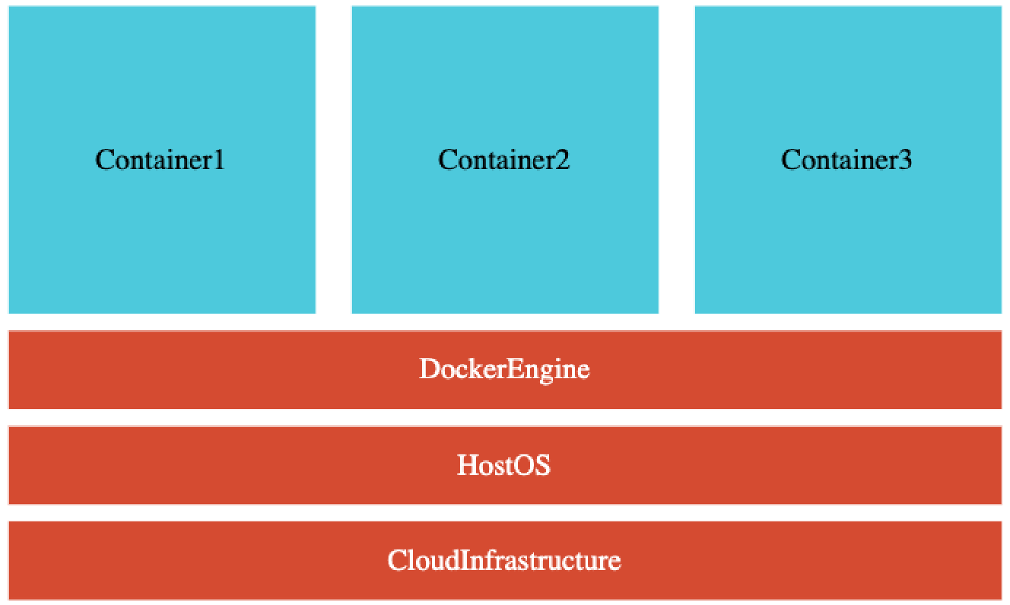
Microservices

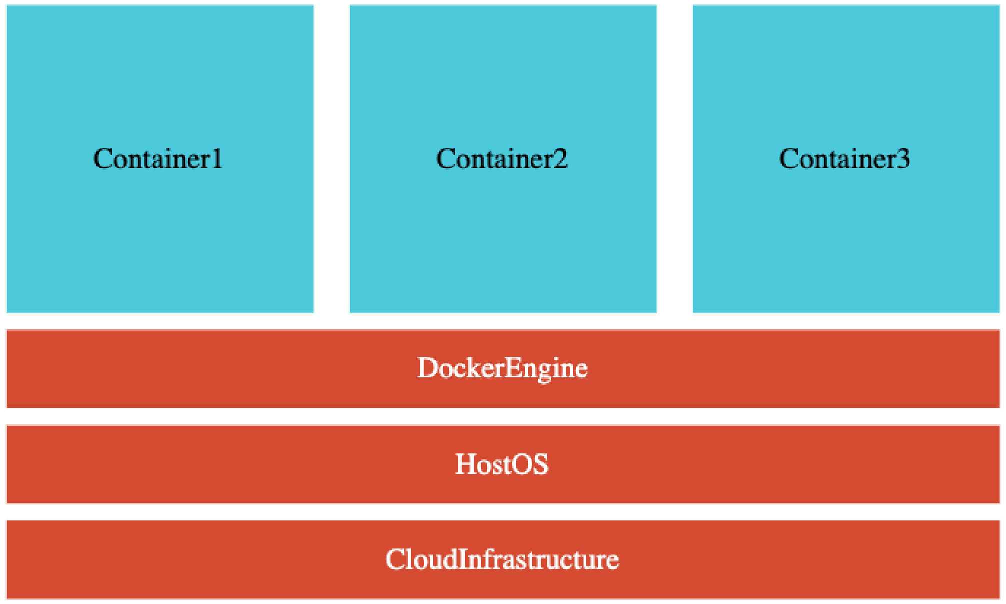
Microservices - V2

- **V2** - Latest Releases of
  - Spring Boot
  - Spring Cloud
  - Docker and
  - Kubernetes
  - **Skip to Next Section :)**
- **V1** - Old Versions
  - Spring Boot v2.3 and LOWER
  - **Continue on to next lecture :(**



