WELCOME

From Chaos to Control: Kubernetes Operators in Practice

Automating infrastructure at scale

Thessaloniki Not-Only Java meetup

Hello!

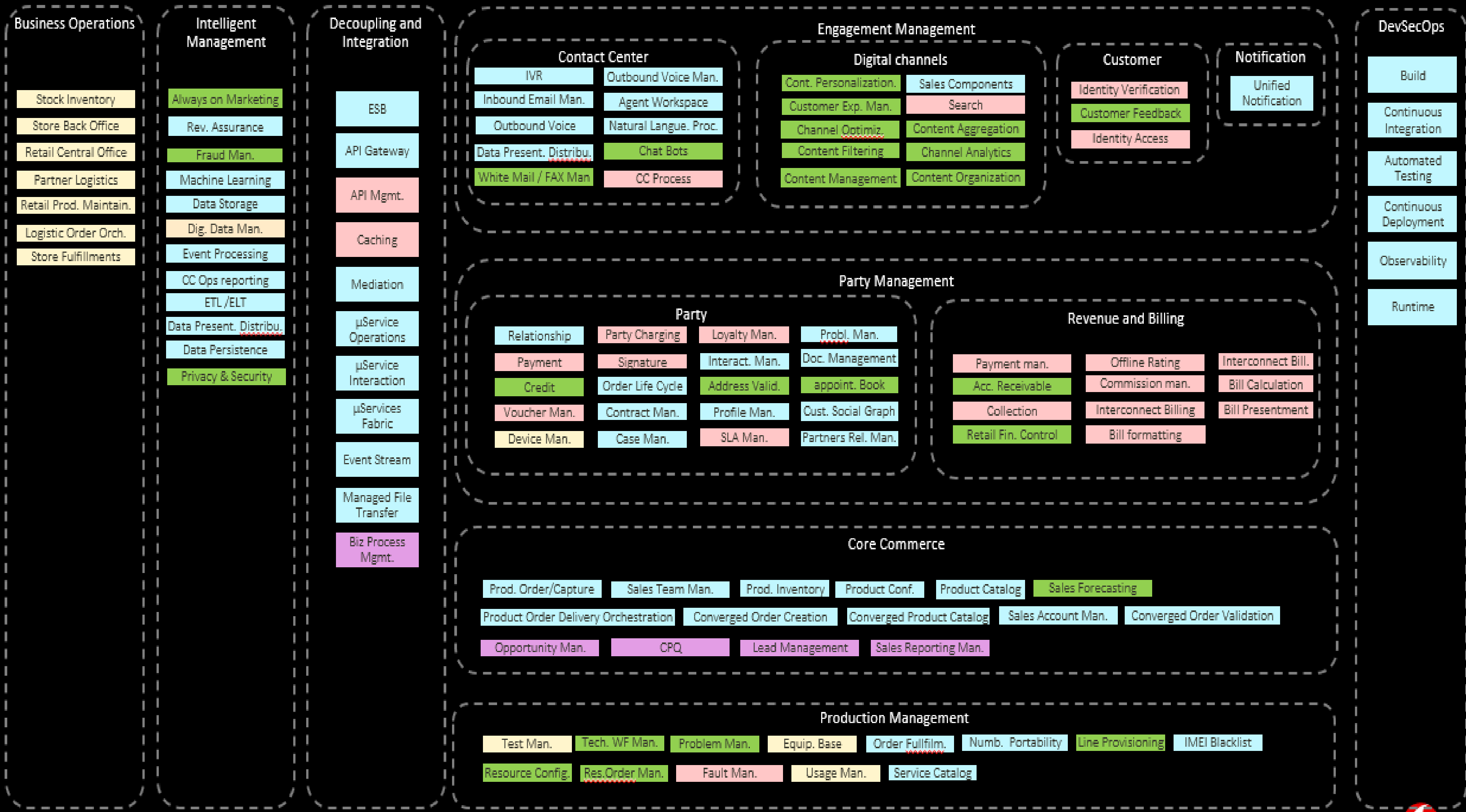
Nikos Ntemkas
Senior API Engineer
Vodafone Greece

Giorgos Binas
API Engineer
Vodafone Greece

Agenda

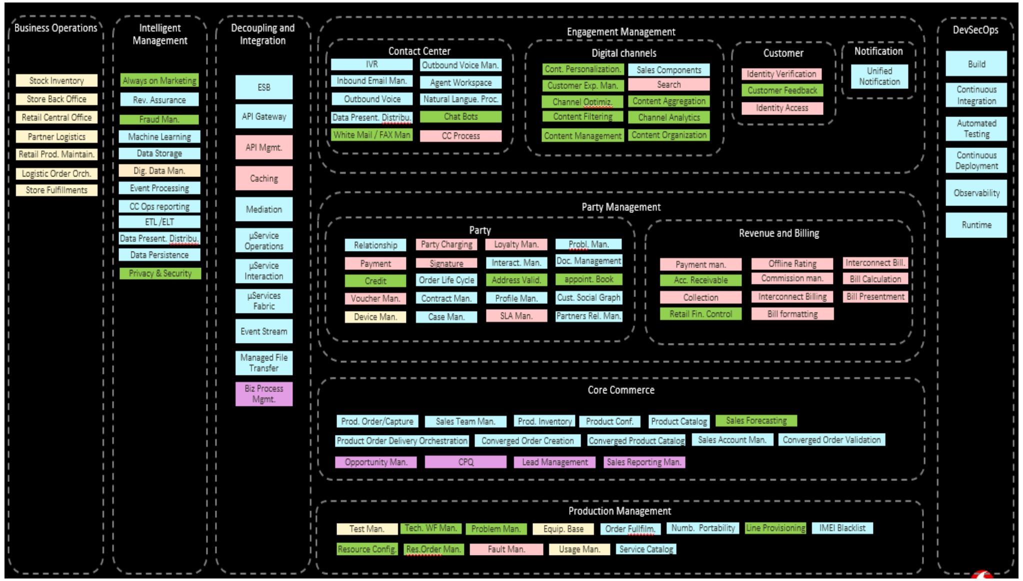
- The Chaos
- The Solution
- Operators in Action
- Building Your Own
- Takeaways

