Zone

Value

t3.xlarge

Burst performance with adequate memory. Low-Throughput components - Grafana, Kibana, Prometheus, Vault, KeyCloak, Wireguard.

Min: 2 nodes

ScaleOut

r5a.2xlarge

Performance/cost combination. Mojaloop components, AddOns and Ingress functions. Scale-Set recommended.

Min: 3 nodes

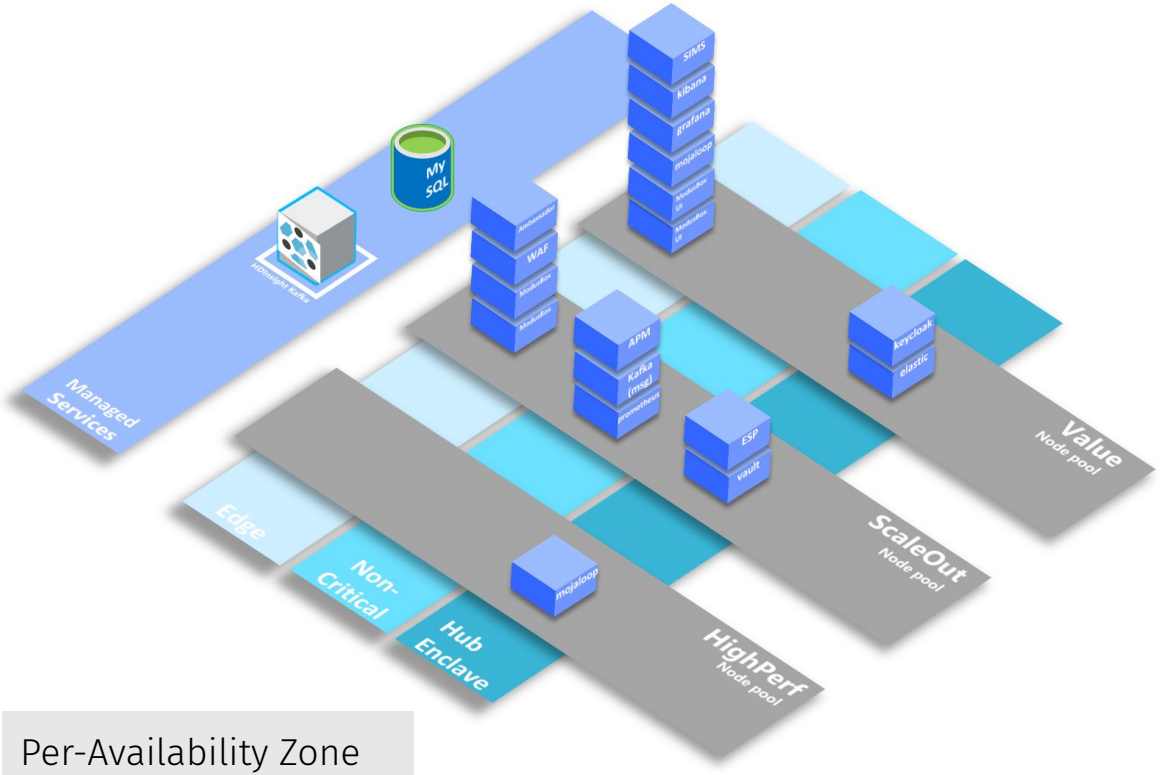
HighPerf

i3.2xlarge

Lowest latency NVMe storage for fast Mojaloop components

Min: 1 nodes

Scheduling Workloads to Physical Nodes (NodePools)



Per-Availability Zone

Value

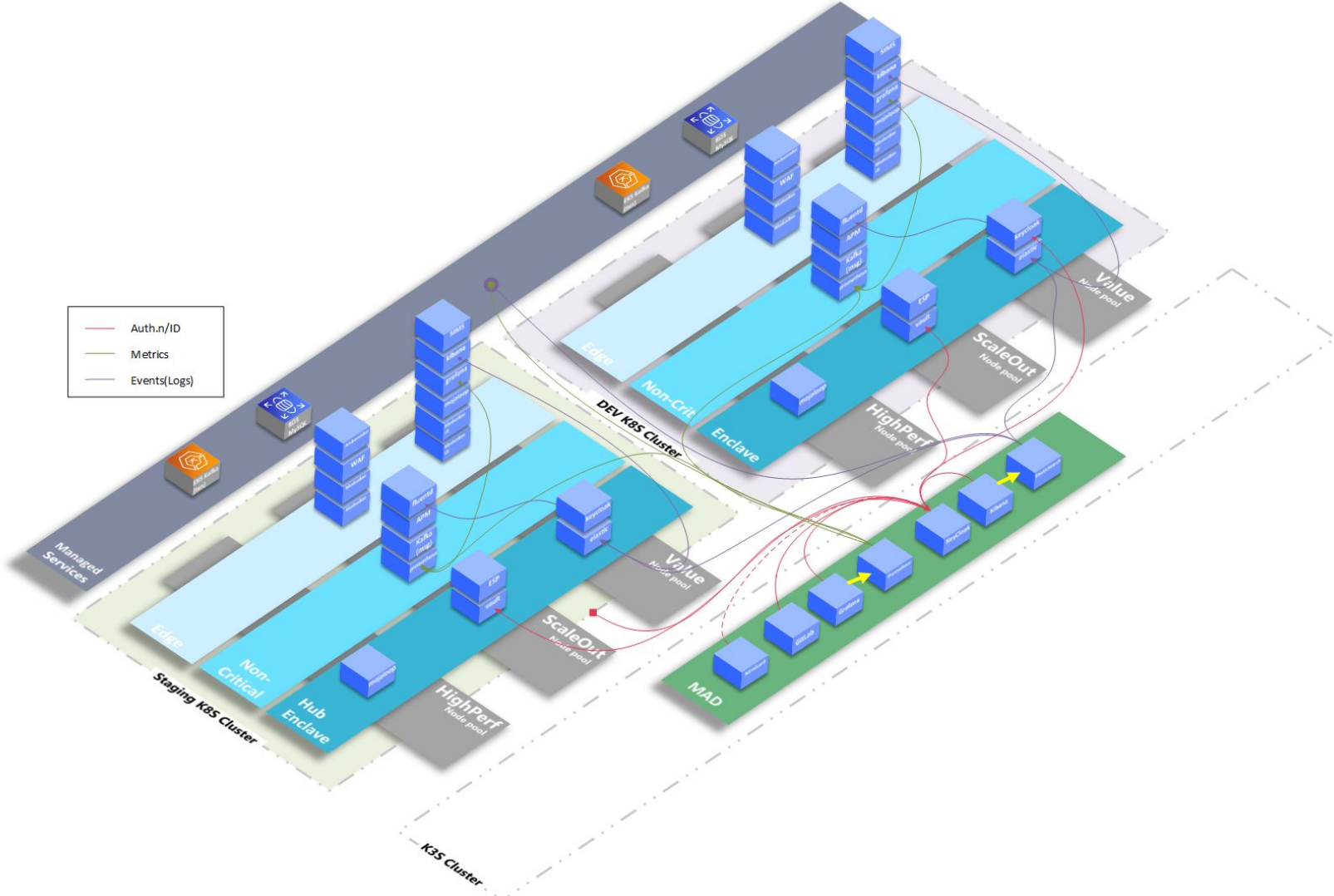
Standard_B4ms
Burst performance with adequate memory. Low-Throughput components - Grafana, Kibana, Prometheus, Vault, KeyCloak, Wireguard.
Min: 2 nodes

ScaleOut

Standard_D8s_v3
Performance/cost combination. Mojaloop components, AddOns and Ingress functions. Scale-Set recommended..
Min: 3 nodes

