Building an on-premise, multitenant serverless platform

Murugappan Chetty
Principal Engineer, Optum

Scale 18x

March 6th, 2020

OPTUM®

About Me

Murugappan Chetty

Serverless, Kubernetes, ISTIO, Opensource contributor, Federated monitoring

itsmurugappan

itsmurugappan

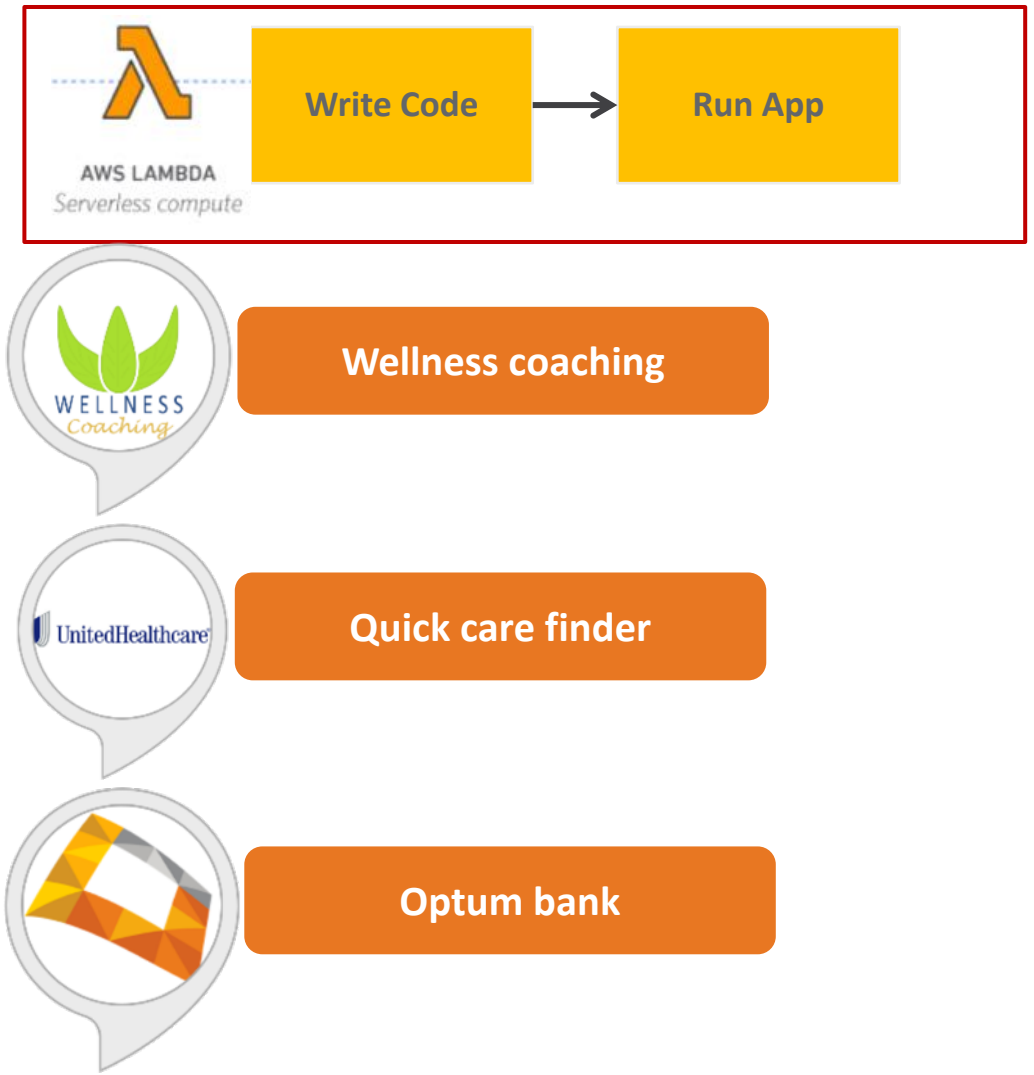
OPTUM®

Agenda

- Intro
- Platform Details
- Platform Management
- Use Cases
- Challenges
- Demo
- Q & A

OPTUM®

Why Serverless ?



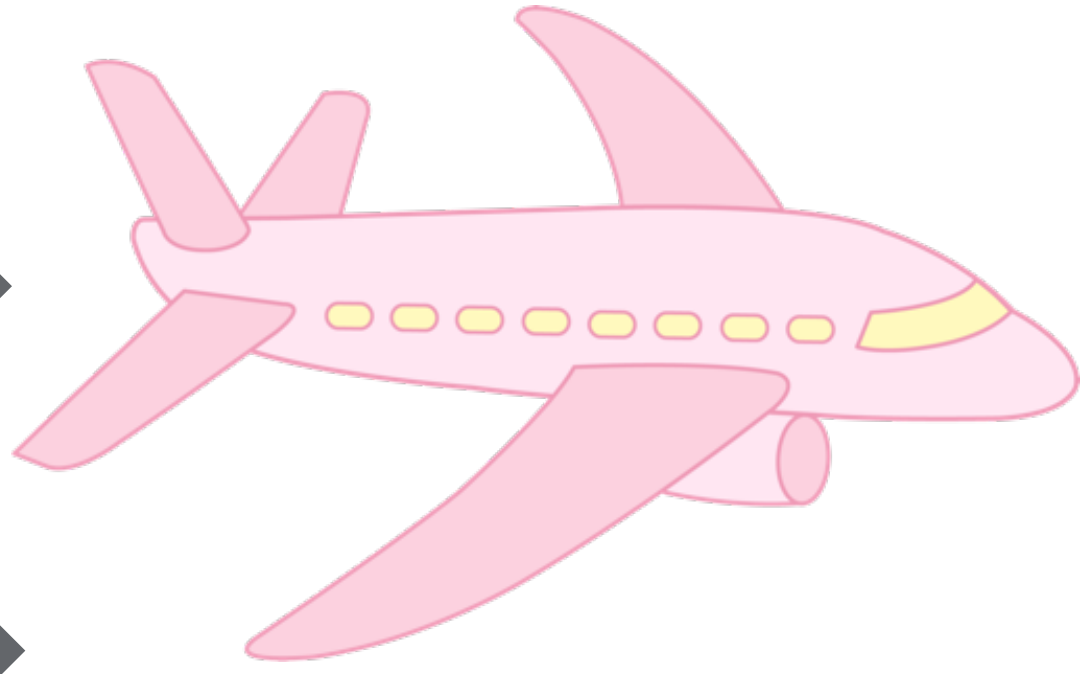
OPTUM®

Head and Tail Winds

Major on-premise workload

Mature container ecosystem

Compelling use cases



Comparison with public offerings

Cost effectiveness

OPTUM®

Platform Details

OPTUM®

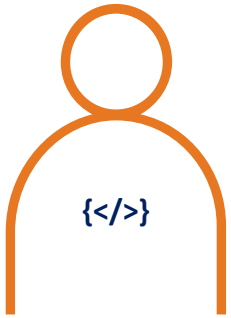
Serverless Platform Principles

- Scale to 0, Request based compute
- Container/kubernetes based
- Support all programming languages (Java, Go, Python, Shell ..)
- Low barrier of entry
- Observability
- Live and Learn!

Personas

Operators

- ✓ Provision serverless platform and provides the URL to deploy code
- ✓ Fully managed compute – provisioning, patching, scaling, monitoring, logging are provided by Ops team
- ✓ Abstraction of servers away from the developer ex: K8s, istio



Developers

- ✓Focus on business logic; Write functions (GO, python, PHP, node.js, ruby, java , .net etc.,)
- ✓Define function config and deploy functions
- ✓Invoke the function URL ex: Code to URL
- ✓Forget servers!