The Chaos


Business Operations
Stock Inventory
Store Back Office
Retail Central Office
Partner Logistics
Retail Prod. Maintain.
Logistic Order Orch.
Store Fulfillments
Intelligent Management
Always on Marketing
Rev. Assurance
Fraud Man.
Machine Learning
Data Storage
Dig. Data Man.
Event Processing
CC Ops reporting
ETL /ELT
Data Present. Distribu.
Data Persistence
Privacy & Security
Decoupling and Integration
ESB
API Gateway
API Mgmt.
Caching
Mediation
µService Operations
µService Interaction
µServices Fabric
Event Stream
Managed File Transfer
Biz Process Mgmt.
Engagement Management
Contact Center
IVR
Outbound Voice Man.
Inbound Email Man.
Agent Workspace
Outbound Voice
Natural Langue. Proc.
Data Present. Distribu.
Chat Bots
White Mail / FAX Man
CC Process
Digital channels
Cont. Personalization.
Sales Components
Customer Exp. Man.
Search
Channel Optimiz.
Content Aggregation
Content Filtering
Channel Analytics
Content Management
Content Organization
Customer
Identity Verification
Customer Feedback
Identity Access
Notification
Unified Notification
DevSecOps
Build
Continuous Integration
Automated Testing
Continuous Deployment
Observability
Runtime
Party Management
Party
Relationship
Party Charging
Loyalty Man.
Probl. Man.
Payment
Signature
Interact. Man.
Doc. Management
Credit
Order Life Cycle
Address Valid.
appoint. Book
Voucher Man.
Contract Man.
Profile Man.
Cust. Social Graph
Device Man.
Case Man.
SLA Man.
Partners Rel. Man.
Revenue and Billing
Payment man.
Offline Rating
Interconnect Bill.
Acc. Receivable
Commission man.
Bill Calculation
Collection
Interconnect Billing
Bill Presentment
Retail Fin. Control
Bill formatting
Core Commerce
Prod. Order/Capture
Sales Team Man.
Prod. Inventory
Product Conf.
Product Catalog
Sales Forecasting
Product Order Delivery Orchestration
Converged Order Creation
Converged Product Catalog
Sales Account Man.
Converged Order Validation
Opportunity Man.
CPQ
Lead Management
Sales Reporting Man.
Production Management
Test Man.
Tech. WF Man.
Problem Man.
Equip. Base
Order Fullfilm.
Numb. Portability
Line Provisioning
IMEI Blacklist
Resource Config.
Res.Order Man.
Fault Man.
Usage Man.
Service Catalog

Modern Enterprise Challenges

- Vodafone is a large Enterprise Organization with millions of customers
- Vodafone's stack is affected by mergers / acquisitions
  - Hundreds of systems, breed of modern / legacy
- Big History – Big Legacy
  - Master Systems-Complex
  - More than 150 deployed systems (including those used by physical Vodafone stores)
- How do you change this type of landscape?
  - Increased stability requirements (nowadays, everything should be near real-time)



Establishing Order

Transforming the Landscape

Requirements

- Gradual Modernization
- Decouple Teams
- Standardization
- Portability
- Deployment orchestration, monitoring