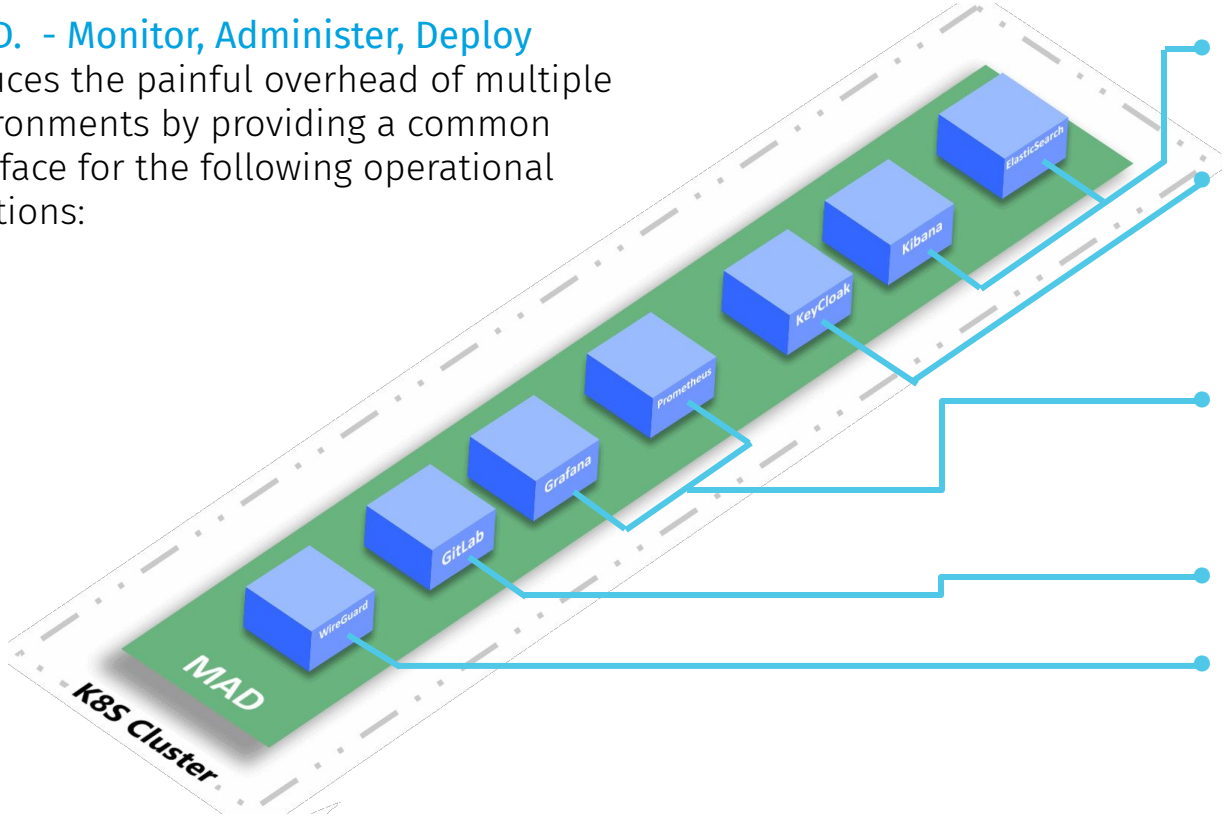
HighPerf

Standard_L8s_v2
Lowest latency NVMe storage for fast mojaloop Components
Min: 1 nodes


Auth.n/ID
Metrics
Events(Logs)
Managed Services
DEV K8S Cluster
Edge
Non-Crit
Enclave
HighPerf
Node pool
ScaleOut
Node pool
Value
Node pool
Staging K8S Cluster
Edge
Non-Critical
Hub Enclave
HighPerf
Node pool
ScaleOut
Node pool
Value
Node pool
MAD
K3S Cluster

Multiple Environments

M.A.D. - Monitor, Administer, Deploy
Reduces the painful overhead of multiple environments by providing a common interface for the following operational functions:



- Monitoring and alerting on events and event thresholds
- Authentication - a single ID may be used throughout environments, with different access levels.
- Monitoring and alerting on Metrics (allows a common ruleset to be used)
- Centralised Code Repository
- Administrative Network Access

Availability Zones - the importance of Latency

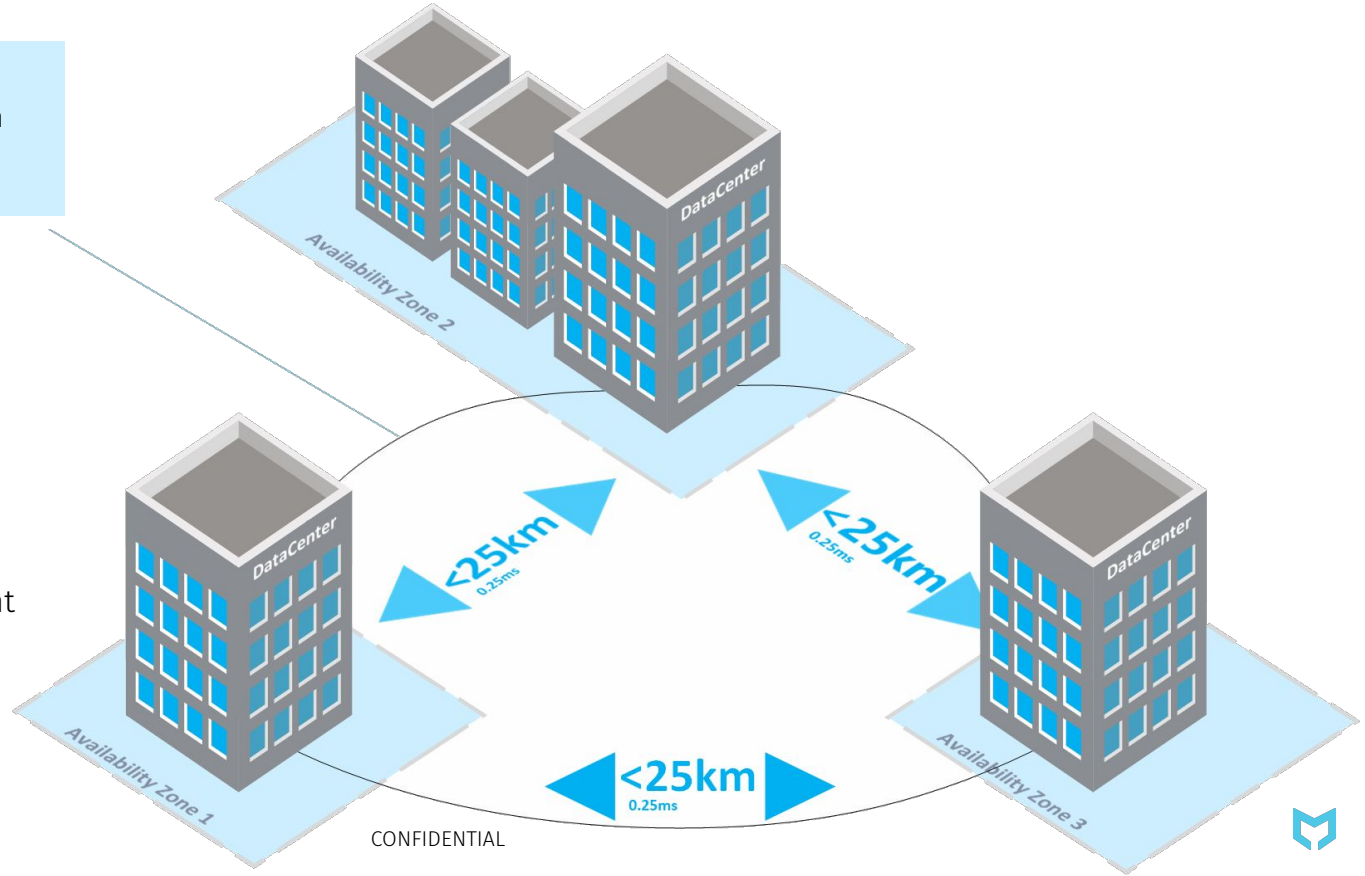
G.652 dark fiber
Latency: 0.25ms / 1km
Speed: upto 40Gbs

AWS and Azure both define “regions” as groups of datacenters in “Availability Zones” with latency below 2ms.

- AWS -VPC, unique L3 subnets
- Azure - vNet with L3 stretch
- Private Cloud:

3 Site Metro Network - redundant dark fiber ring.

Each Zone capable of **66%** production workload.