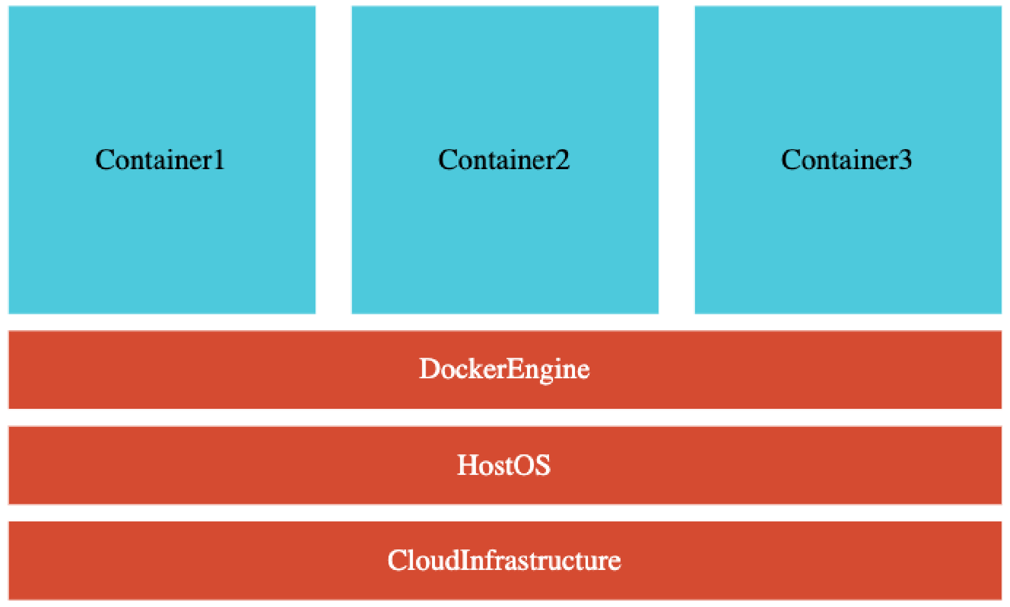
Microservices - V2

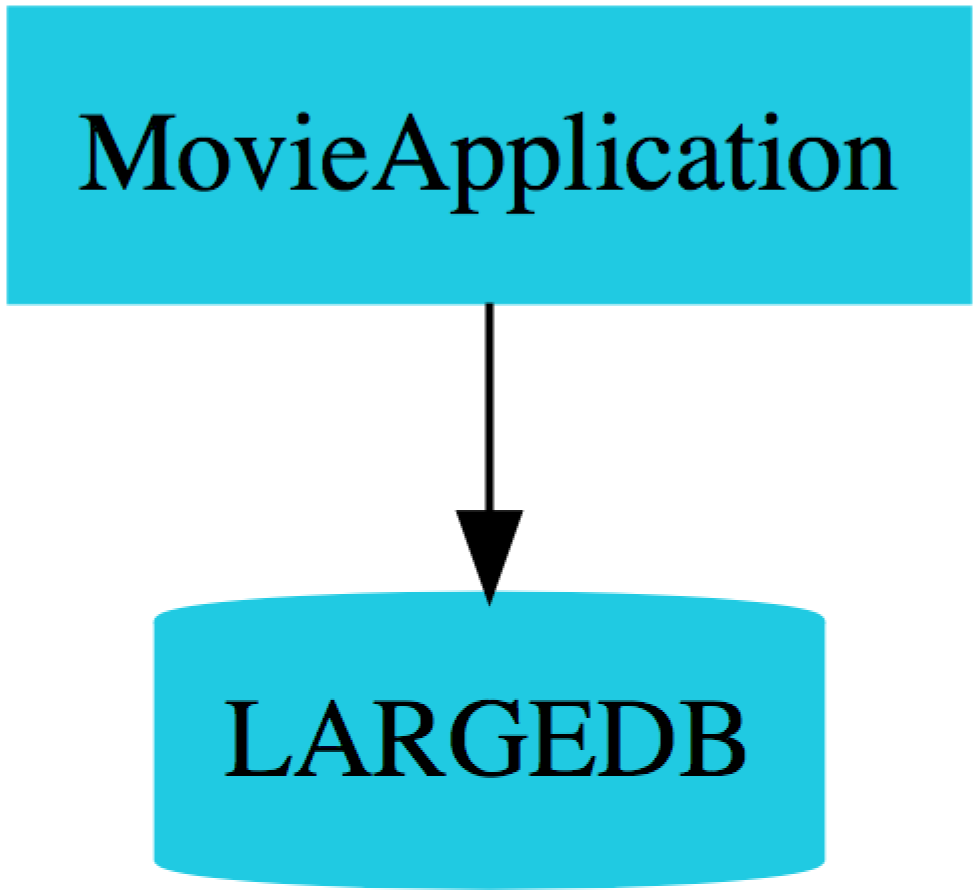
- You have **skipped V1**
  - Go to next lecture!
- You have **completed V1**
  - Option 1: **Start from Zero Again:**
    - Go to the next lecture!
  - Option 2: **Get a Quick Start:**
    - Jump to **"Step 21 - QuickStart by Importing Microservices"**
      - Same microservices as V1: **Currency Exchange** and **Currency Conversion**
      - Very little changes in **Eureka Naming Server**
      - **Step 21** helps you set these up and get started quickly!