Reality Check

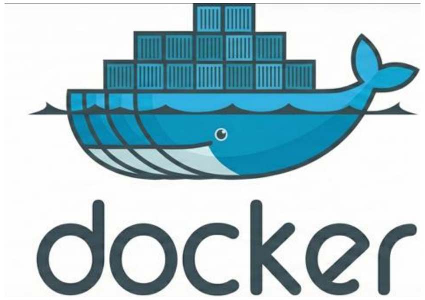
What People Think They Need

- Split the monolith
- Add Docker
- Deploy to Kubernetes

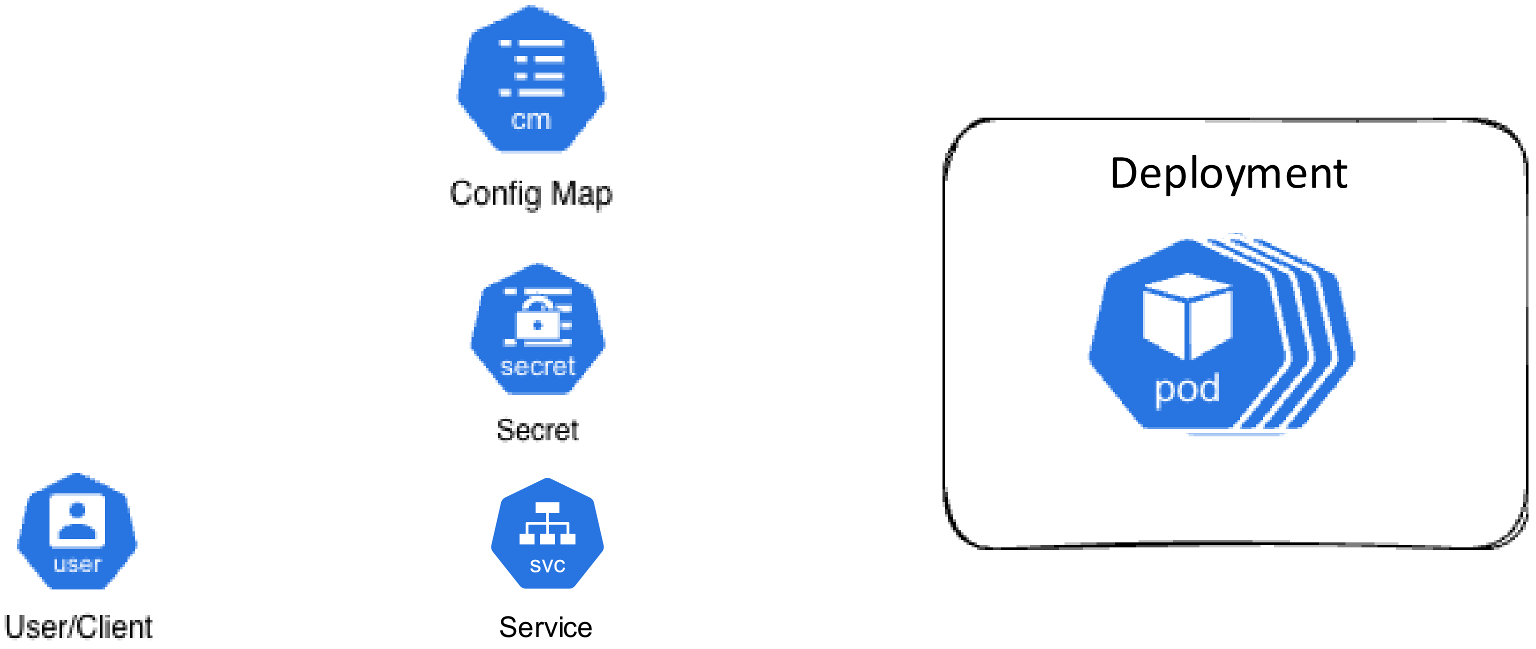
What You Actually Need

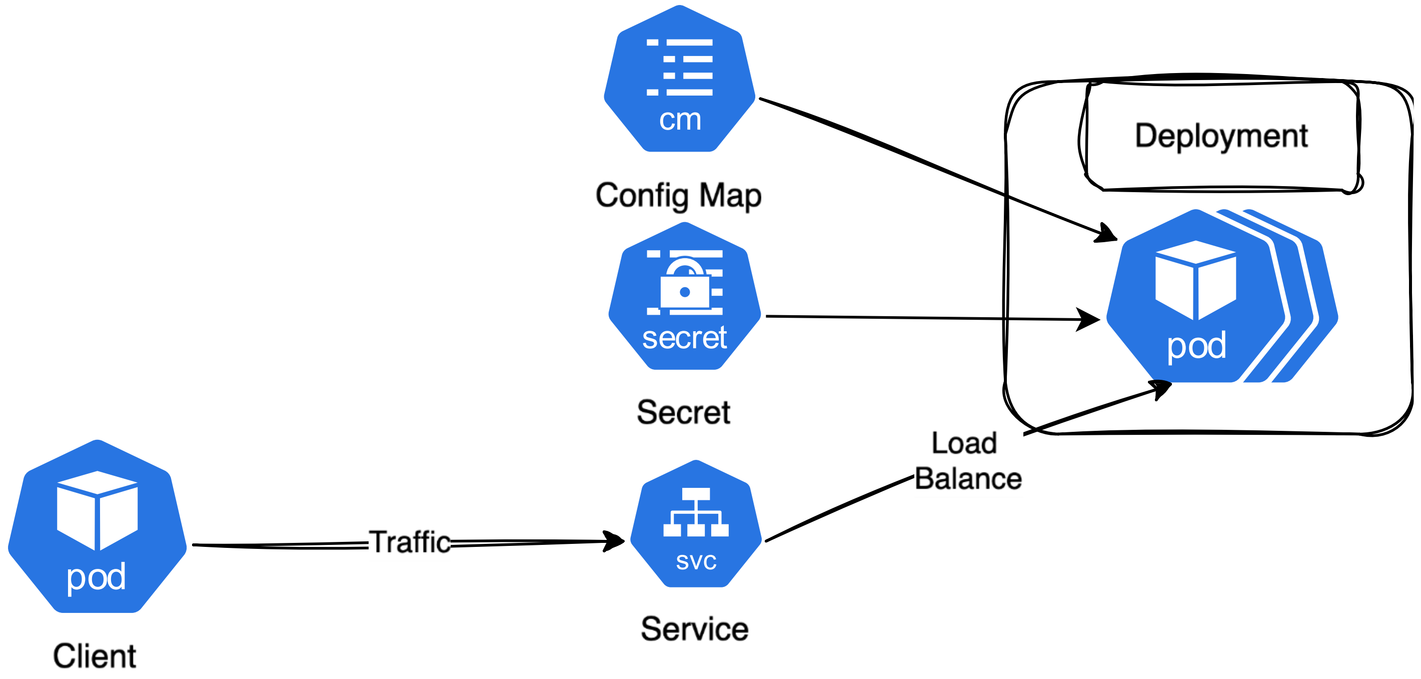
- API Gateway and Routing Layer
- Configuration Management
- Secret Management
- Inter-Service Communication
- Schema and Contract Versioning
- Service Registry and Health Checks
- Circuit Breakers and Rate Limiting
- Retry with Backoff and Idempotency
- Observability Stack
- Distributed Tracing Context Propagation
- Event Sourcing / Outbox Pattern
- Message Deduplication and Ordering
- Load Balancing and Auto-Scaling
- API Versioning and Deprecation Policy
- Data Consistency and Sagas
- Service Discovery
- Dead Letter Queues and Poison Message Handling
- Cost and Latency Profiling