Database & Replication - Managed Services

	Azure database for MySQL	AWS - RDS-MySQL	AWS - Aurora MySQL
Multi-Master offering	No	No	Proprietary
Failover Mode & Duration	New instance instantiated from Storage Replica. ~4mins.	Synchronously Replicated Instances - unavailable until failover. ~5mins	No Failover. Unavailable node will timeout current transactions.
Read Response	>4ms (Inter-AZ + DB)	>4ms (Inter-AZ + DB)	Local
Write Response	DB+ ~8ms (repl time)	DB+ ~8ms (repl time)	DB+ ~8ms (repl time)
SLA Availability %	99.99%	99.95%	99.99%
SLA Outage Time	4m22s / month	21m54s / month	4m22s / month

MySQL is inherently designed as a single-master database

Database & Replication - OnPrem

Worst-Case Scenario: No Parallelism

max_dop=1

Target : 300 DOPS
Txn Time: **0.3ms**

With synchronous replication, a 25km link gives us a RTT **0.25ms**

Small margin of 0.05ms for overheads.

Sharding is critical! Maximum Parallelism.

On perconaDB (multi-master) - `wsrep_causal_reads=ON` prevents reads from serving stale data at the expense of read performance.

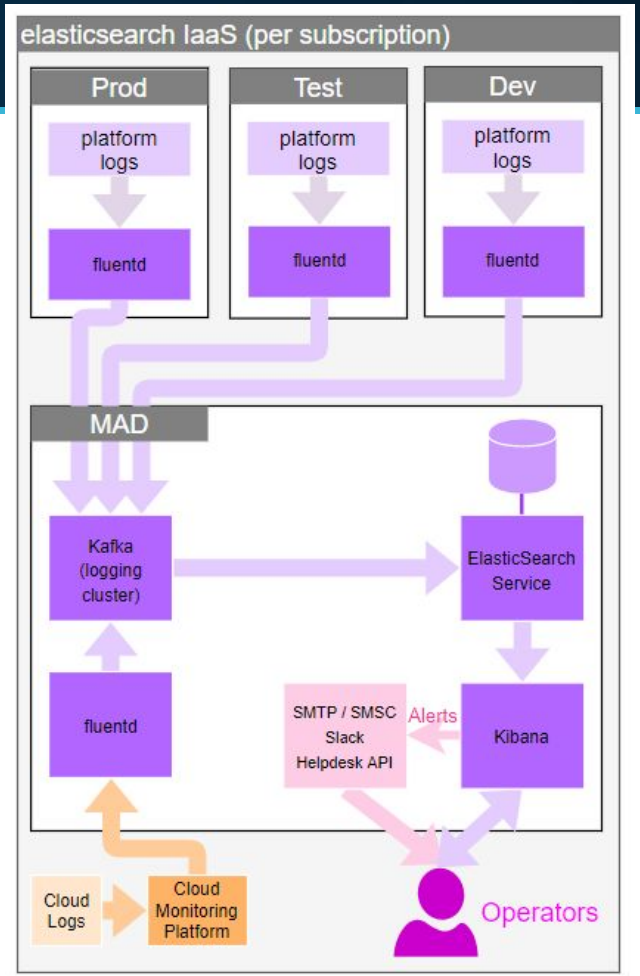
Consequences:
Cluster will move at the performance of slowest server to certify a commit.

Timeout on certify = “Transaction Failed” for end-user.

MySQL is inherently designed as a single-master database

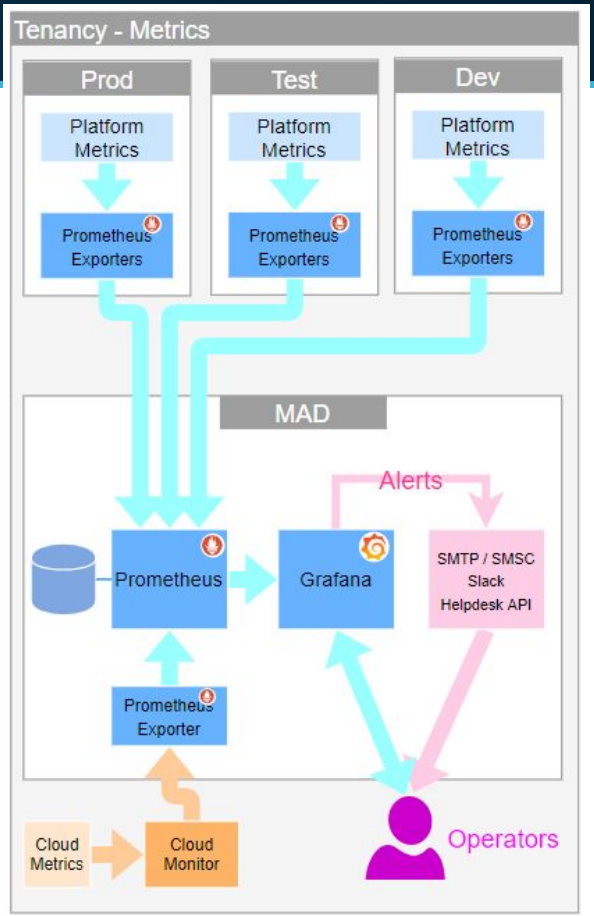
Event Logging and Management

- Logs collected by a fluentd exporter(s) in each env.
- A fluentd exporter collects cloud-platform events.
- Logs are passed to kafka for scalable, reliable transmission.
- ElasticSearch collects Events from Kafka Logging Cluster. LogData is retained for all envs for historical analysis.
- Kibana is used for querying, dashboarding and alerting.



Performance and Metrics

- Prometheus instances in each env collect and export metrics
- Dedicated prometheus exporter collects cloud metrics and exports
- Central PrometheusDB and Grafana instance offer “single-pane-of-glass” performance monitoring and alerting of all envs.
- Historical performance data is retained from all envs for future analysis