OPTUM®

Platform Capabilities

MULTITENANCY

- Kubernetes Namespaces
- RBAC

SECURITY

- ISTIO Policy
- ISTIO RBAC
- TLS
- Keycloak

OBSERVABILITY

- Prometheus
- Kiali
- Kubernetes logs
- Jaeger/Zipkin

RESILIENCY

- Chaos Engineering
- Selfhealing automation
- Cross data center load balancing

SERVING

- Knative serving
- Function/Service lifecycle

BUILD

- Build Packs
- Openfaas CLI
- Jib / Ko / Fabric8
- Tekton

EVENTING

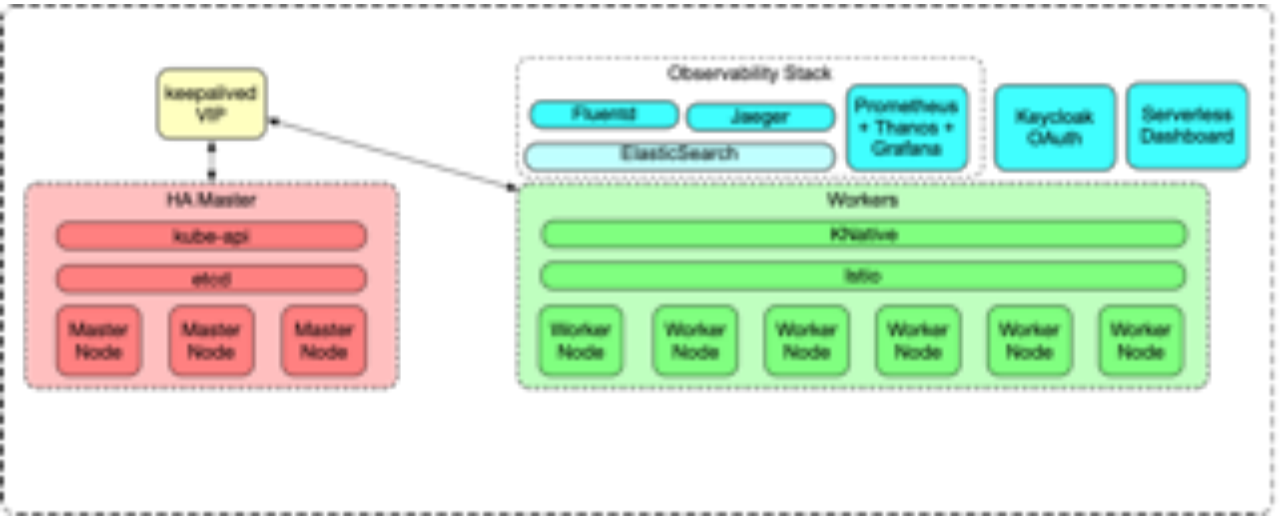
- Knative Eventing
- Cloud Events

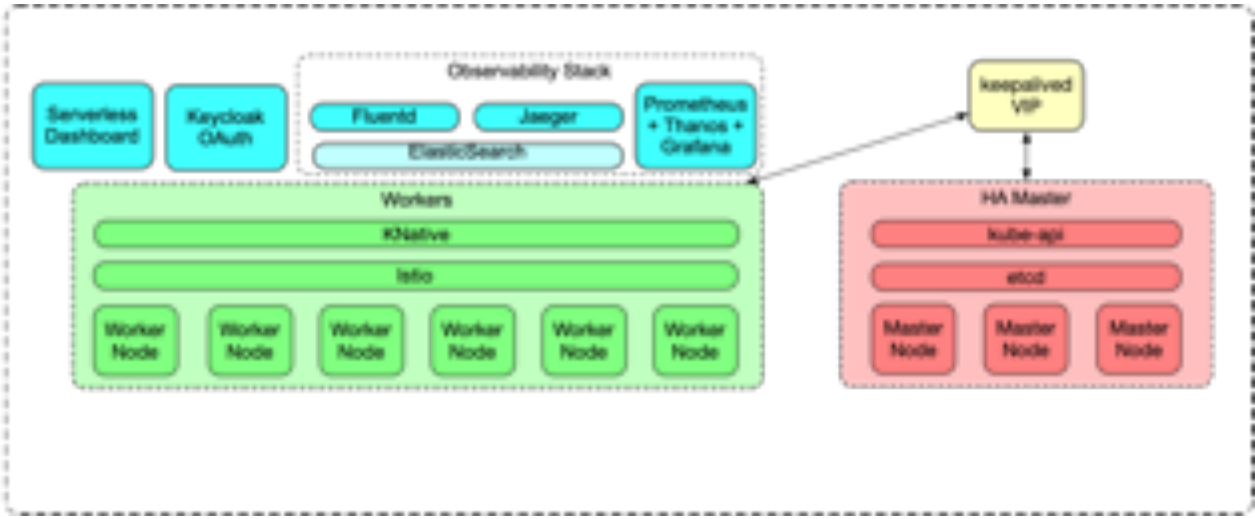
USER AGILITY

- **Inhouse**
- API & Swagger
  - Provision namespaces
  - Function management
  - Apply security policies
- Comprehensive guides
- KN cli

OPTUM®

Platform Components





OPTUM®

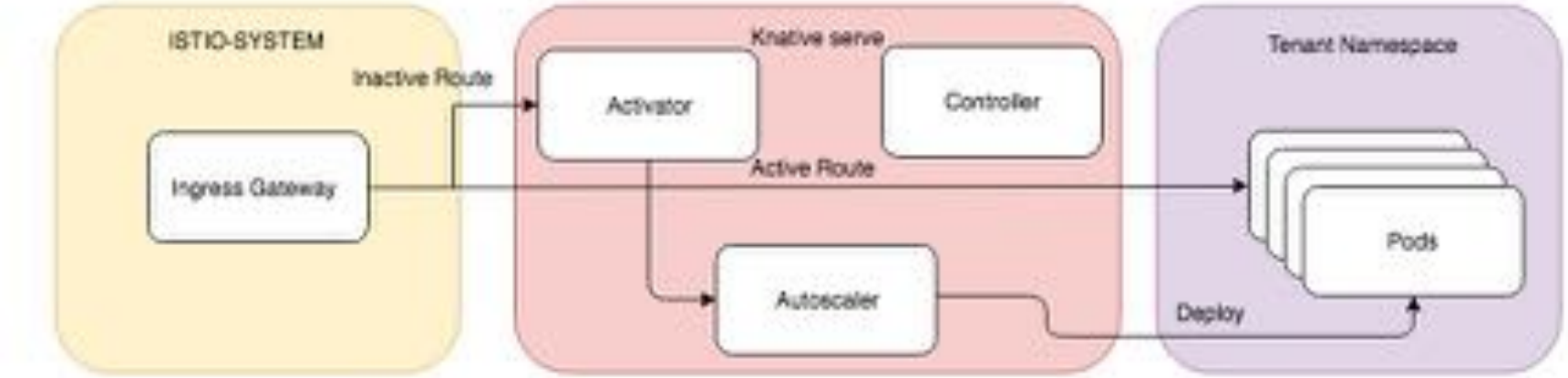
Kubernetes

What does it take to deploy a service today ?

- Need to write 2 manifests - deployment & service
- No per-request load balancing
- No traffic splitting
- Auto scaling limitations
- No concurrency control

Knative

- Opensource – 400+ contributors, 60+ companies (Google, VMWare, IBM, Redhat, SAP, Pivotal …)
- Serving, Eventing
- Multitenant
- Main Components – Activator, Autoscaler, Controller, Webhook and Queue proxy sidecar
- Istio/Gloo for networking
- Only activator in the request path for initial calls