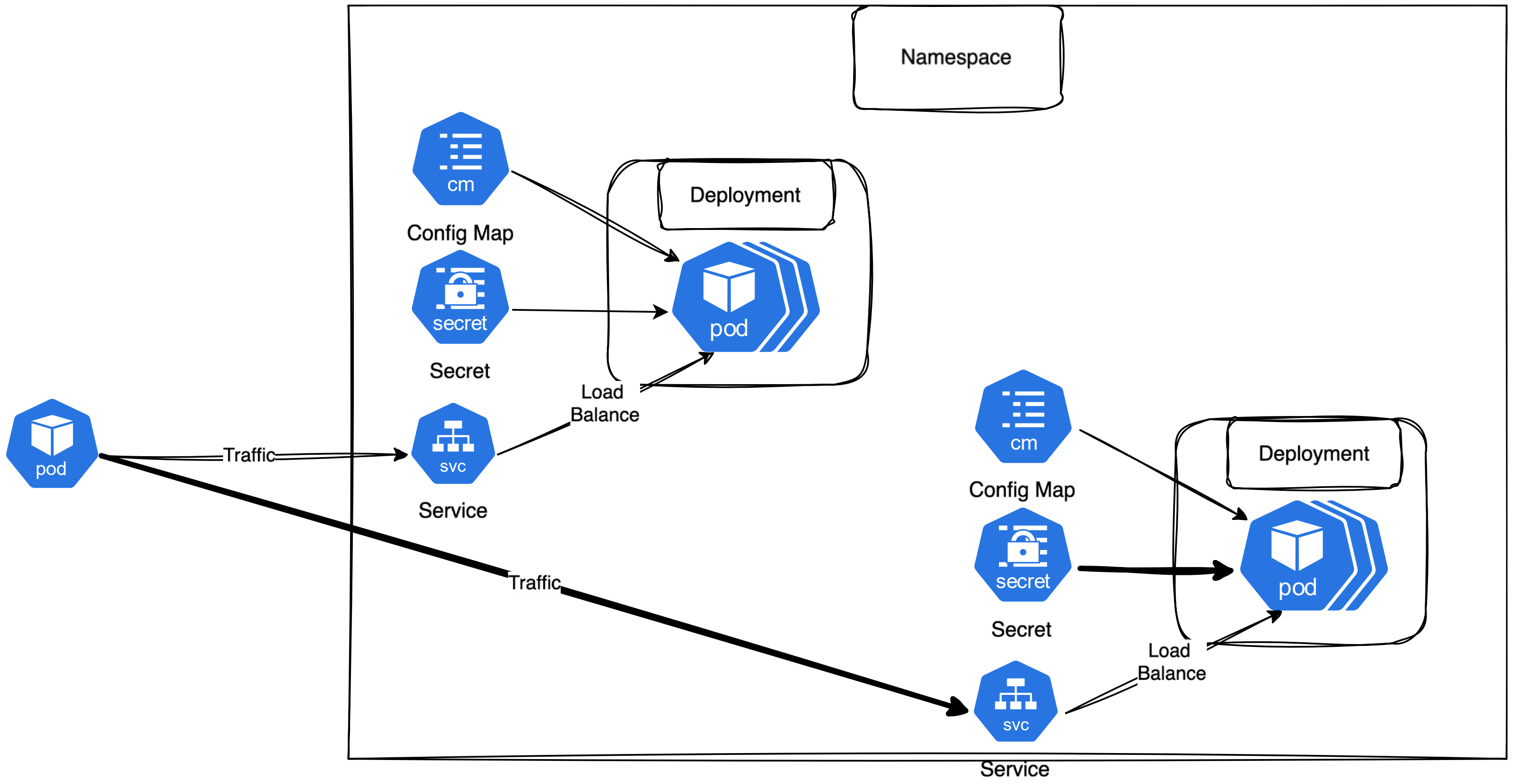
Pod

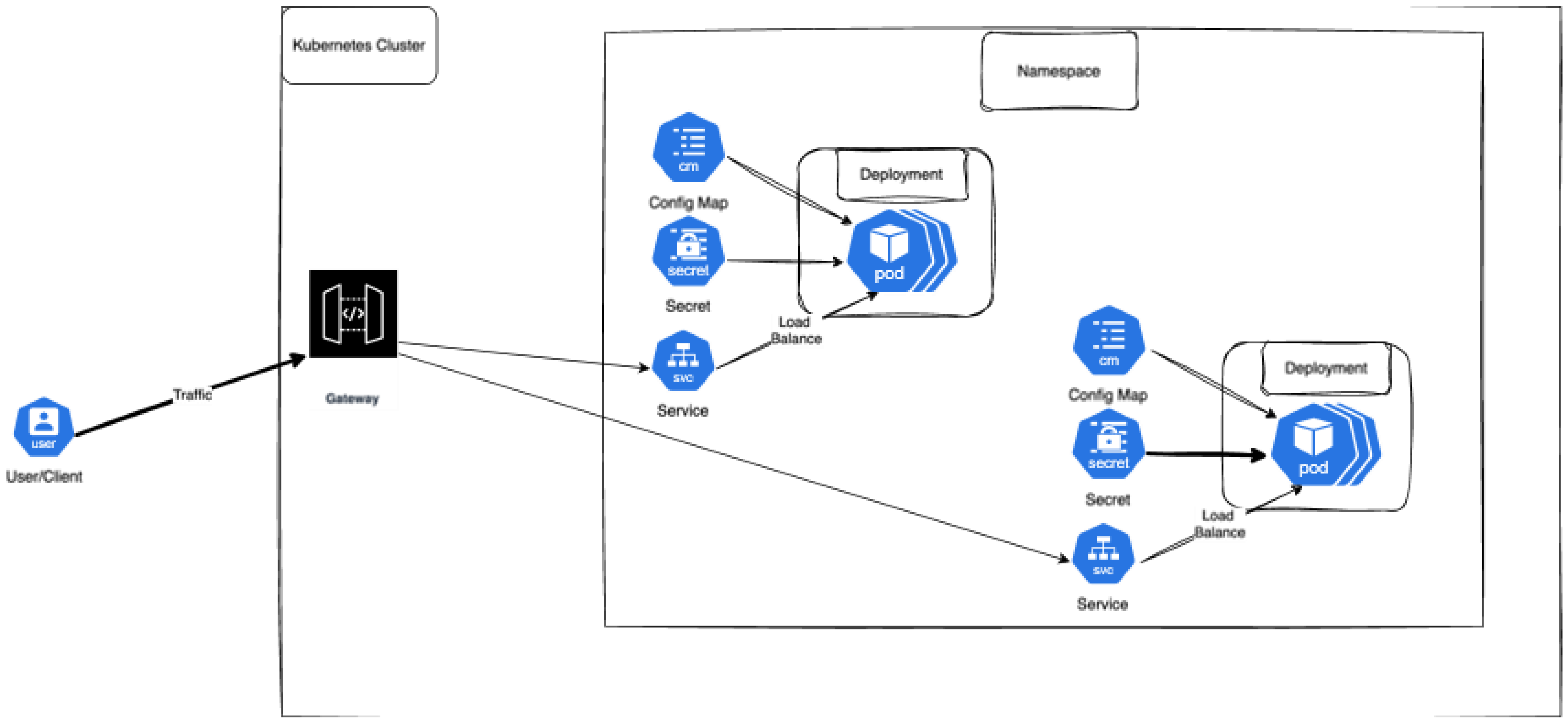


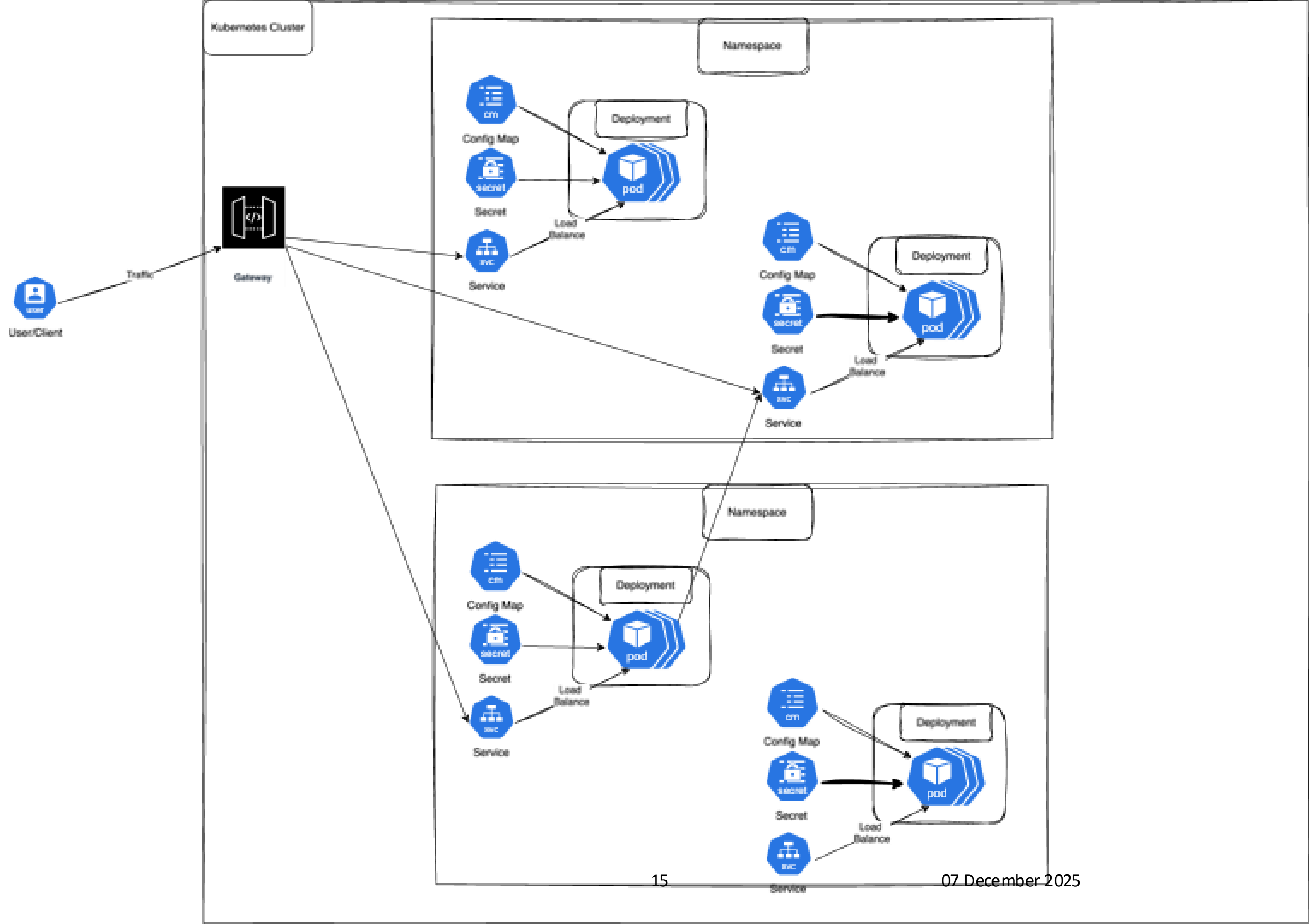
07 December 2025

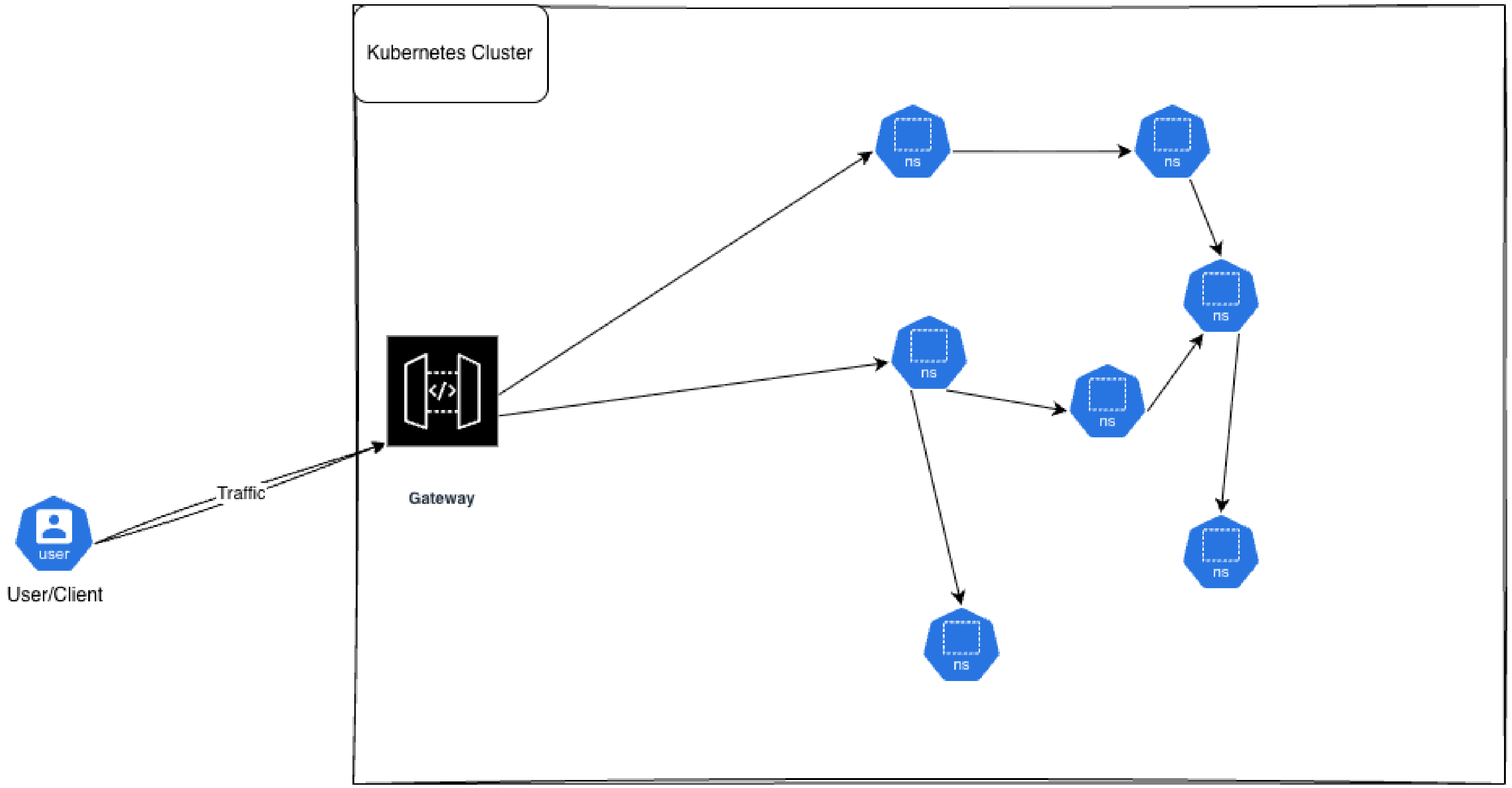

cm
Config Map
secret
Secret
user
User/Client
svc
Service
Deployment
pod


cm
Config Map
secret
Secret
Deployment
pod
Load Balance
pod
Client
Traffic
svc
Service