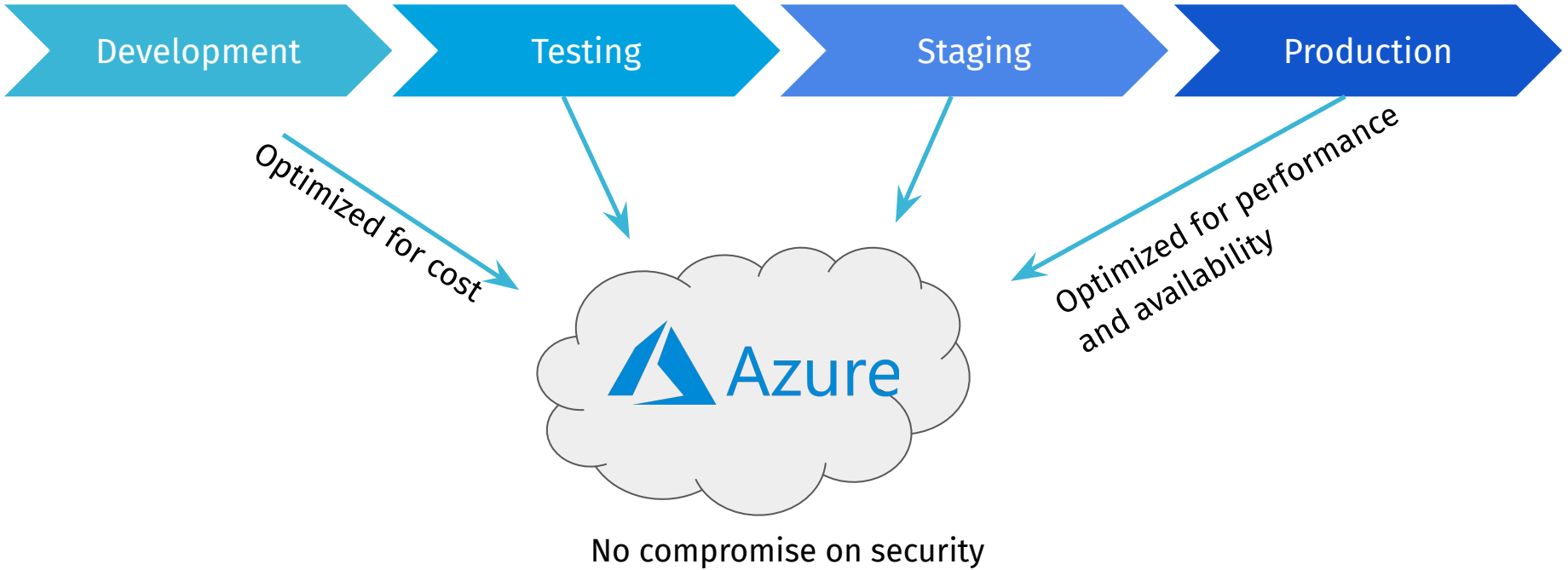


Designing For Azure

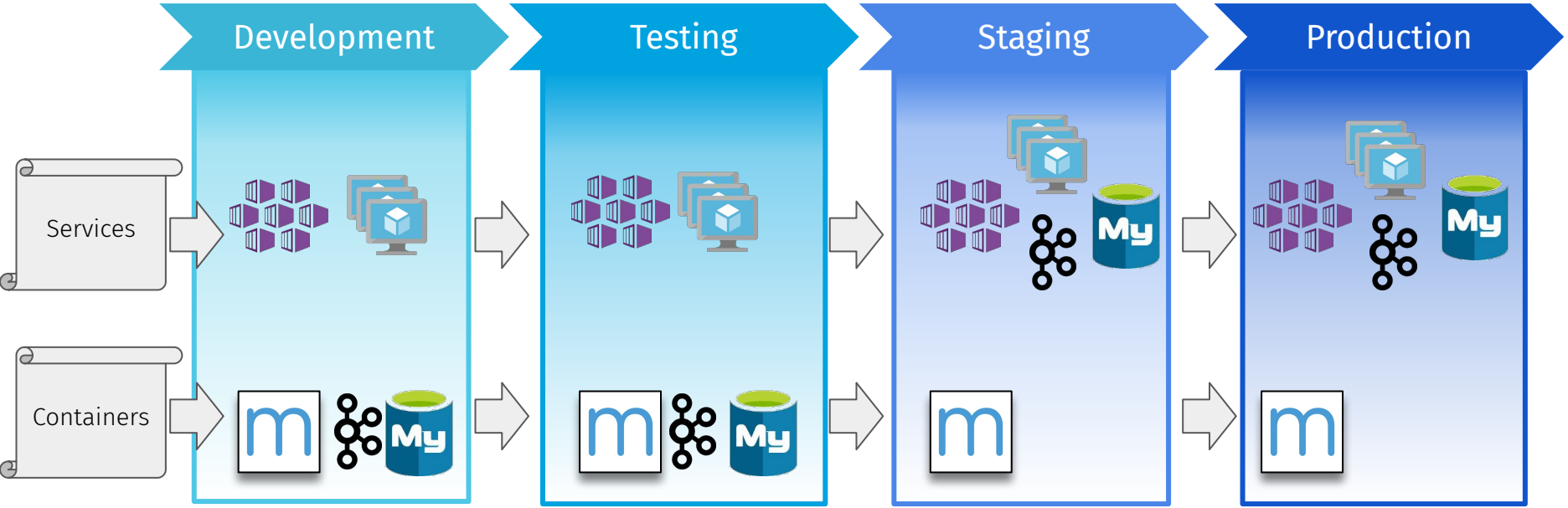
Defining an Architecture for a successful mojaloop implementation in Microsoft Azure

Four Steps – Four Environments - Azure

How does application code get to production?