OPTUM®

Knative – Serving Resources

Configuration

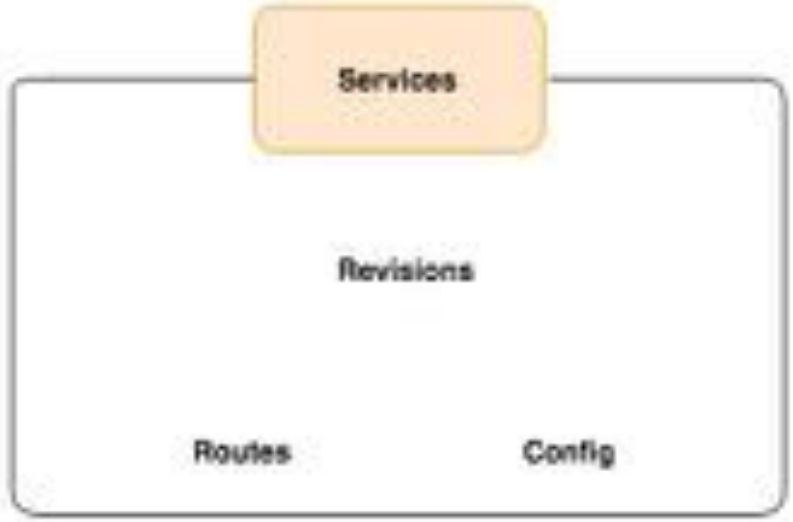
Current desired configuration

Revision

Immutable object. Point in time for code and snapshots

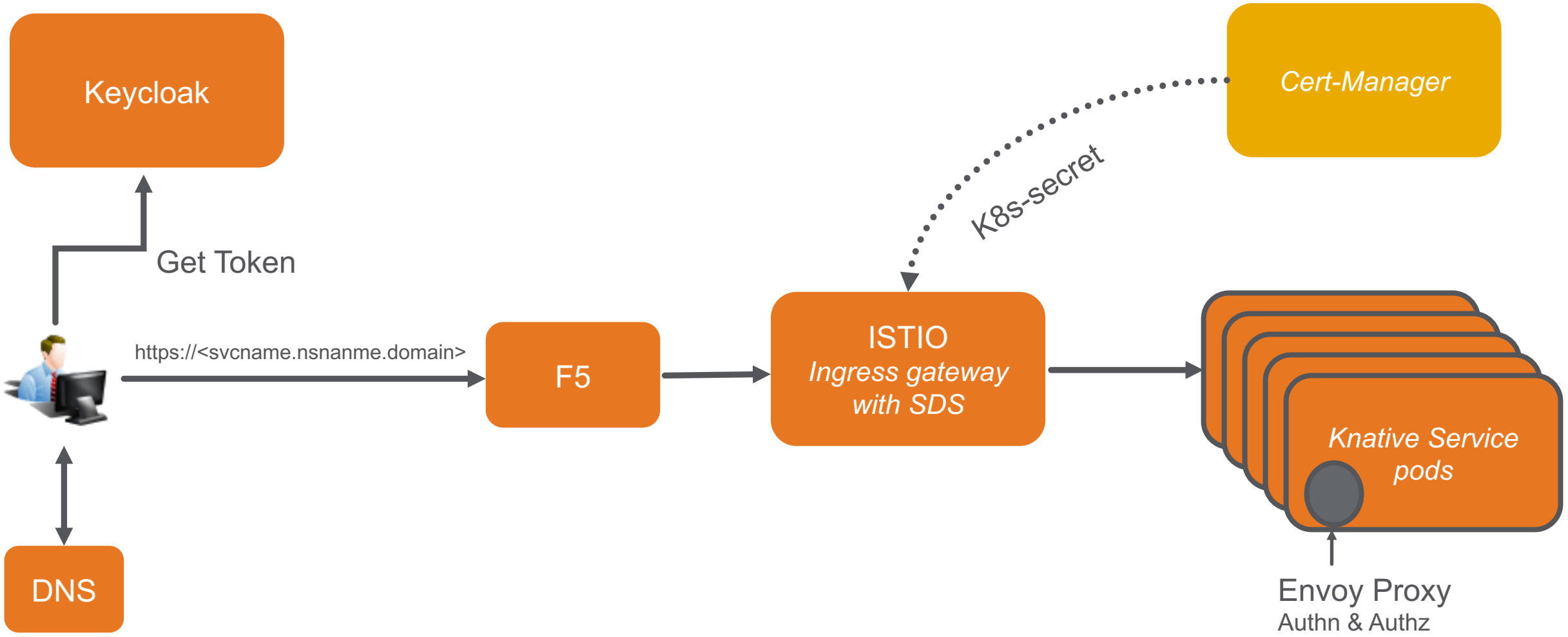
Route

Maps traffic to revisions



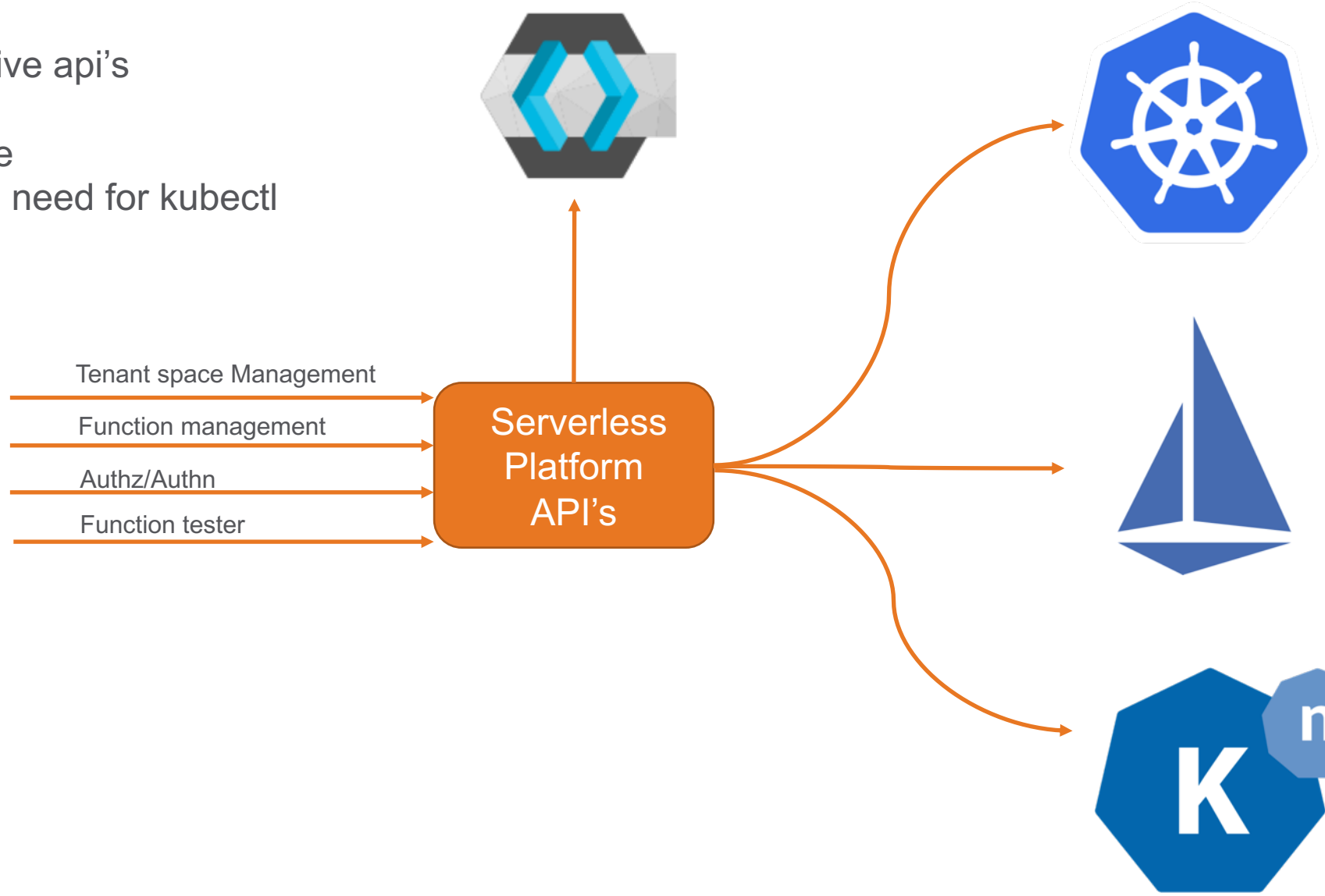
OPTUM®

ISTIO



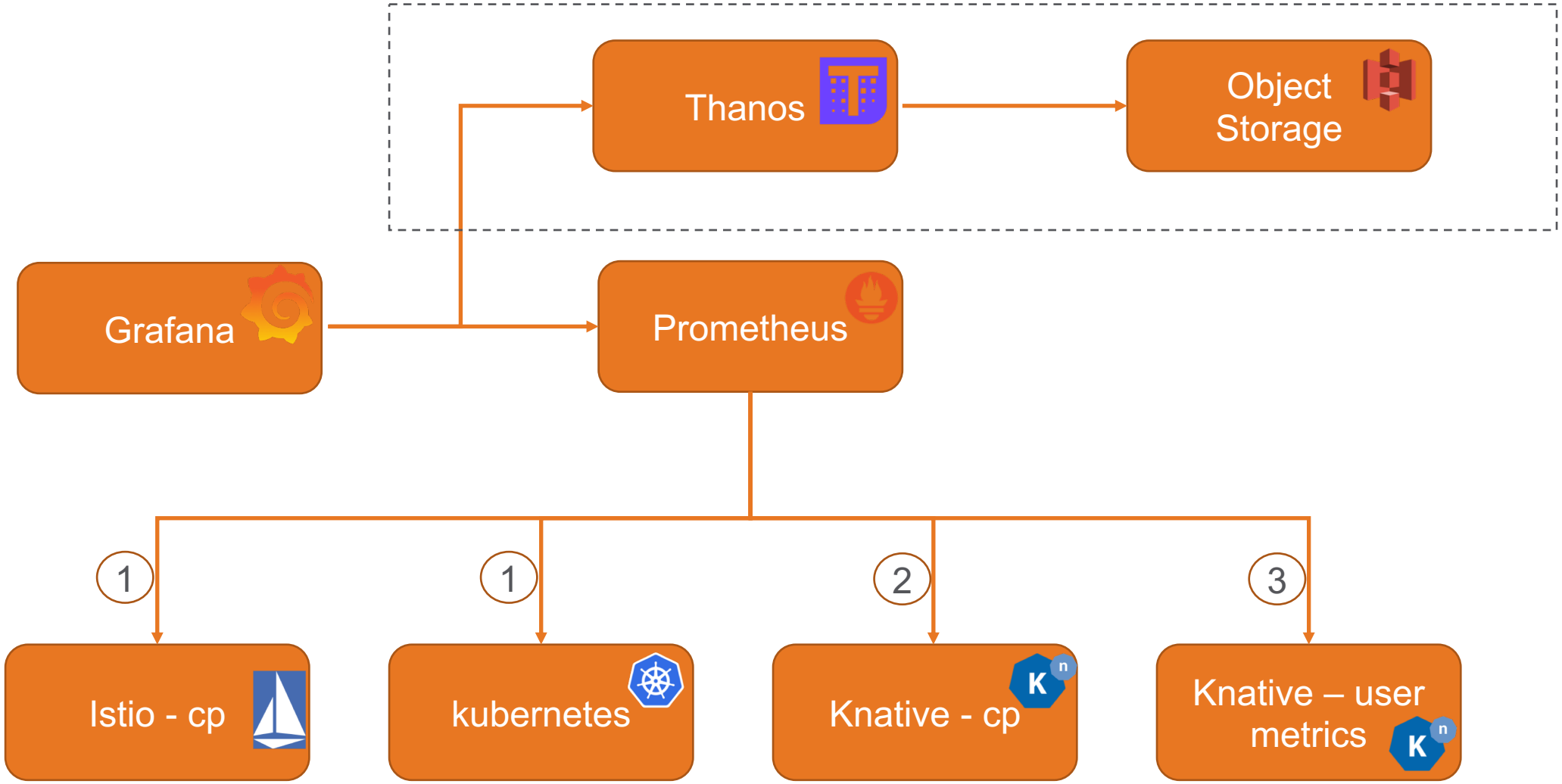
Service Deployment made easy

1. Abstracts k8s, istio and knative api's
2. Opinionated
3. Unified developer experience
4. **Self service** - eliminates the need for kubectl and other cli's
5. **Enforces standards**
    1. Resource restriction
    2. Run as non root user
    3. Function Versioning
6. CICD and Automation