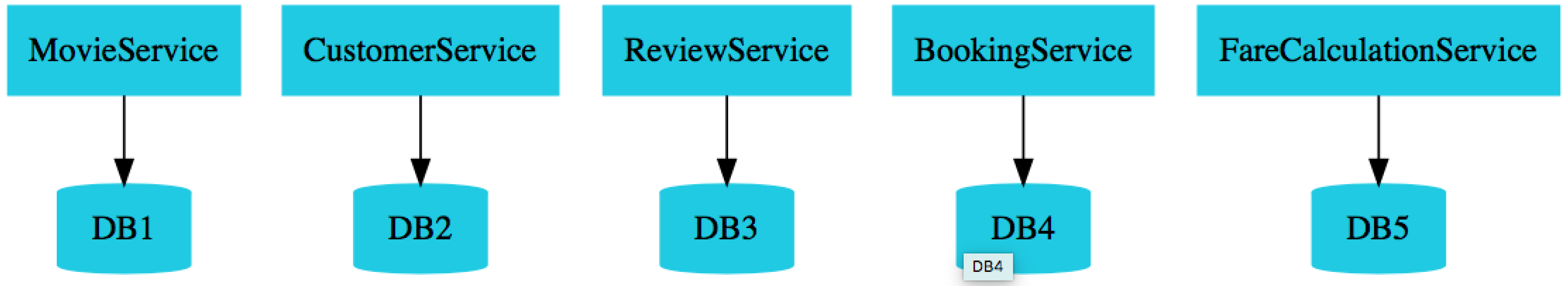
Microservices - V2 - What's New

- Microservices Evolve Quickly
- **Important Updates:**
  - Latest Versions of Spring Boot & Spring Cloud
    - **Spring Cloud LoadBalancer** instead of Ribbon
    - **Spring Cloud Gateway** instead of Zuul
    - **Resilience4j** instead of Hystrix
  - **Docker**: Containerize Microservices
    - Run microservices using Docker and Docker Compose
  - **Kubernetes:** Orchestrate all your Microservices with Kubernetes



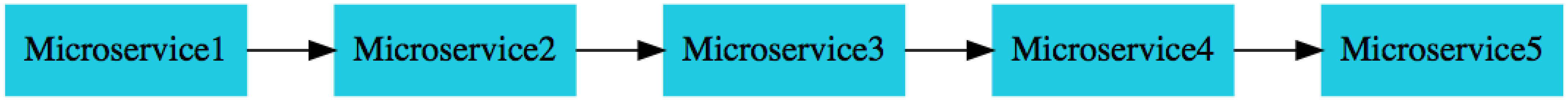


Monolith



Microservices

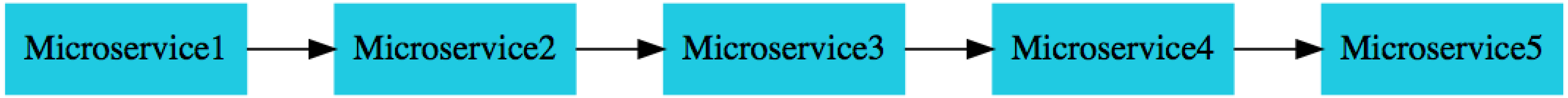
What is a Microservice?



Small autonomous services that work together

Sam Newman

What is a Microservice?

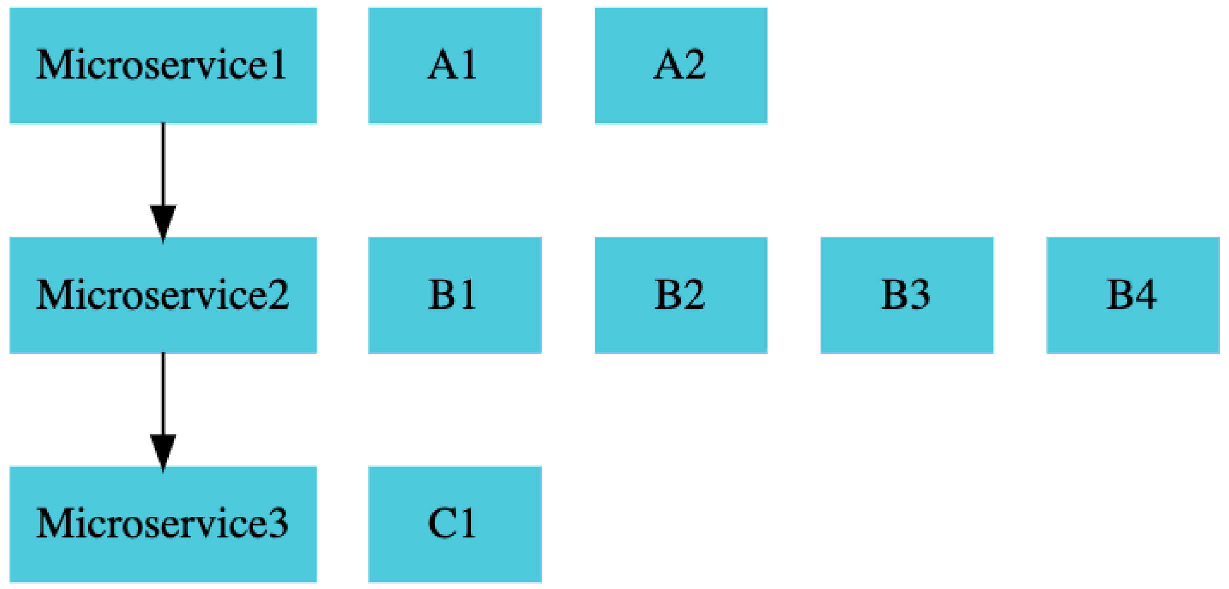


Approach to developing a application as a suite of small services, each running in its own process and communicating with lightweight mechanisms often an HTTP resource API.
These services are built around business capabilities and independently deployable by fully automated deployment machinery.
There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