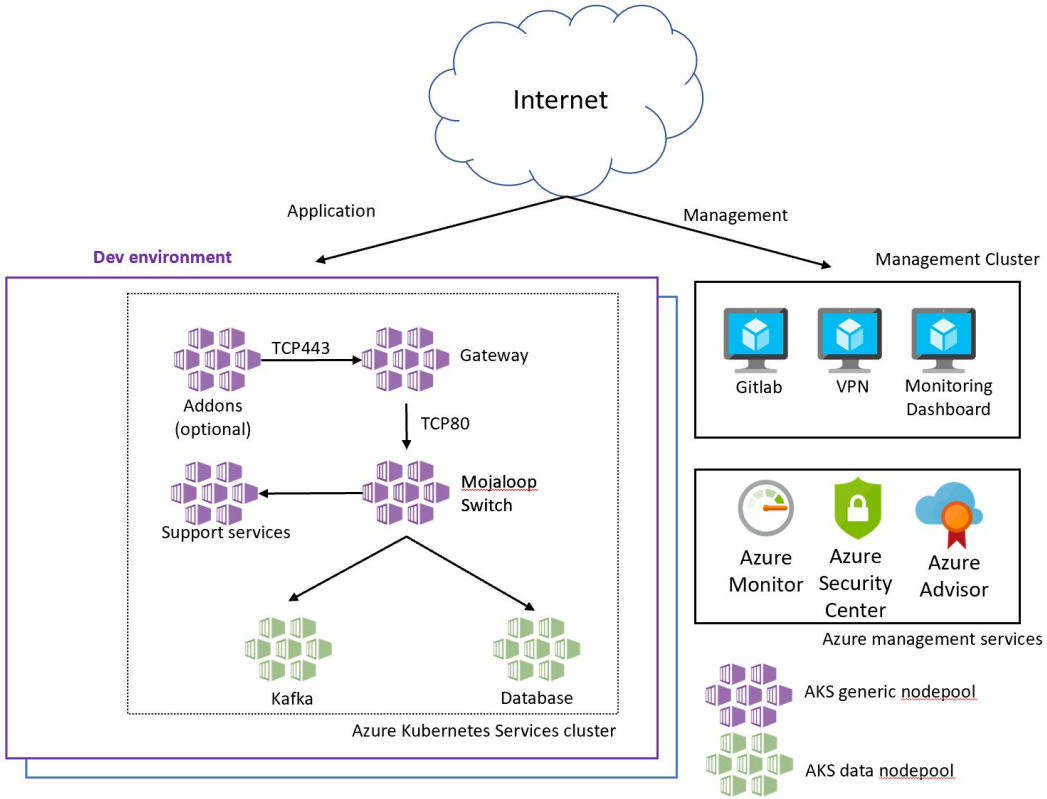


Four Steps – Four Environments - Platform vs App

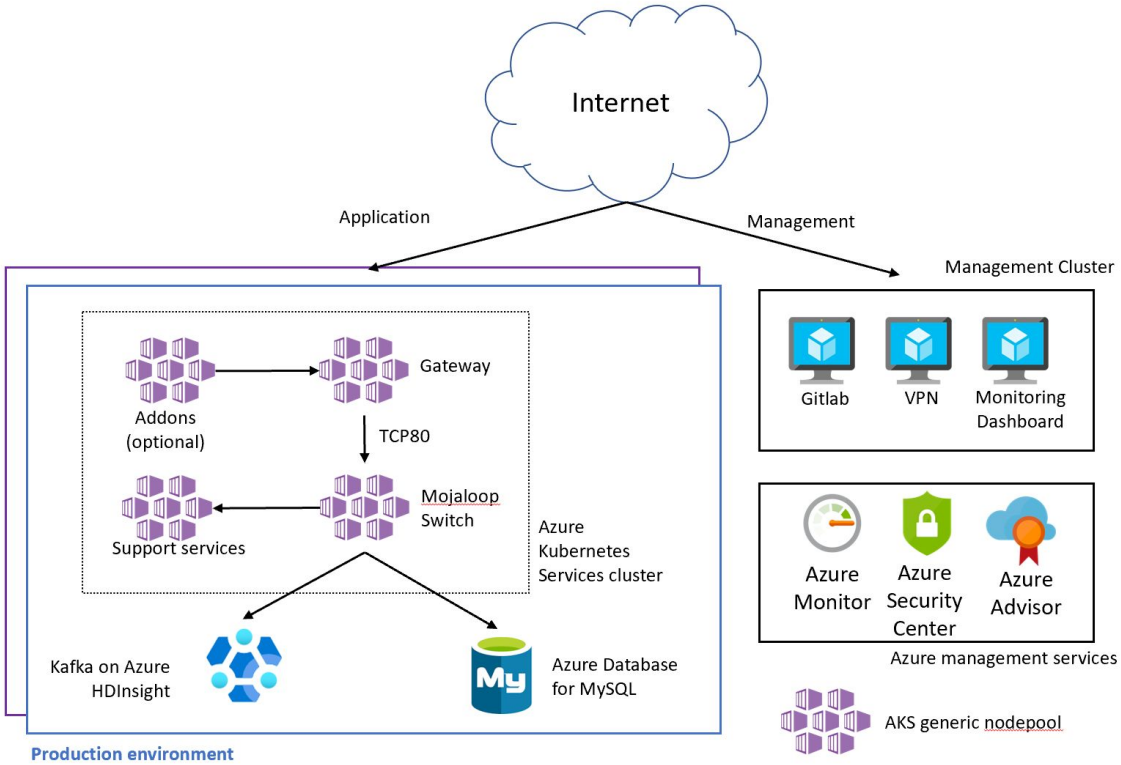


Platform optimized for each cloud and environment type
Application rollout cloud and environment agnostic

Example: Development environment in Azure

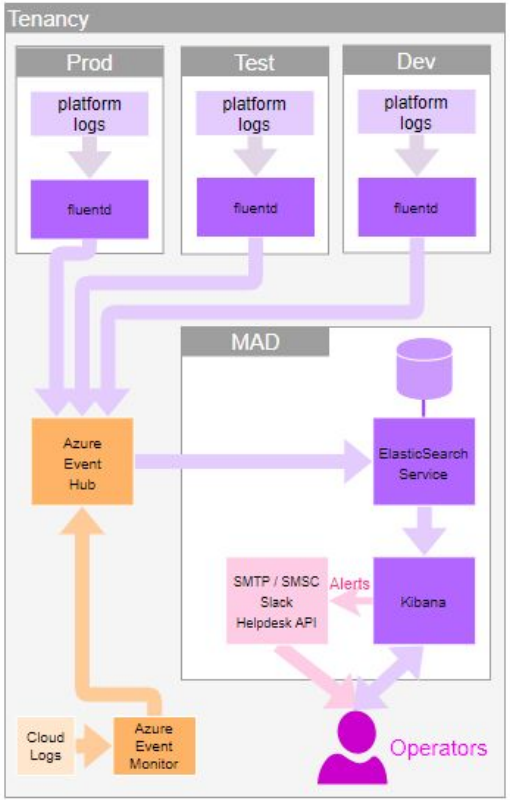


Example: Production environment in Azure



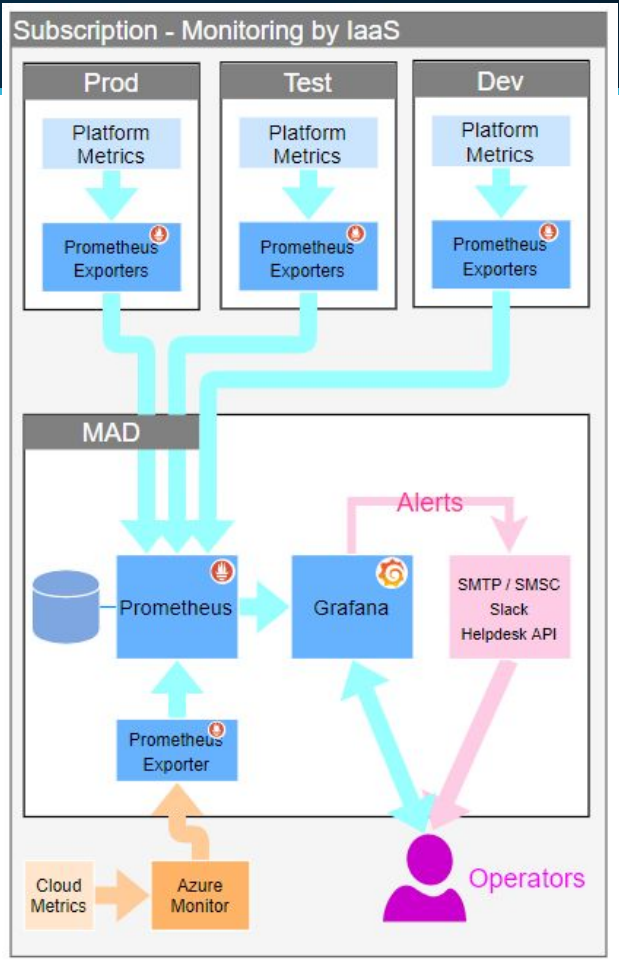
Event Logging and Management

- Logs collected by a fluentd exporter(s) in each env.
- Azure EventHub receives cloud platform events from Azure Event Monitor, as well as fluentd exporters. EventHub is less expensive than a Kafka cluster.
- ElasticSearch collects events from Azure Event Hub
- Kibana is used for querying, dashboarding and alerting.



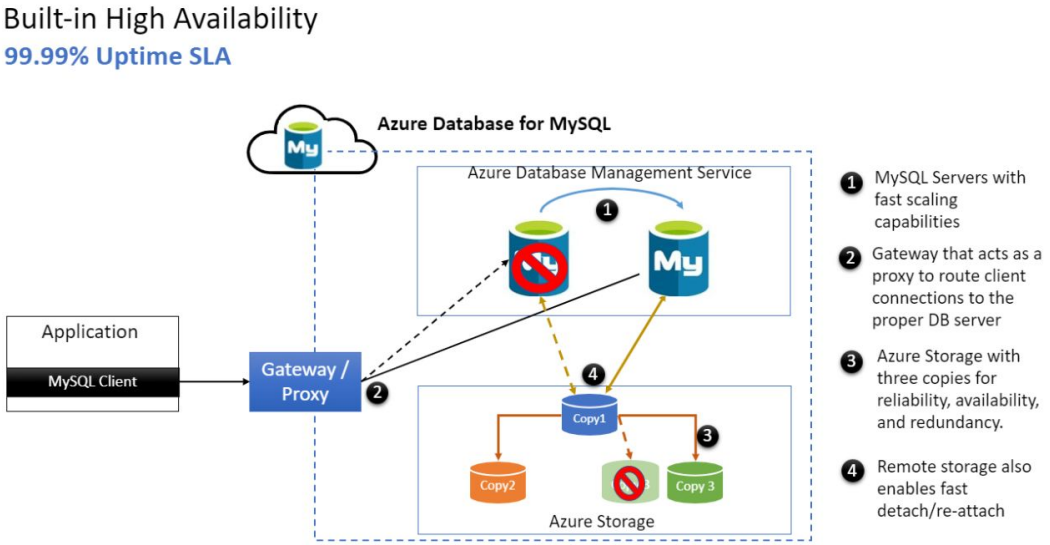
Performance and Metrics

- Prometheus instances in each env collect and export metrics
- Dedicated prometheus exporter collects cloud metrics and exports
- Central PrometheusDB and Grafana instance offer “single-pane-of-glass” performance monitoring and alerting of all envs.
- Historical performance data is retained from all envs for future analysis



In-region resiliency: Azure Database for MySQL

- Azure Database for MySQL offers an in-region SLA of 99.99%
- Data is stored in multiple places, and in case of a failure failover happens automatically
- No cross-AZ replication supported



https://docs.microsoft.com/azure/mysql/select-right-deployment-type

Cross-region resiliency

- Stateless infrastructure and app deployment
  - IaC automation will deploy the Kubernetes clusters and other elements of the cloud platform in the secondary region, as well as the application

- Database replication: largest factor for the Recovery Point Objective (RPO)
  - Point-in-time backups will offer around 1 hour RPO
  - Read-replicas will offer a variable RPO (seconds/minutes), depending on the distance between the databases and data churn rate

- Global Load Balancing will converge customers to the new region
  - Anycast-based (same IP address advertised from different regions): Azure Front Door
  - DNS-based (intelligent DNS resolution): Azure Traffic Manager

mojaloop-IaC

Accelerated and Repeatable delivery of recommended practice

The story so far.....IaC **v1**

Requirement:	Build a performant, highly available mojaloop switch in Azure consisting of 3 or more environments.