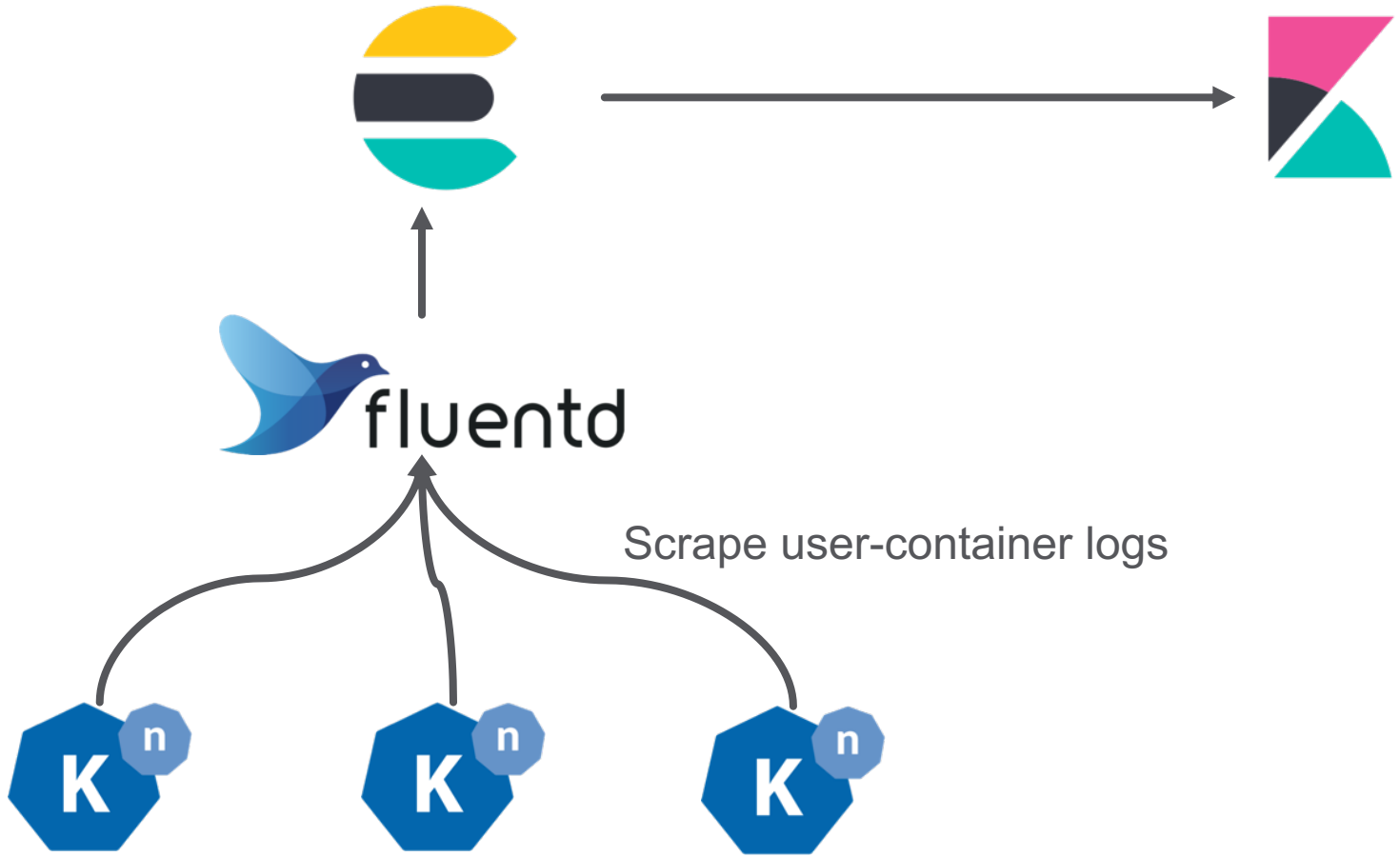
OPTUM®

Metrics



OPTUM®

Logs - EFK

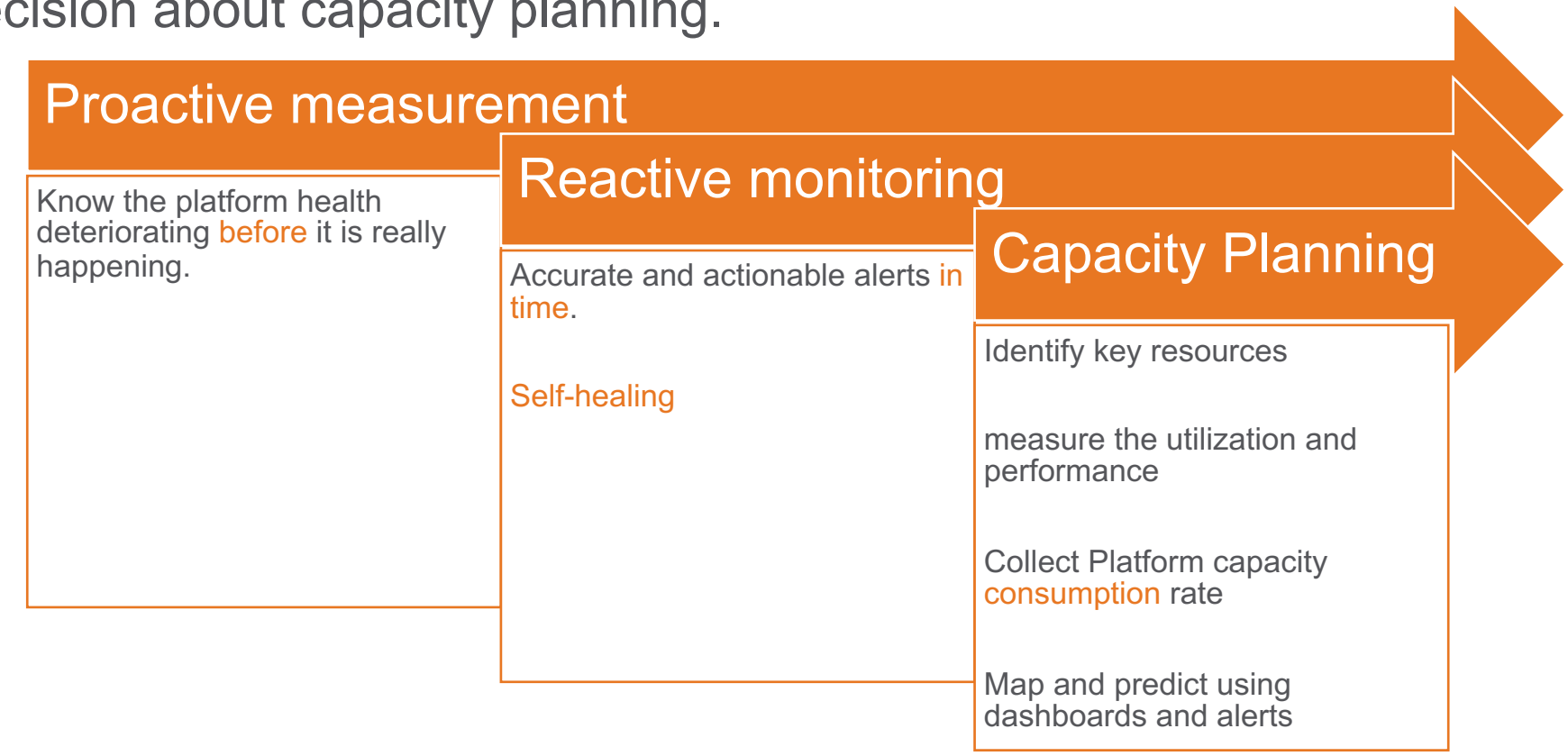


Platform Management

OPTUM®

Operator: Cluster Health and Capacity Planning

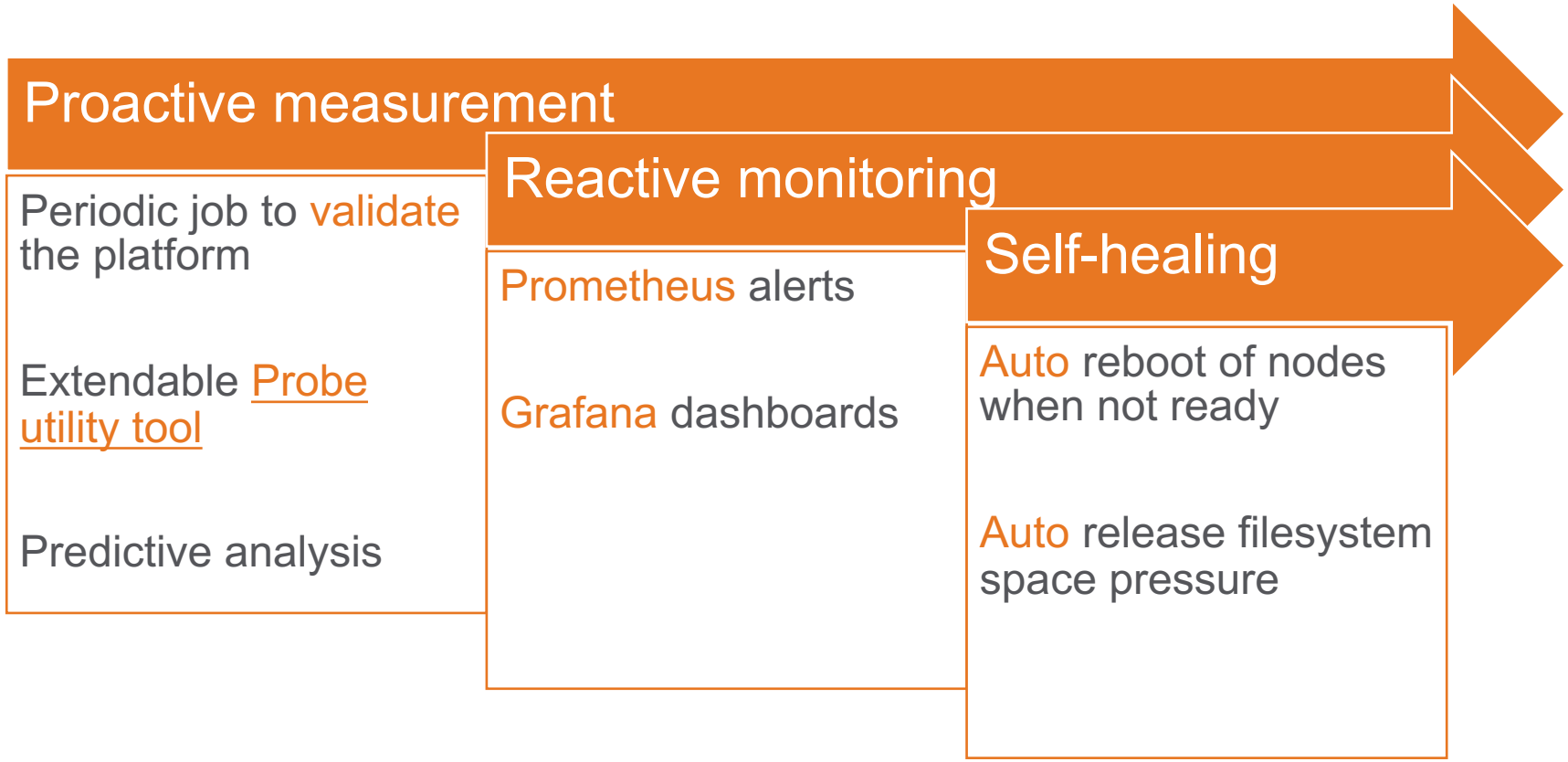