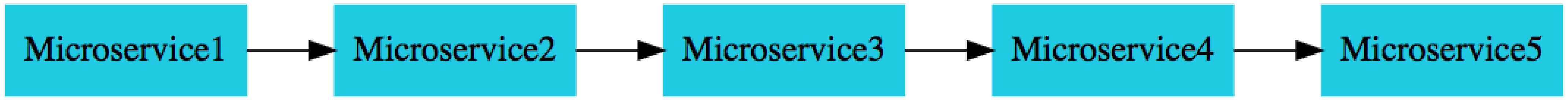
James Lewis and Martin Fowler

Microservices for me

- REST
- Small Well Chosen Deployable Units
- Cloud Enabled

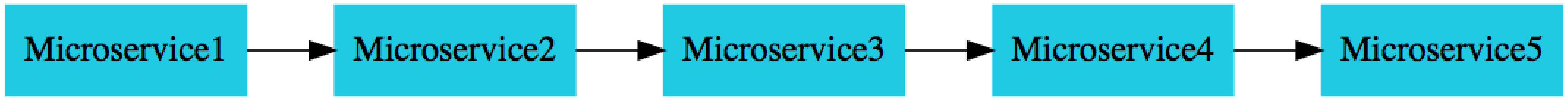


Microservices - Challenges



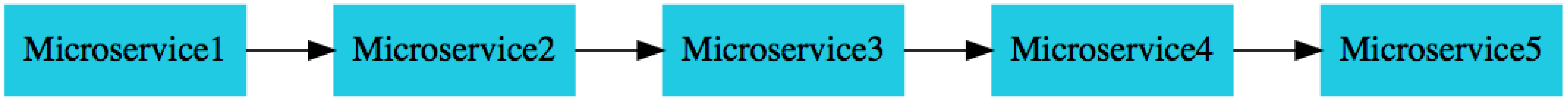
- Bounded Context
- Configuration Management
- Dynamic Scale Up and Scale Down
- Visibility
- Pack of Cards
- Zero Downtime Deployments

Microservice - Solutions



- **Spring Cloud** Umbrella Projects
  - Centralized Configuration Management (Spring Cloud Config Server)
  - Location Transparency - Naming Server (Eureka)
  - Load Distribution (Ribbon, Spring Cloud Load Balancer)
  - Visibility and Monitoring (Zipkin)
  - API Gateway (Zuul, Spring Cloud Gateway)
  - Fault Tolerance (Hystrix, Resilience4j)
- **Docker**: Language Neutral, Cloud Neutral deployable units
- **Kubernetes**: Orchestrate Thousands of Microservices

Microservices - 3 Key Advantages



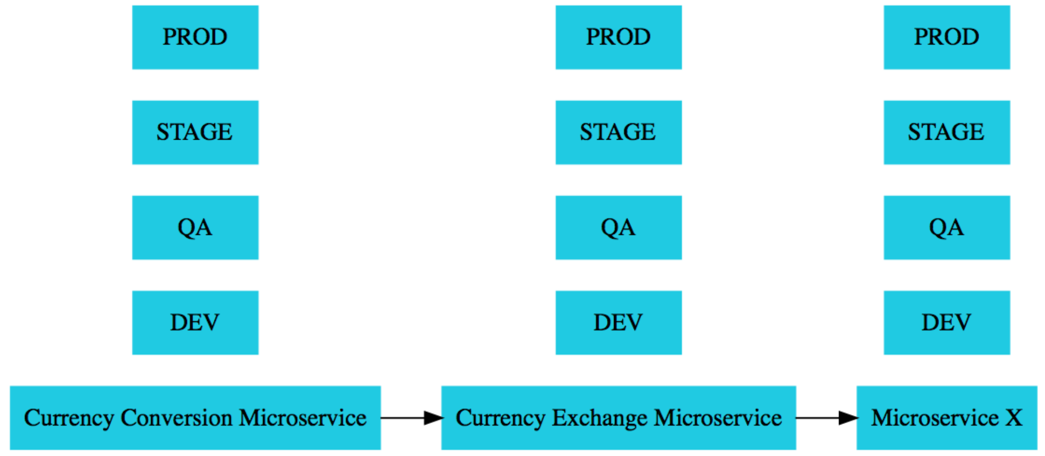
- New Technology & Process Adoption
- Dynamic Scaling
- Faster Release Cycles