- Scant Idempotency
  - Redeployment of anything is a “big deal”
- Expensive Choices - Managed Services
  - The quest for reliability and performance
- Highly Procedural
  - Minimal Terraform
  - Heavy reliance on shell scripting
  - First this, then that.

e2eDT (End-to-End Deployment Time):

- 2+ days – Infrastructure Fabric
- 2 days – Firewall Configs
- 2 days – Component Deployment
- 2 days – Manual Component Configuration
- 6+ days – Snagging, Fixing, connecting

The story so far.....IaC **v2**

Requirement:	Build a performant mojaloop switch in AWS consisting of 3 or more environments, with lower operational cost than IaCv1.

- Improved Idempotency
  - Removal of PFsense improves idempotency
  - Redeployment of anything is still a “big deal”
- Less Expensive Choices
  - Single Availability Zone
- Highly Procedural
  - Minimal Terraform
  - Heavy reliance on shell scripting
  - “First this, then that.”

e2eDT (End-to-End Deployment

- 2+ days – Infrastructure Fabric
- 2 days – Firewall Configs
- 2 days – Component Deployment
- 2 days – Manual Component Configuration
- 6+ days – Snagging, Fixing, connecting

The story so far.....IaC **v3**

Requirement:	Build several performant mojaloop switches in AWS consisting of 3 or more environments, and remove after use.

- Pipeline Idempotency
  - Re-running pipeline no longer re-deploys environment
  - GitLab selected as repository and CI
- Component Choices
  - Choices lean towards economy more than availability – limited performance target
- Declarative Configuration
  - Small changes do not have big impacts
  - Significant Terraform investment

e2eDT (End-to-End Deployment Time):

- 1 day - preparation
  - MAD tier : bootstrap, code repo, networks, management tools, pipline
- 1 day – platform deployment
  - Pods and Storage – Automated Testing!
- 1 day – testing and final config
  - Custom config, manual testing
- 2 days – handover
  - monitoring , documentation

Coming Soon - IaC **v4**

Coming Soon!

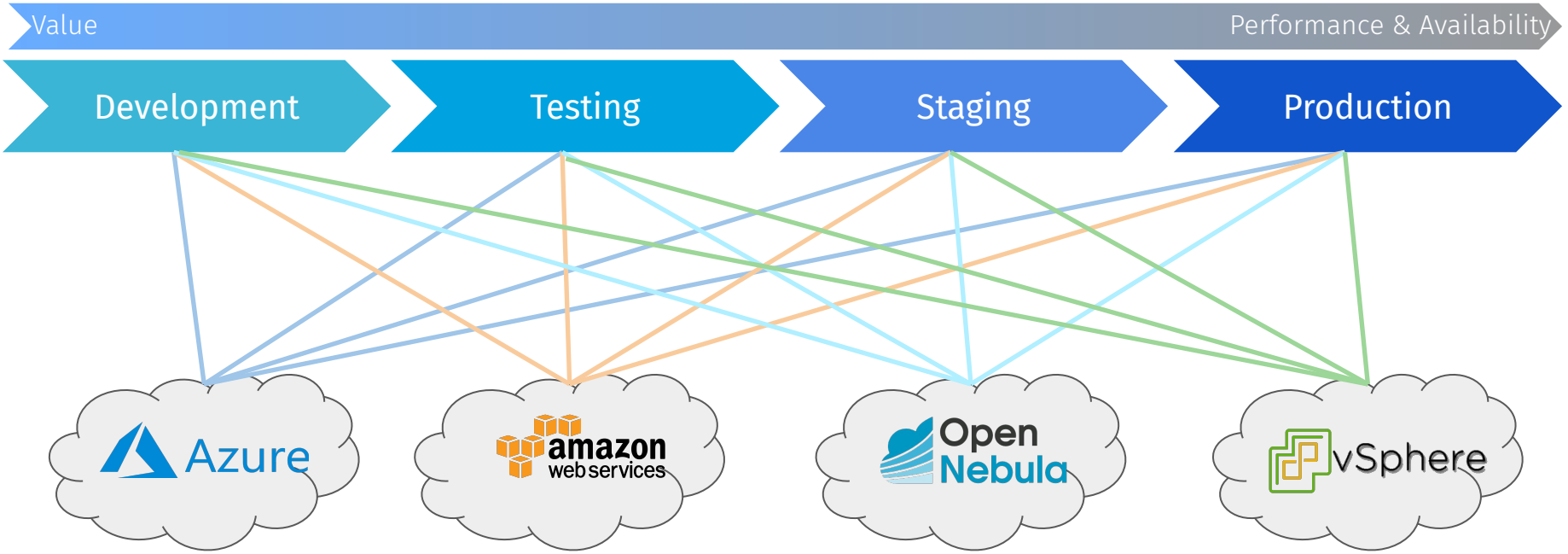
- Should be prepared for an Open-Source release
  - Code and Documentation
- Aligned with Community Recommended Practice Architecture
- “Add Simplicity” – consolidate functionality into fewer ‘tiers’
  - Micro-segmentation - policy-based network security
  - Kubernetes policies vs multiple clusters
- Improve efficiency
  - Reviewing node sizes for production
  - Consolidate workloads onto fewer VMs.
  - Minimise footprint outside K8S
- Increase Security

e2eDT target

Coming Soon - IaC **v4 -** Security Improvements

- Security Information and Event Management (SIEM)
  - Sourcing data from logs, metrics, or external tools such as Azure Security Center or AWS GuardDuty
- Actively monitoring the container usage
  - Identify unusual patterns
  - Scanning containers looking for CVEs
  - Use Web Application Firewalls to inspect traffic at the gate
- Ensure that the security standards are met at all times
  - RBAC
  - Strong Pod Security Policies
  - Restrictive Network Policies
  - CIS Kubernetes Benchmarks

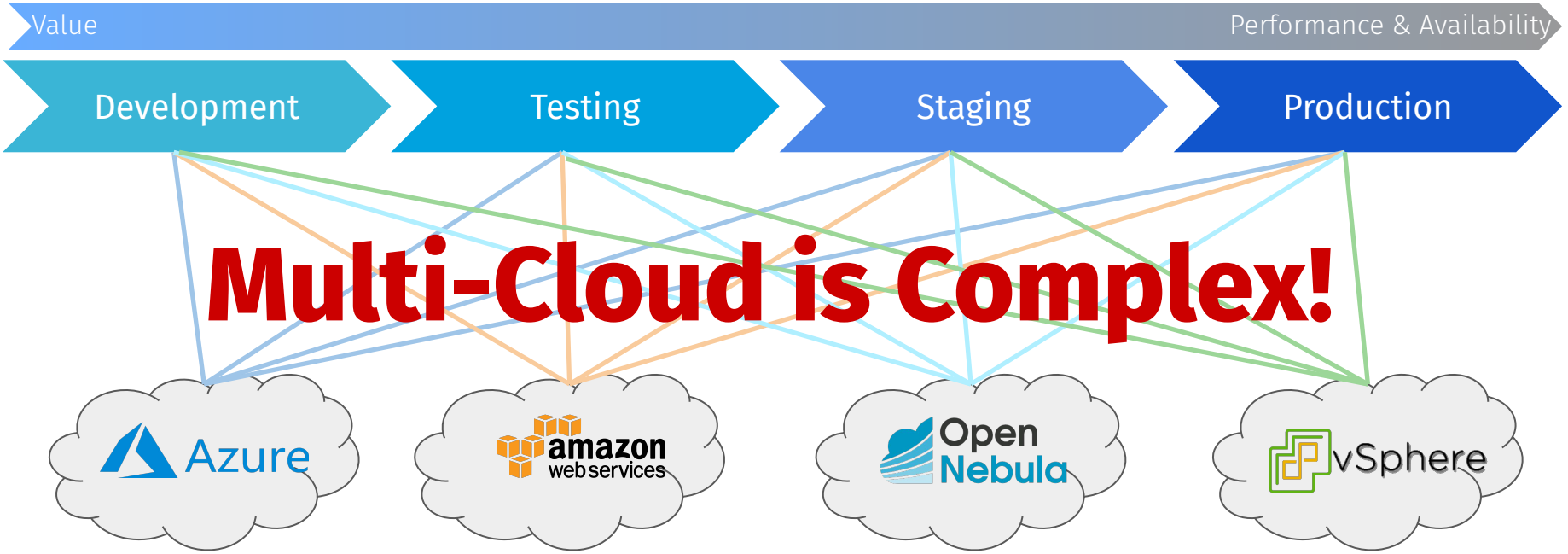
The Dream: Four Environments - Any Cloud