As an Operator, I want to know the resource consumption of the cluster to make fact based decision about capacity planning.



OPTUM®

Operator: Cluster Monitoring and Self healing

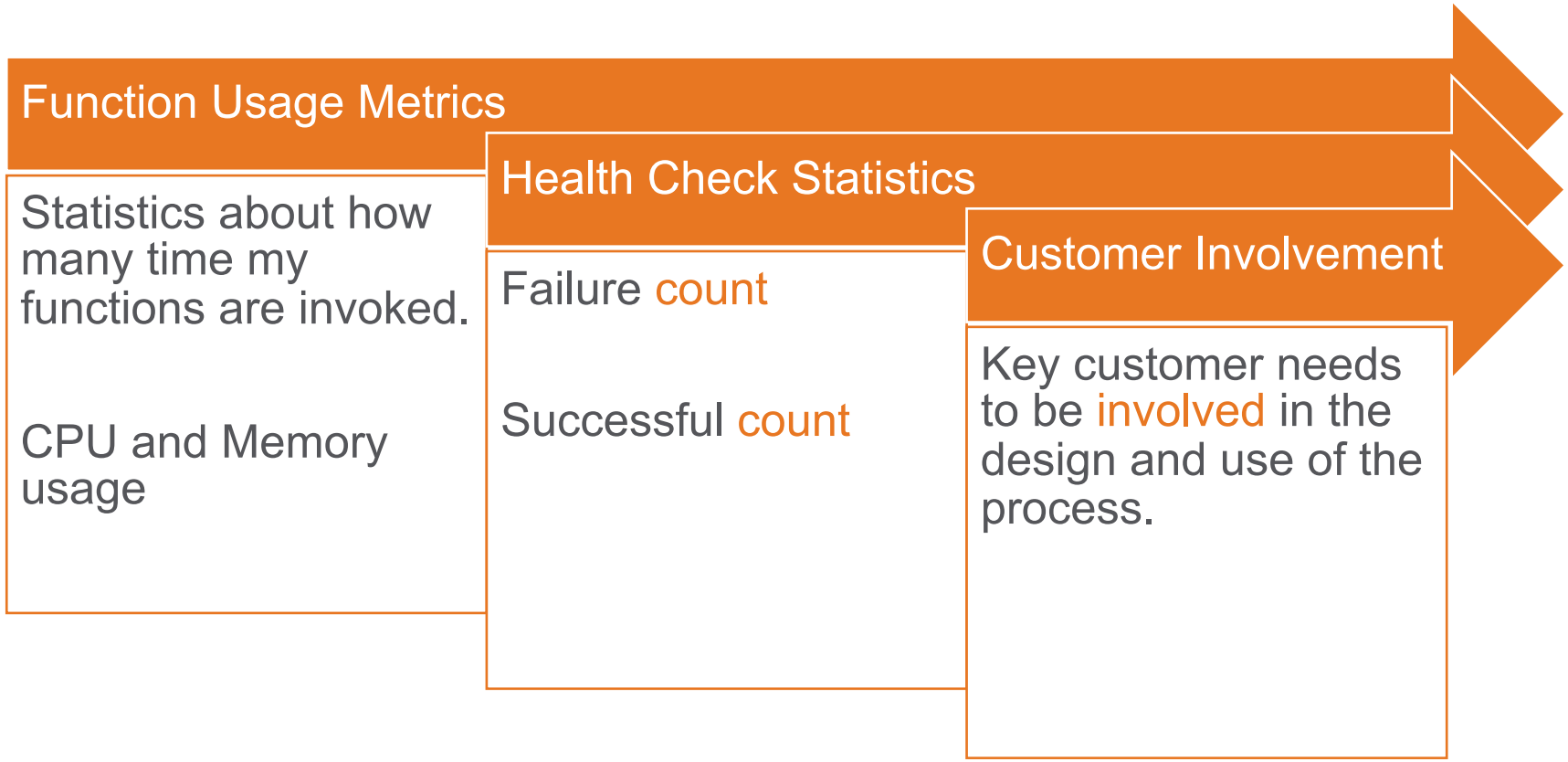
As an operator, I want to ensure the platform is highly available, reliable, and serviceable