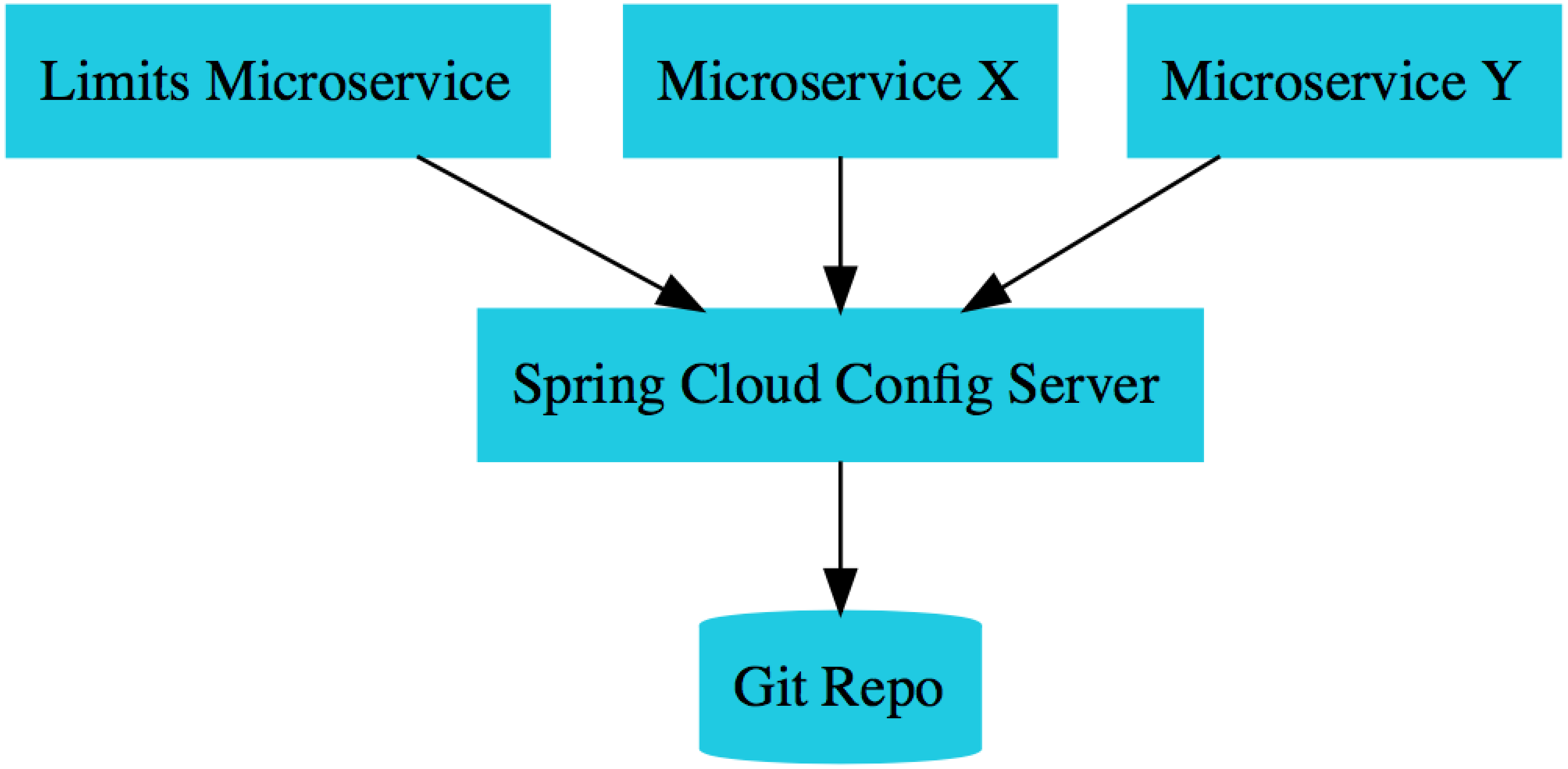
Ports Standardization

Application	Port
Limits Microservice	8080, 8081, ...
Spring Cloud Config Server	8888
Currency Exchange Microservice	8000, 8001, 8002, ..
Currency Conversion Microservice	8100, 8101, 8102, ...
Netflix Eureka Naming Server	8761
API Gateway	8765
Zipkin Distributed Tracing Server	9411