Namespace
Deployment
cm
Config Map
secret
Secret
pod
Load Balance
svc
Service
pod
Traffic
Traffic
Deployment
cm
Config Map
secret
Secret
pod
Load Balance
svc
Service


Kubernetes Cluster
Namespace
Config Map
cm
Deployment
secret
pod
Secret
Load Balance
svc
Service
Gateway
Traffic
user
User/Client
cm
Config Map
Deployment
secret
pod
Secret
Load Balance
svc
Service


Kubernetes Cluster
Namespace
Deployment
Config Map
Secret
Service
Load Balance
Gateway
Traffic
User/Client
cm
secret
svc
pod
user


Kubernetes Cluster
user
User/Client
Traffic
Gateway
ns
ns
ns
ns
ns
ns
ns
ns

Enter Kubernetes

Kubernetes is a **container orchestration platform** that automates **deployment**, **scaling**, and **management** of containerized applications.

- **Decoupled Teams** → Namespaces & Containers (Isolation).
- **Orchestration** → Controllers (Self-healing state).
- **Standardization** → Declarative YAML (One language for infra).
- **Portability** → Cloud Agnostic (Runs anywhere).

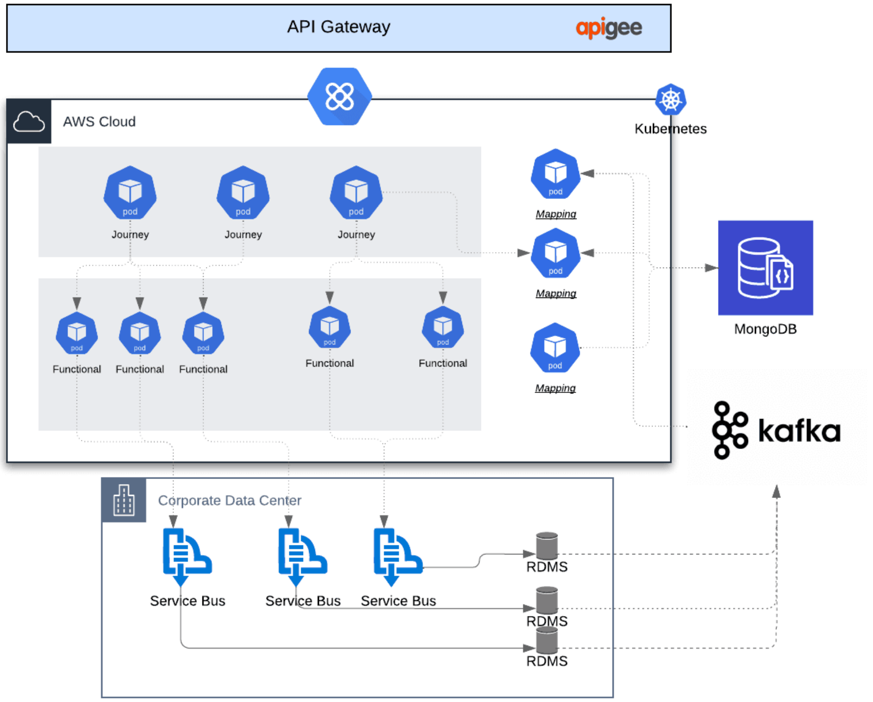
- **Pod**: The smallest unit (One or more containers).
- **Deployment**: Manages the Pods (replicas, updates).
- **Service**: The networking glue (how Pods talk).
- **ConfigMap/Secret**: Configuration management.