OPTUM®

Developers: Users of this platform

As a developer, I want to see my function metrics, and health check statistics.



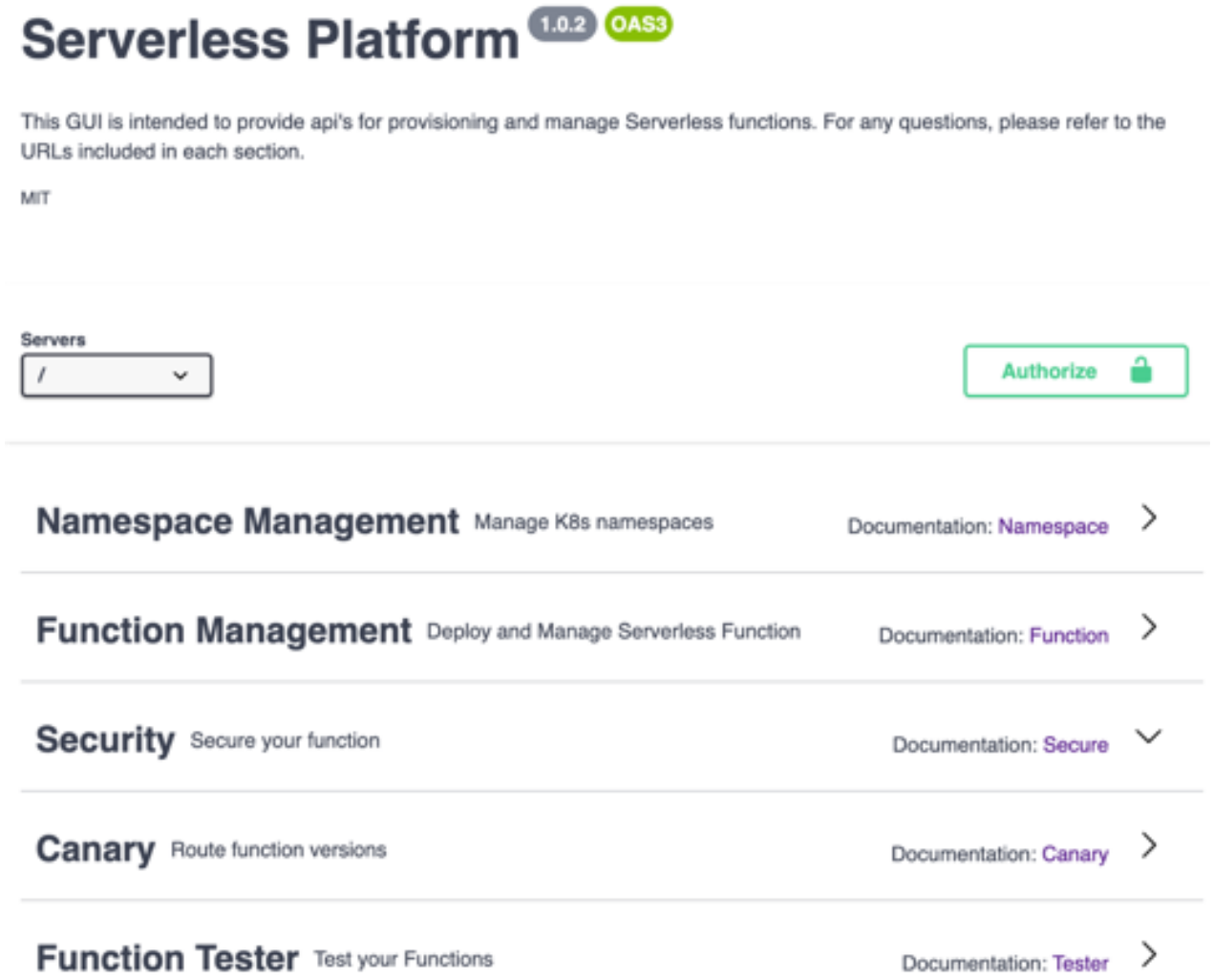
OPTUM®

User Agility

OPTUM®

User Agility – Build & Deploy

- Build Packs
- Tekton pipelines
- Ko/Jib/Fabric8
- Openfaas CLI
- Inhouse Serverless Platform API’s
- Kn cli



OPTUM®

User Agility - Observe

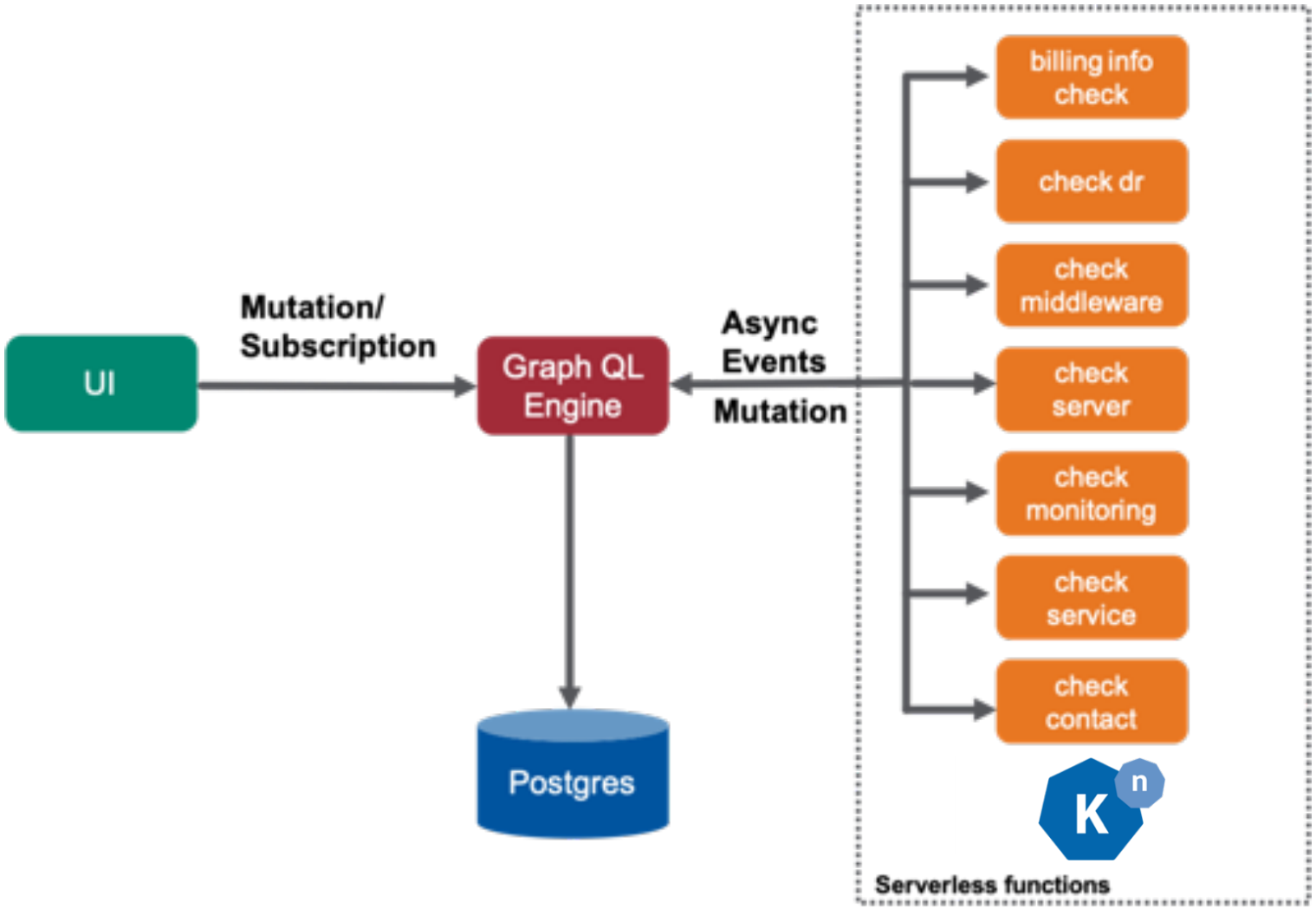
- Grafana User Dashboards
- Logs – Kibana, Kail