No compromises on security

No compromises on compliance

The Dream: Four Environments - Any Cloud



No compromises on security

No compromises on compliance

IaCv4 - PolyNimbus strategy

IaC today (IaCv3)

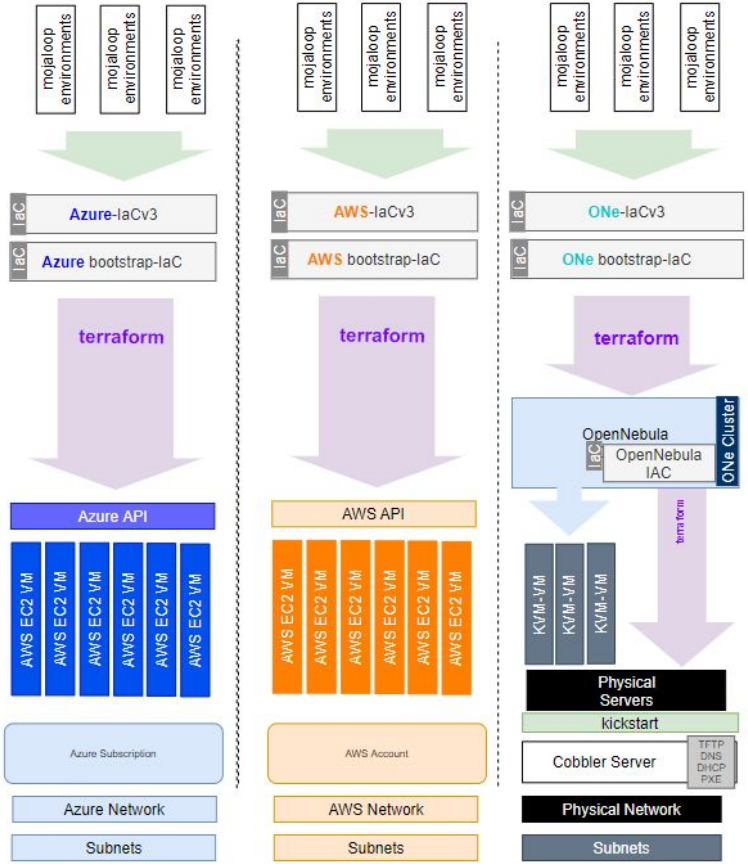
Unique codebase for each provider.

Different MBX customers in different clouds are using different versions of IaC.

Versions are not Synchronised.

On-Premises deployments are similar, but cannot be identical to public-cloud deployments.

A lot of work expended automating OpenNebula deployment - but every customer is different.



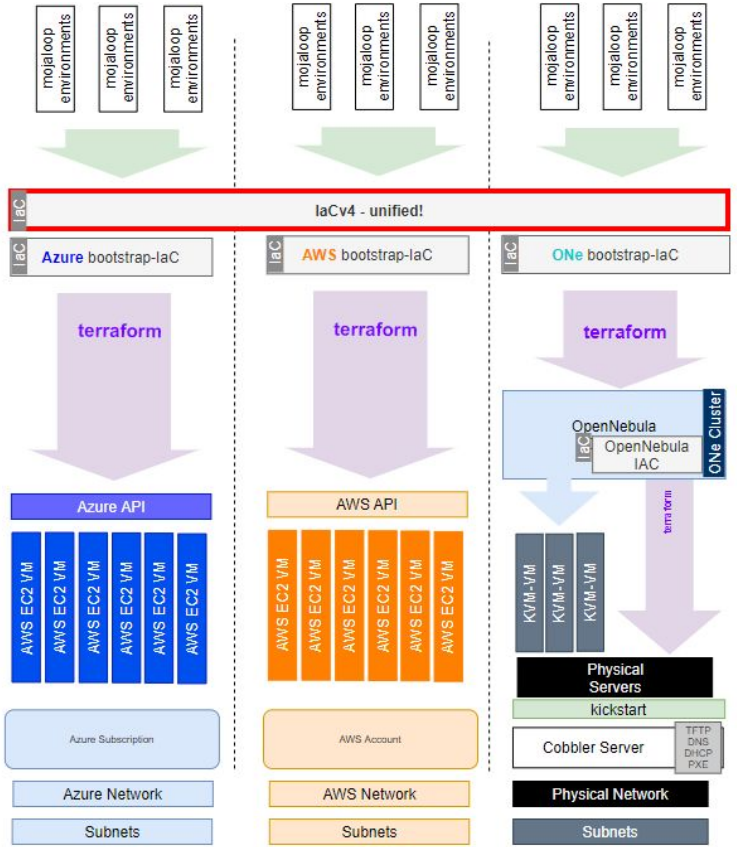
IaCvX - PolyNimbus strategy

Unifying the IaC codebase

- Simplifies deployment in different clouds
- Less code to maintain
- Easier for community to consume
- Allows for recovery to a different cloud if yours becomes unavailable.

However...

- “Bootstrap” modules vary between clouds.
- On-Prem is still highly complex - costs more upfront
- Clouds are still isolated from each other



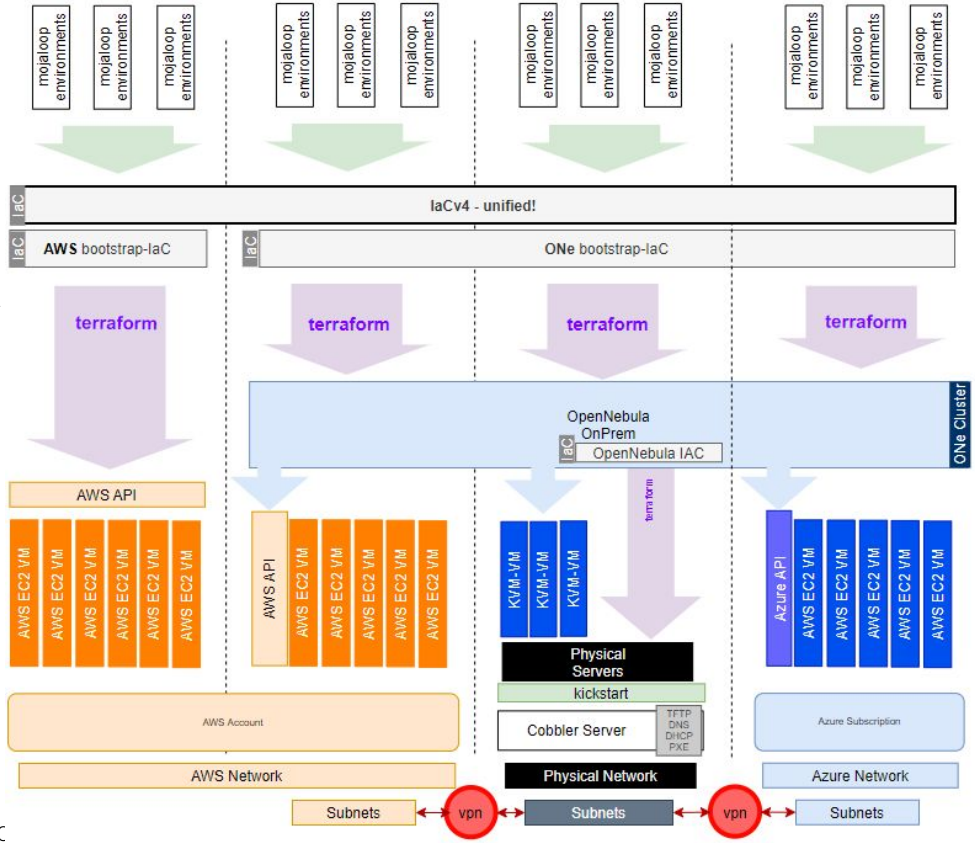
IaCvX - PolyNimbus strategy

Multi-Cloud deployment

- Limited connectivity between clouds
- Single codebase on-prem and public cloud
- Strong Business continuity possibilities
- Potential for scale-out/hybrid-cloud model
- Limited management from OpenNebula
- Converged Monitoring platform
- Retains separate bootstraps for public cloud.