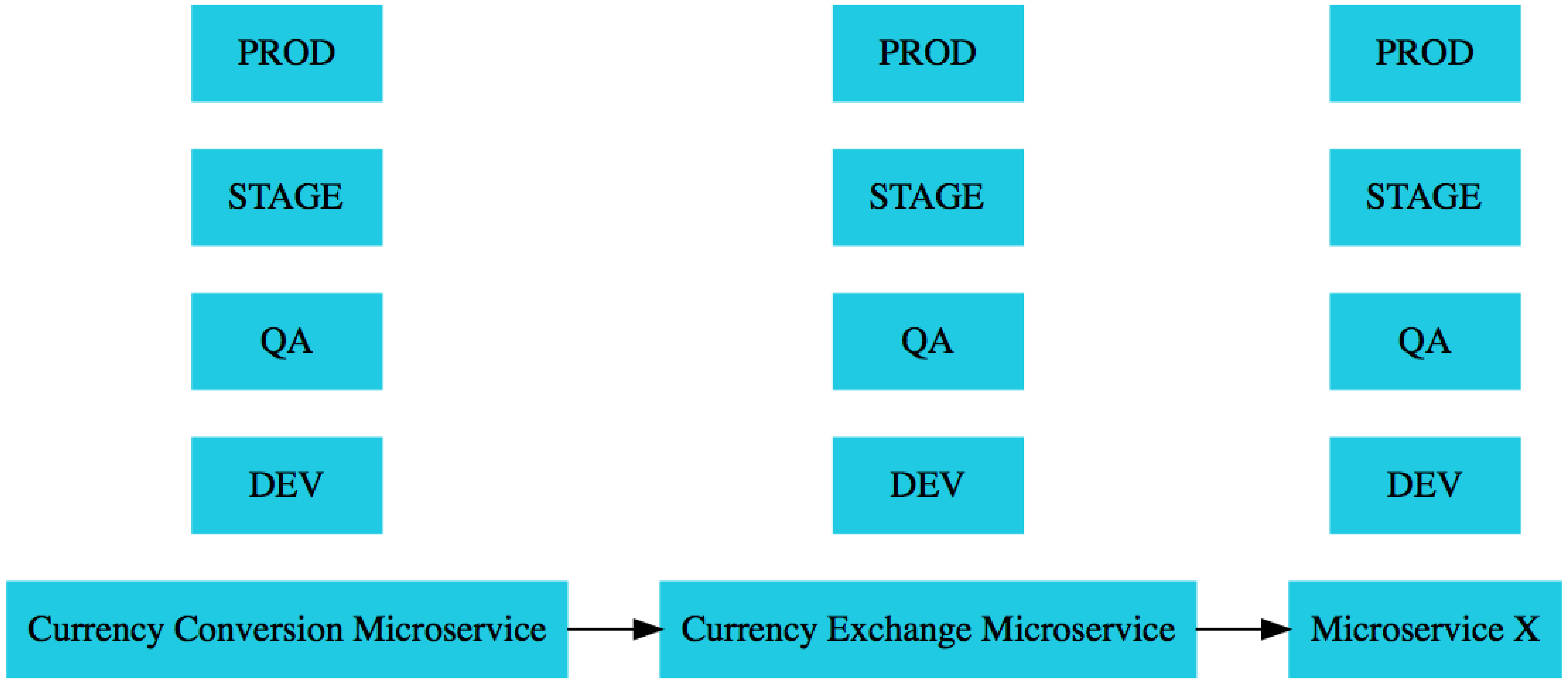
Need for Centralized Configuration

- Lot of configuration:
  - External Services
  - Database
  - Queue
  - Typical Application Configuration
- Configuration variations:
  - 1000s of Microservices
  - Multiple Environments
  - Multiple instances in each Environment
- How do you manage all this configuration?

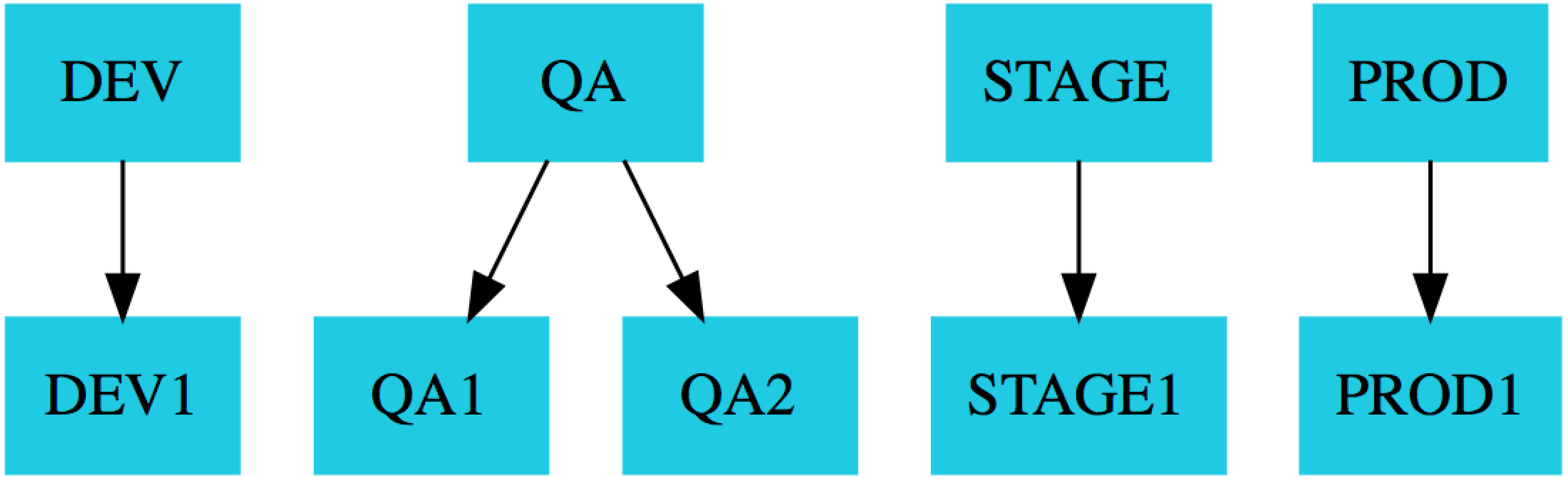

PROD
STAGE
QA
DEV
Currency Conversion Microservice
PROD
STAGE
QA
DEV
Currency Exchange Microservice
PROD
STAGE
QA
DEV
Microservice X



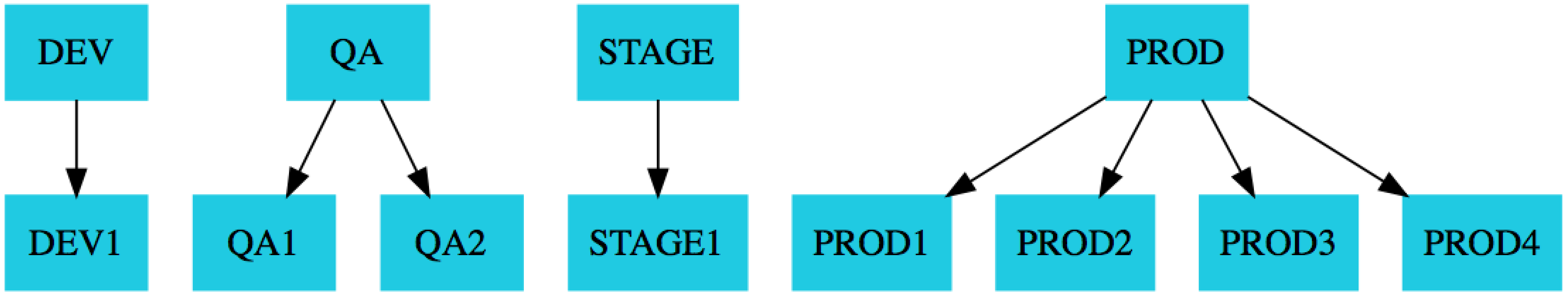
Config Server


PROD
STAGE
QA
DEV
Currency Conversion Microservice
PROD
STAGE
QA
DEV
Currency Exchange Microservice
PROD
STAGE
QA
DEV
Microservice X