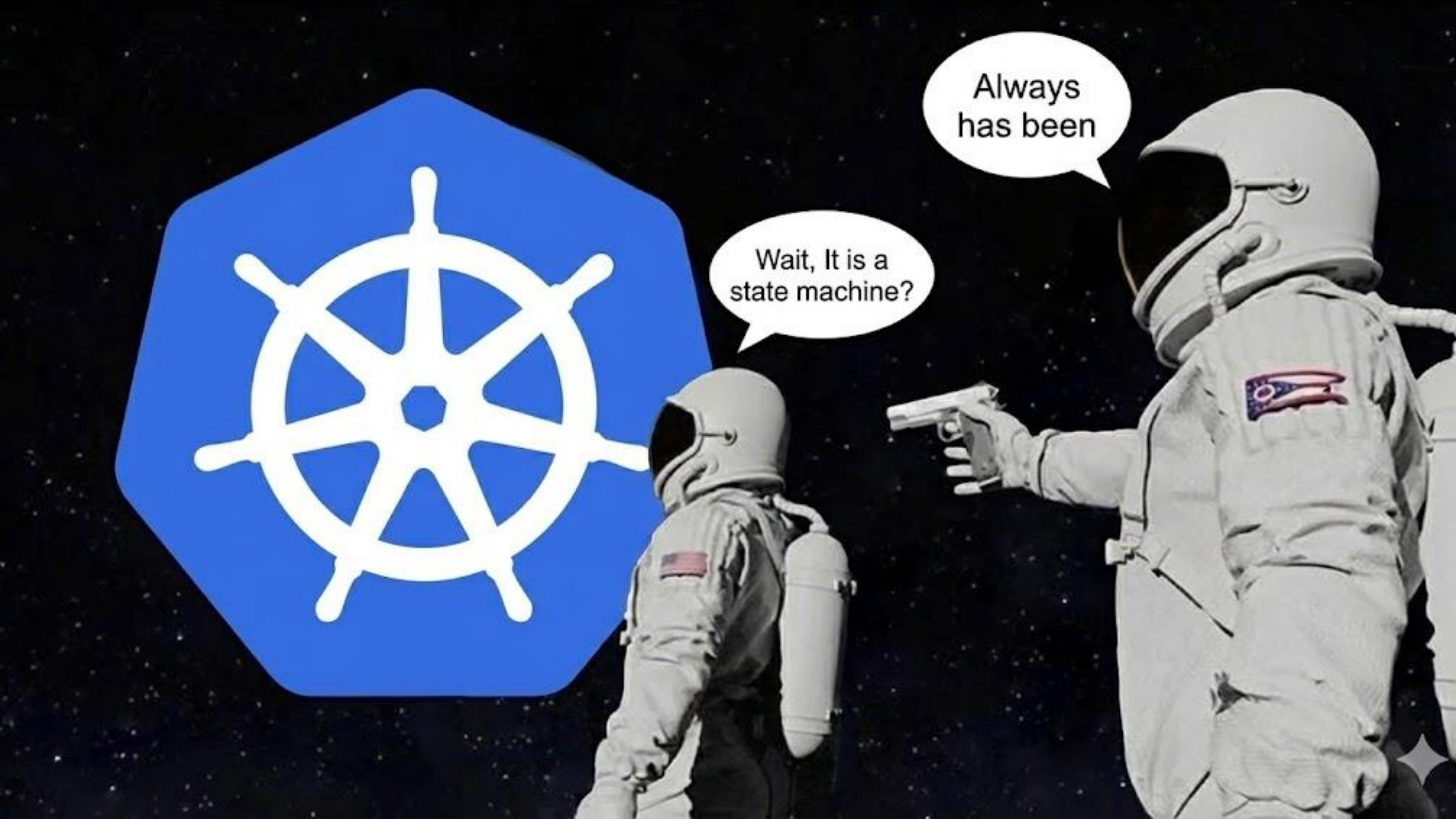
It's so much more than that…

200 Microservices Later – Dxl (Digital Experience Layer)

- The Win:
  - Achieved Standardization & Resilience.
  - Successfully deployed ~200 microservices.

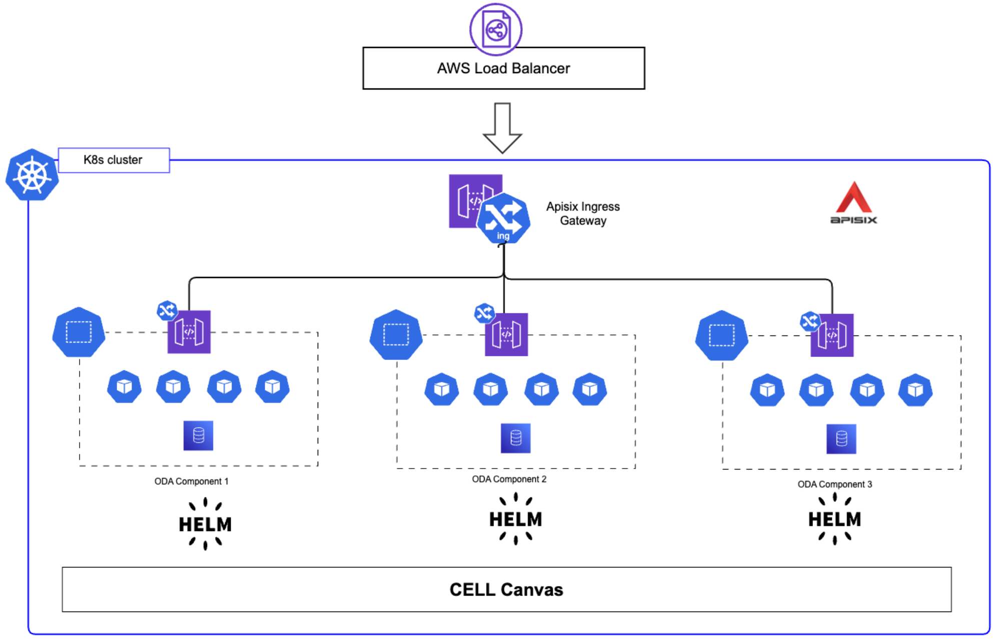
- Lessons Learned:
  - External Dependencies: Routing required manual configuration on external Load Balancers.
  - Operational Bottlenecks: Cross-team communication and tickets slowed us down.




Wait, It is a state machine?
Always has been

Achieving Control

CELL: Componentized Enterprise Logical Layer

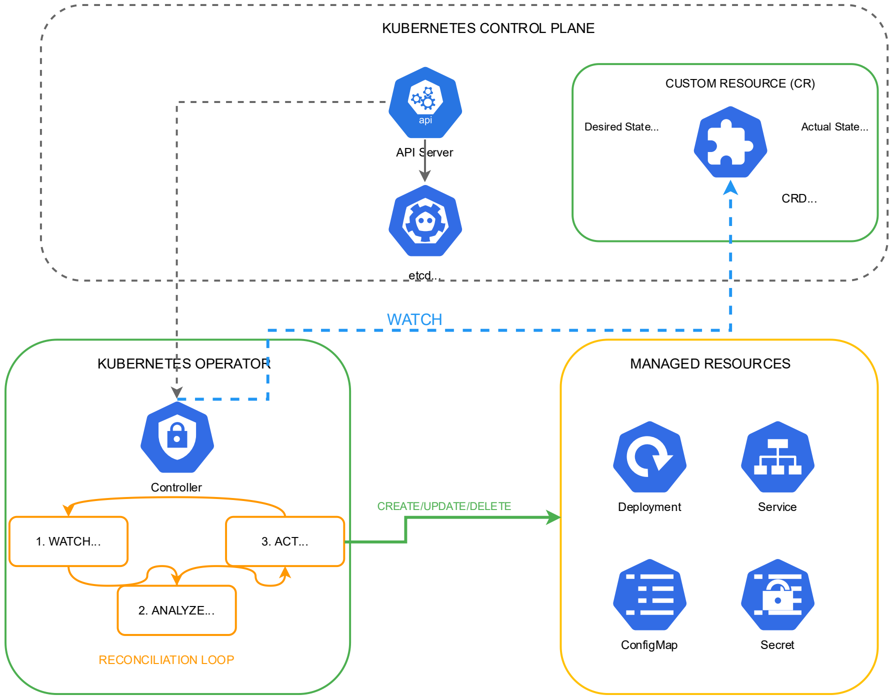


shared libraries

C2 General

But what is a Kubernetes Operator ?