OPTUM®

Use Cases

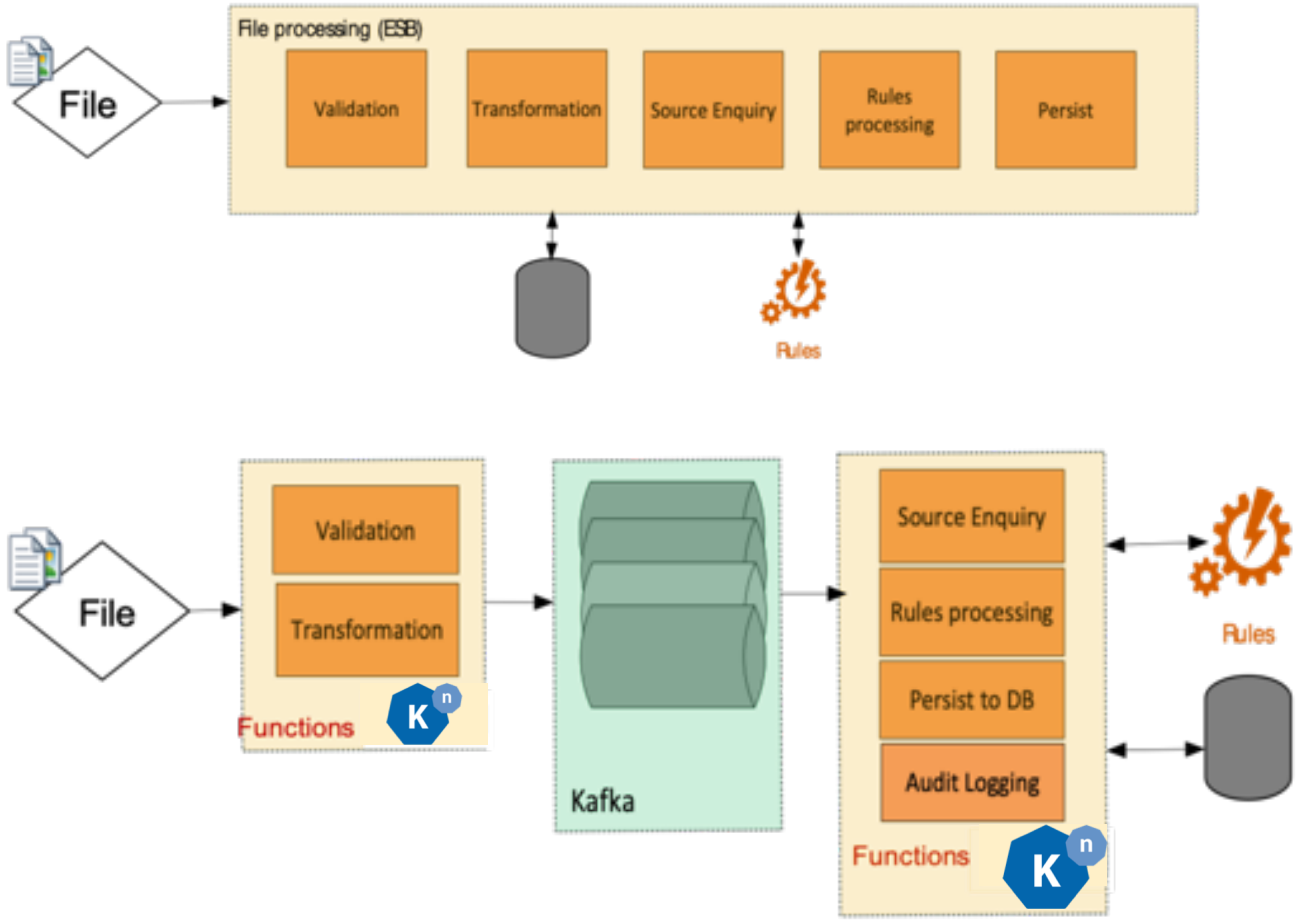
OPTUM®

Infrastructure Automation



OPTUM®

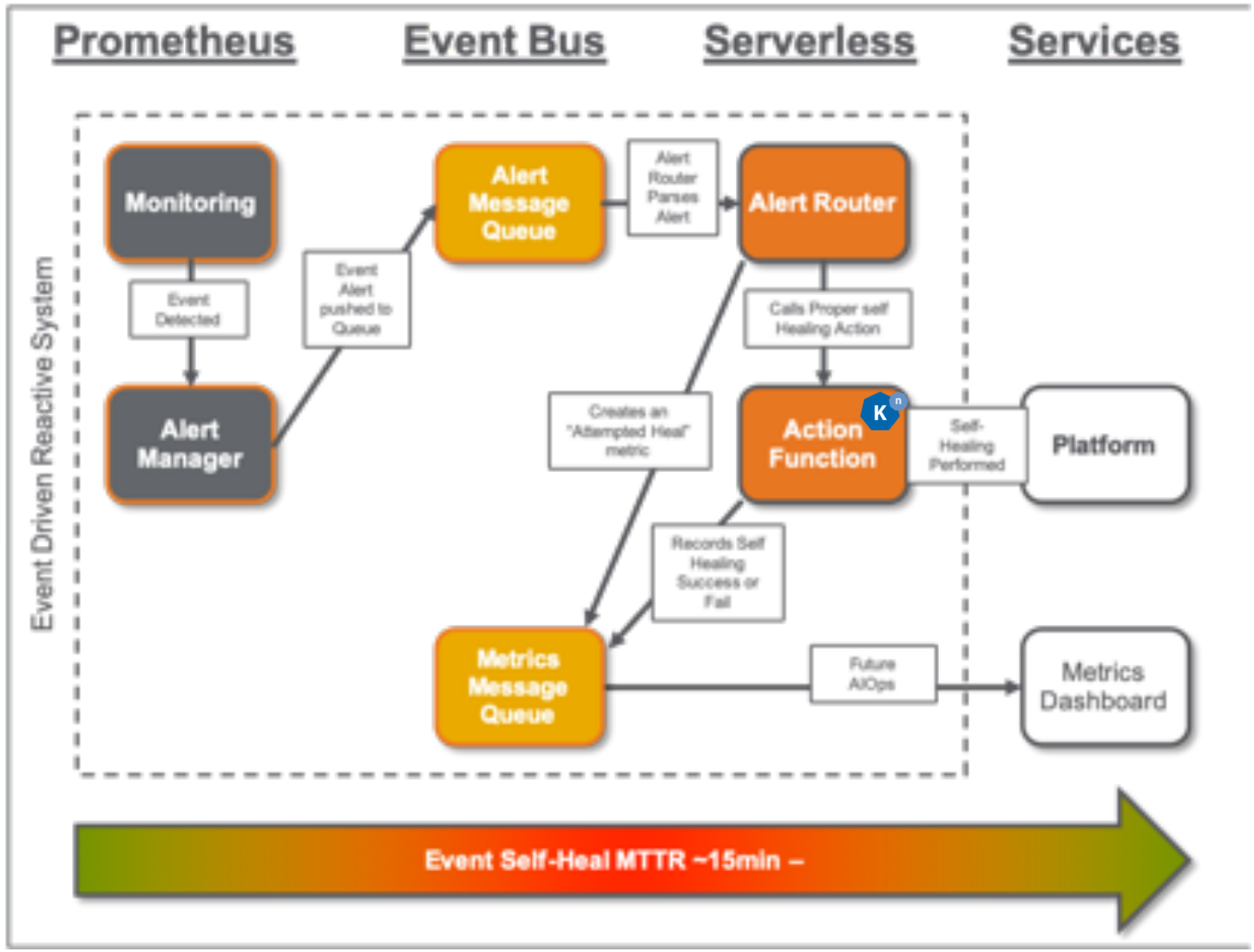
File Processing – ESB Vs Serverless



OPTUM®

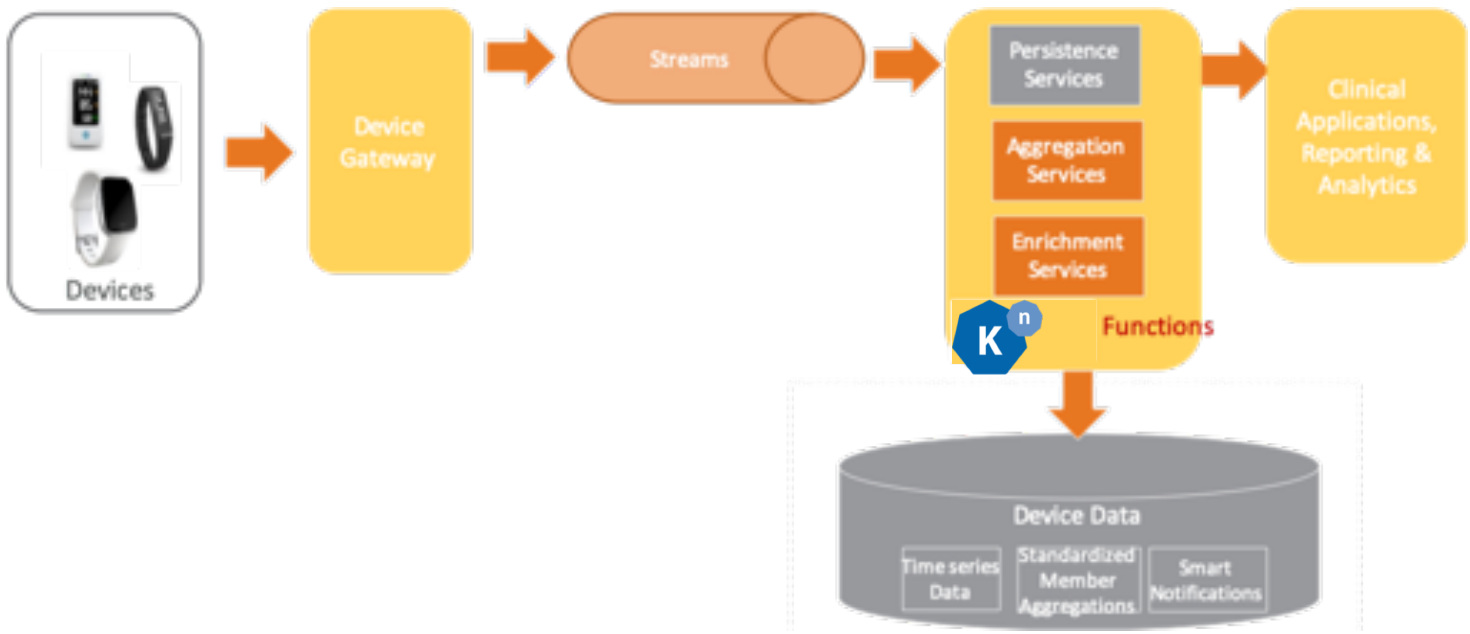
Self Healing

Standardized and Repeatable platform for self healing activities.



OPTUM®

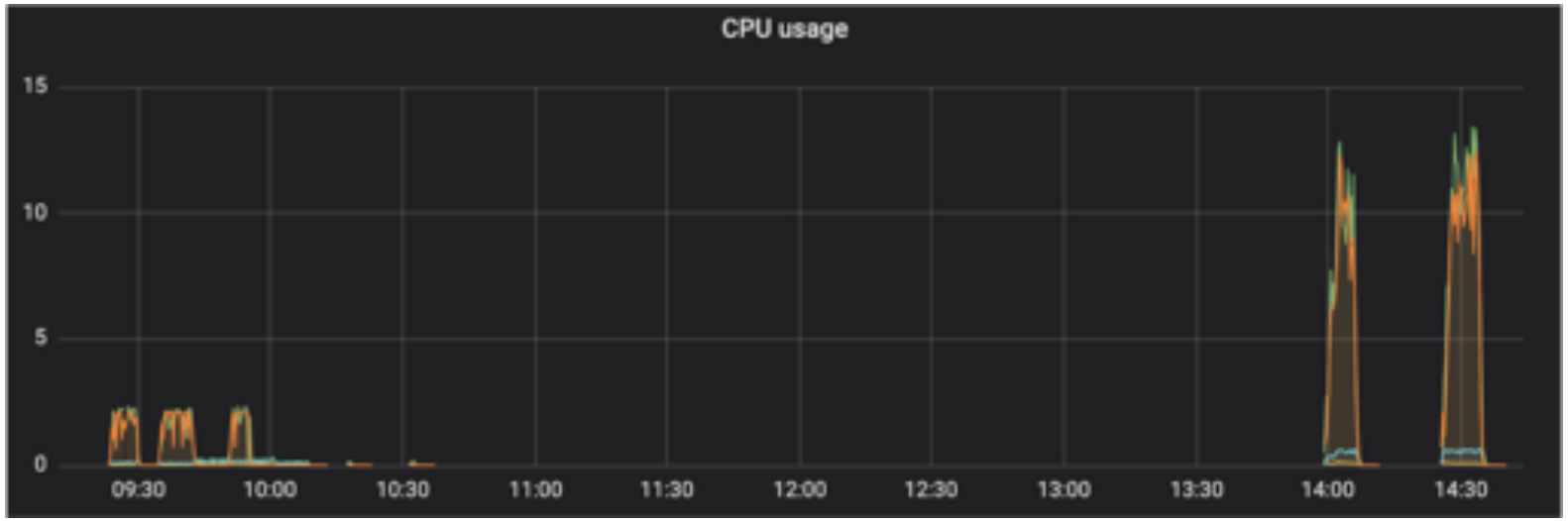
IOT


Devices
Device Gateway
Streams
Persistence Services
Aggregation Services
Enrichment Services
Functions
Clinical Applications, Reporting & Analytics
Device Data
Time series Data
Standardized Member Aggregations
Smart Notifications

OPTUM®

ML UseCase - Benchmark


```
{ "transactions":                 51,
  "availability":                 100.00,
  "elapsed_time":                 299.17,
 "successful_transactions":       51,
  "failed_transactions":          0,
  "longest_transaction":          134.38,
  "shortest_transaction":         77.02
}
```


```
{ "transactions":                 383,
  "availability":                 100.00,
  "elapsed_time":                 412.81,
"response_time":                  25.49,
"successful_transactions":        383,
  "failed_transactions":          0,
  "longest_transaction":          114.94,
  "shortest_transaction":         4.41
}
```


OPTUM®

Other Use Cases

Use Cases
Infrastructure Team API's – Server info, network config etc
ETL Jobs
Voicemail processing