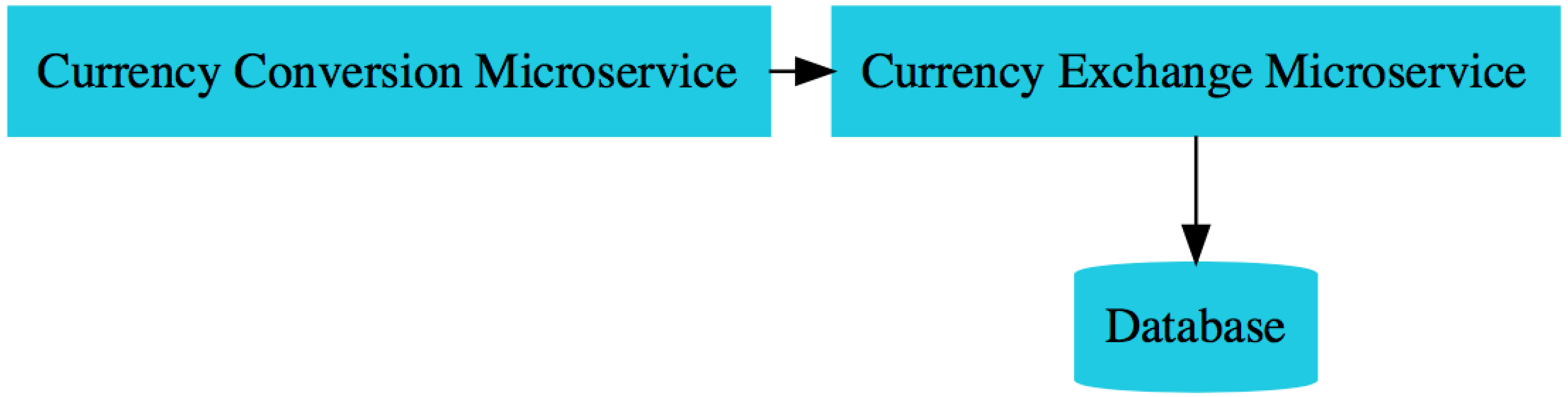
Environments



Environments



Environments


Currency Conversion Microservice
Currency Exchange Microservice
Database

Microservices Overview

Currency Exchange Microservice

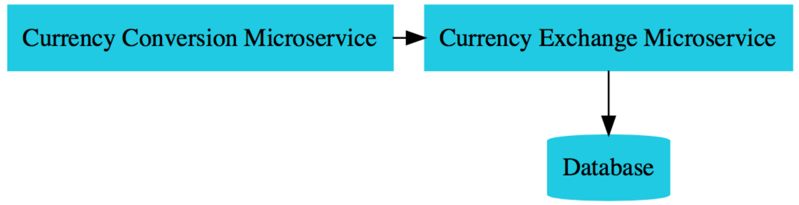
What is the exchange rate of one currency in another?

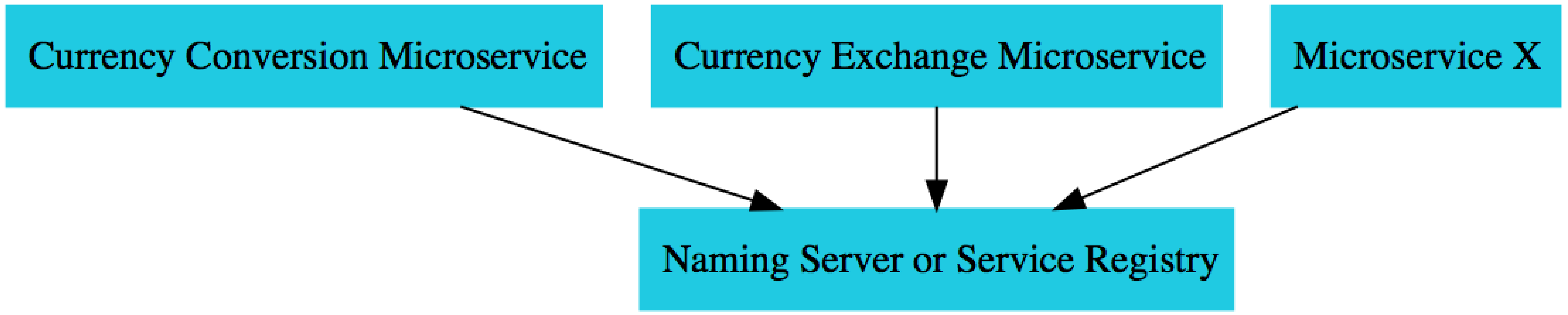
```
http://localhost:8000/currency-exchange/from/USD/to/INR

{
"id":10001,
"from":"USD",
"to":"INR",
"conversionMultiple":65.00,
"environment":"8000 instance-id"
}
```