However...

- Complex to manage and support
- On-Prem is STILL highly complex
- Whilst stretched subnets are possible, they require complex design.



CC

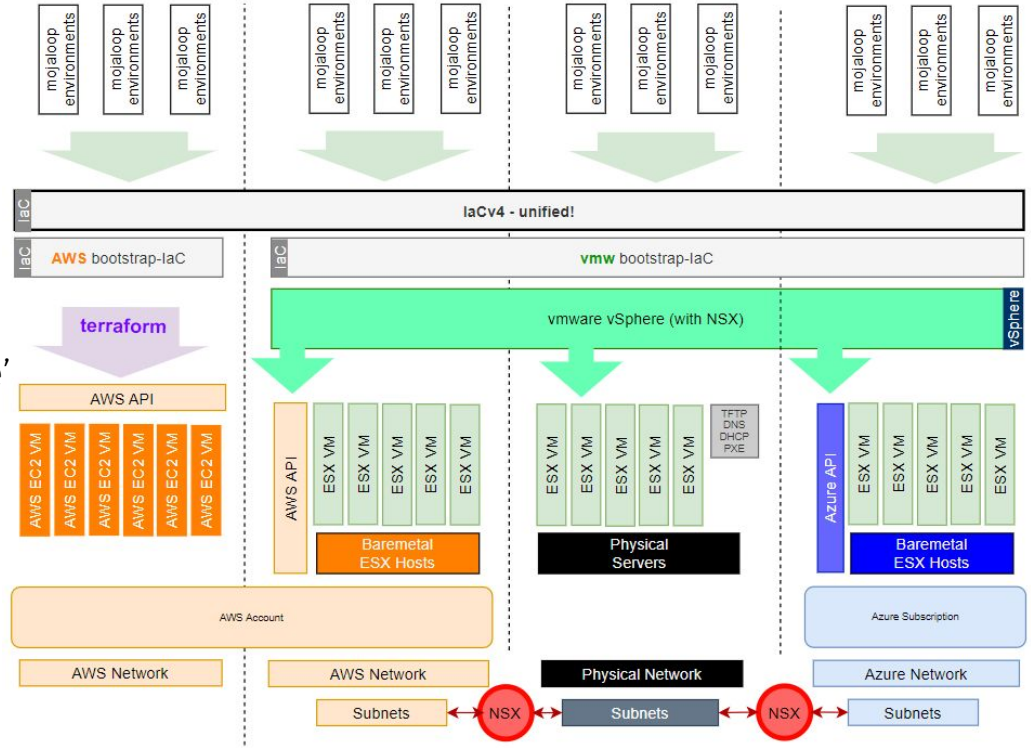
IaCvX - PolyNimbus strategy

vmware Virtual Cloud Platform

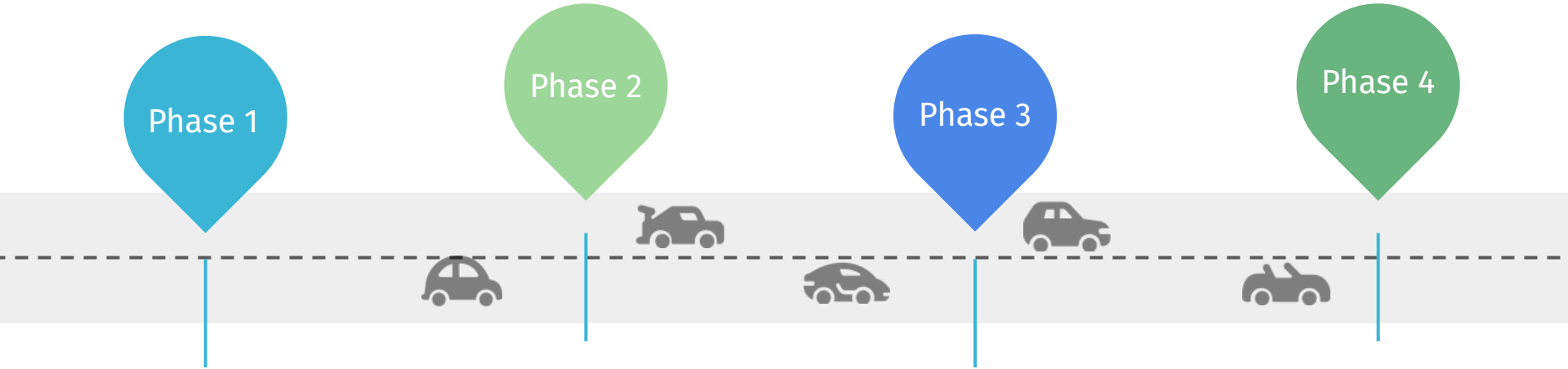
- One Bill to pay
- Single codebase
- Transparent, interconnected networking
- Single Pane of Glass
- Accepts helm-charts natively
- Supports K8S API
- Cloud-Independant ‘Managed K8S service’
- Cheapest run-cost for larger, or multiple small environments
- Enterprise-Grade support

However...

- Scary initial costs
- No Open Source - entirely proprietary.



Infrastructure Roadmap: mojaloop-IaC v4



- Resource Scheduling config
- Containerized Management Tools (K8s, WireGuard etc)
- Improved monitoring & logging

- New ingress (Ambassador) -Eliminate HAProxy+API-GW
- Concatenate Clusters
- Initial KeyCloak implementation
- OpenNebula IaCv4

- Managed K8S - initial implementation
- Initial IaC cross-platform structure
- KeyCloak integration with K8S RBAC
- vmWare PoC

- Successful migration of lower-tier customer environments
- Plan in-place for migrating higher tier customer environments
- OSS Release-Ready

Looking Further Forward

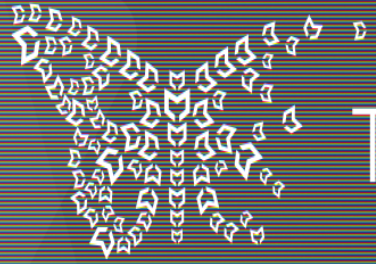
Some suggested next-steps.....(or presentations)

- Kafka deep-dive – swimming in the event stream
  - Performance impacts of stretch vs shard. Hadoop?
  - Topics in low-tier services (e.g. Azure EventHub) : API usage
  - Topic retention tuning (High Availability vs Disaster Recovery)

- Transaction Replication – a story in 3 parts
  - Active/Active/Active – database sharding and performance
  - Eventual Consistency?

- Geo-Replication for resilience - A disaster waiting to happen?
  - Understanding & mitigating transactional loss at planet Scale.

- Kubernetes Operators - Automating for success



Thank You

Anti-Affinity Rules - example

Database Pod helm except

```
metadata:
  name: database-pod
  labels:
    security: db1
spec:
  containers:
  - name: ocp
    image: docker.io/ocpqe/database-pod
```

API Pod hem excerpt

```
metadata:
  name: api-pod
  affinity:
    podAntiAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: security
            operator: In
            values:
            - db1
        topologyKey: kubernetes.io/hostname
  containers:
  - name: pod-antiaffinity
    image: docker.io/ocpqe/api-pod
```

Scheduler will PREFER to keep these two pods separate as long as adequate resources are available to do so. Should no further nodes be available, or nodes lost, pods may be forced together.