Serving ML models

Challenges

OPTUM®

Challenges

Challenges	Solution
DB Connections - No connection pooling	Dedicated microservice for handling DB connections. GraphQL engine for data persistence and retrieval
Default resource allocation for pods	Enforce users to set resources.
Cold starts	Mitigate cold starts
Long running functions	longer timeouts, microservices
Java functions	Graal VM’s

Using Java on Serverless Platform

Poor Start Up & High Memory

```
SpringCassandraApplication in 21.066 seconds
Used: 269,075,552 B
```

Ahead-Of-Time (AOT) & Native Image

```
serverless-quarkus-knative-java-ms Quarkus 1.1.1.Final) started in 0.412s.
IMAGE                                MEM USAGE / LIMIT     MEM %
serverless-quarkus-knative-java-ms   21.92MiB / 7.769GiB   0.28%
```

- Quarkus: Kubernetes Native Java framework tailored for GraalVM and HotSpot, crafted from best-of-breed Java libraries and standards.
- Developed by RedHat with the goal to make Java a leading platform in Kubernetes & Serverless
- Designed to have “Supersonic” start up times and low memory footprint
- Quarkus uses a single reactive engine for both imperative & reactive code

OPTUM®

Final Thoughts

Timeout is configurable (can go as long as u want)
HPA doesn’t scale to 0
Cluster Local option
Stateful workloads and PVC’s
Back up
Eventing
kubeflow

OPTUM®

Questions & Answers

OPTUM®

Thank you!

OPTUM®


OPTUM®