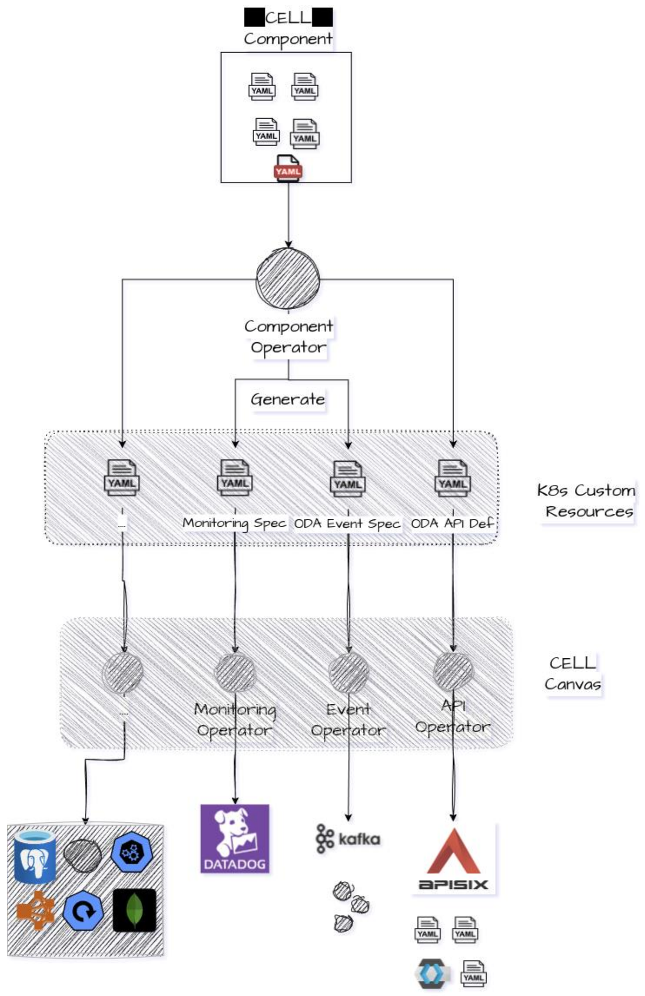
- Kubernetes is a **state** machine
- Everything in Kubernetes is a **Resource**
- Each Resource describes the **desired state**
- We can **extend** Kubernetes API by creating **Custom Resource Definitions** (CRDs)
- Operators **watch** these resources
- When the **actual state** differs from the **desired state**, the Operator **reconciles it**.
- Operators automatically **adjust** the cluster



Live Demo

C.E.L.L Canvas High Level Operation

- **Cell Component (The Source of Truth)**
  - Developers define a single CELL Component (YAML).
  - This abstracts complexity: No need to know Apisix, Kafka, or Datadog specifics.
- **The Reaction**
  - The Component Operator detects the deployment.
  - It automatically generates specialized sub-resources (API Defs, Event Specs, Monitoring Specs).
- **The Execution (Bottom)**
  - Specialized Operators (API, Event, Monitoring) pick up these new resources.
  - They configure the external infrastructure (Apisix, Kafka, Datadog) without human intervention.



C.E.L.L Development Efficiency Result
Examples

	Manual	Automatic C.E.L.L
API Gateway Exposure	15 Days	3 seconds
New Notification Channel	15 Days	3 seconds
New Environment Setup (incl. applications!)	4 months	16mins

Ephemeral Environments!

Also… Reusability!

Lessons learned from production

1. **The "Simpler Solution" Rule**
   - Operators are code, code is a liability. Only build an Operator if you need Active Reconciliation

2. **The "God Operator" Trap**
   - Isolation is key. Run 1 Operator instance per domain. Do not build a monolith that manages Kafka, APIs, and Monitoring all at once.

3. **The State Machine Mindset**
   - Reconciliation > Scripting. Your operator isn't a script that runs once. It's a loop. Ensure your logic is idempotent (safe to run 100 times)

4. **Resilience is Mandatory**
   - Requeue is your friend. If the external API (e.g., Apisix) is down, don't crash. Return Requeue: true and try again in 5 seconds.

5. **The Circular Dependency**
   - Watch your chain. If Operator A waits for Operator B, which waits for Operator A, your cluster will deadlock. Keep dependencies linear.

Choosing your weapon

GoLang

- **The Industry Standard:** Vast community, extensive documentation, first-class citizen in K8s.
- **Performance:** Instant boot times, low memory footprint out-of-the-box.
- **Philosophy:** Toolchain-based. You generate code, then you edit it.
- **Verdict:** Great for pure infrastructure components where every MB counts.

Java

- **The Enterprise Reality:** Leveraging existing team expertise (no learning curve for Go).
- **The Framework Power:** RedHat-backed Java Operator SDK + Quarkus = “Spring Boot for K8s.”
- **The ‘Native’ Trade-off:**
  - **Problem:** JVM memory/startup is heavy for Operators.
  - **Solution:** Native Compilation (GraalVM).
  - **Cost:** Destroys CI/CD speed (slow builds) & makes debugging harder.

Live Demo

Our Bet on Tomorrow

AI Agents

Declaratively specify required MCP servers in component manifest

Operators automatically deploy and wire up the needed MCP infrastructure

"Deploy an AI agent, get its entire context infrastructure - automatically»

Everything as Code, Everything Automated

Policy-as-code across all enterprise processes

Zero-touch infrastructure provisioning

Composable at Every Layer

Cluster-as-a-Service through meta-operators

Component marketplaces within the enterprise

Operators that deploy operators or whole clusters

Resources

- https://sdk.operatorframework.io
- https://javaoperatorsdk.io
- https://javaoperatorsdk.io
- https://github.com/Nikontem/website-operator-go
- https://github.com/Nikontem/website-operator-java

- https://www.linkedin.com/in/nikos-ntemkas-34093684/
- https://www.linkedin.com/in/george-binas-1b51a1105/

Questions?

Thank you

Let’s connect

Discover opportunities at Vodafone Tech Hub Thessaloniki

Together we can