Currency Conversion Microservice

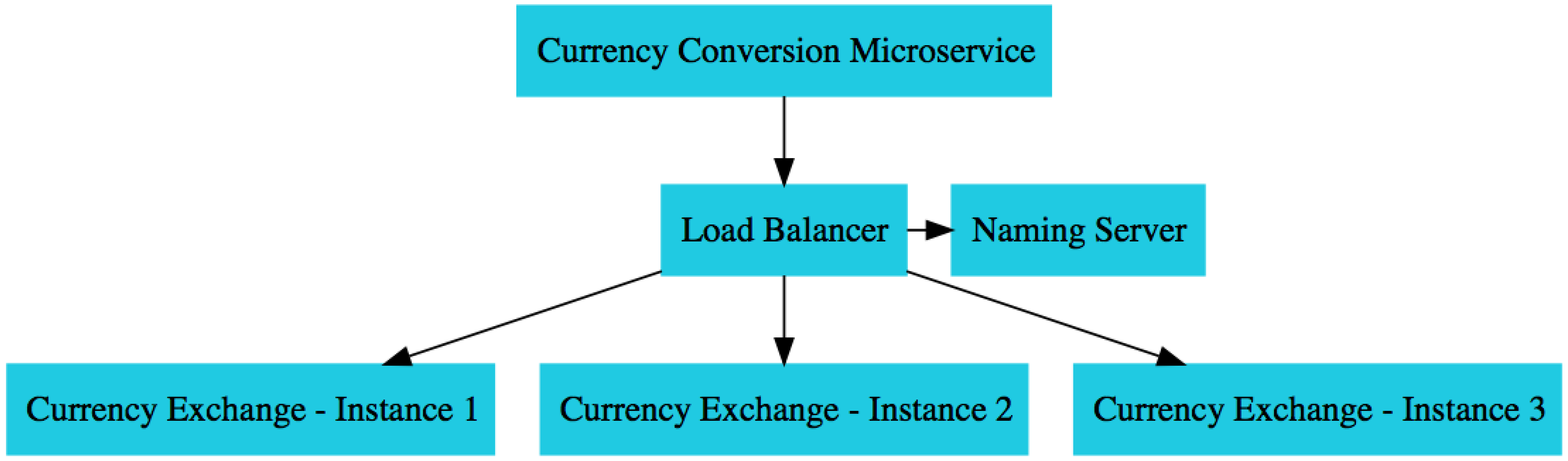
Convert 10 USD into INR

```
http://localhost:8100/currency-conversion/from/USD/to/INR/quantity/10

{
  "id": 10001,
  "from": "USD",
  "to": "INR",
  "conversionMultiple": 65.00,
  "quantity": 10,
  "totalCalculatedAmount": 650.00,
  "environment": "8000 instance-id"
}
```


Currency Conversion Microservice
Currency Exchange Microservice
Database


Currency Conversion Microservice
Currency Exchange Microservice
Microservice X
Naming Server or Service Registry

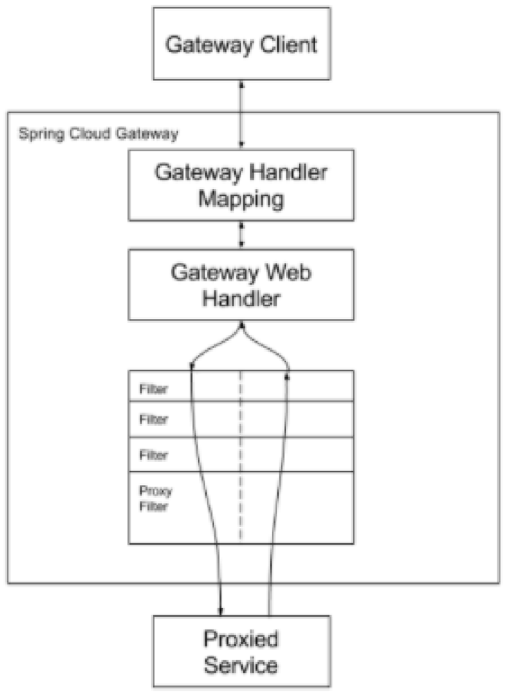
Naming Server


Currency Conversion Microservice
Load Balancer
Naming Server
Currency Exchange - Instance 1
Currency Exchange - Instance 2
Currency Exchange - Instance 3

Load Balancing

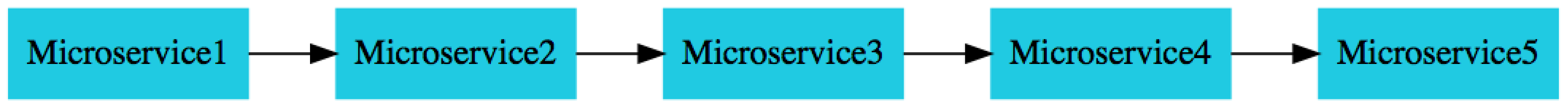
Spring Cloud Gateway

- Simple, yet effective way to route to APIs
- Provide cross cutting concerns:
  - Security
  - Monitoring/metrics
- Built on top of Spring WebFlux (Reactive Approach)
- Features:
  - Match routes on any request attribute
  - Define Predicates and Filters
  - Integrates with Spring Cloud Discovery Client (Load Balancing)
  - Path Rewriting



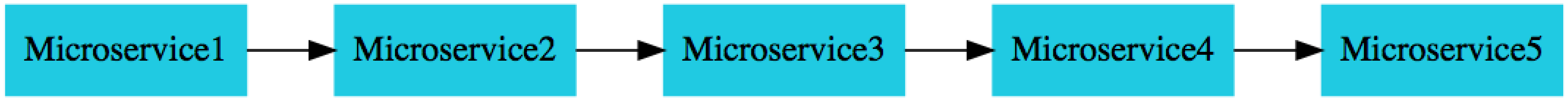
From *https://docs.spring.io*

Circuit Breaker

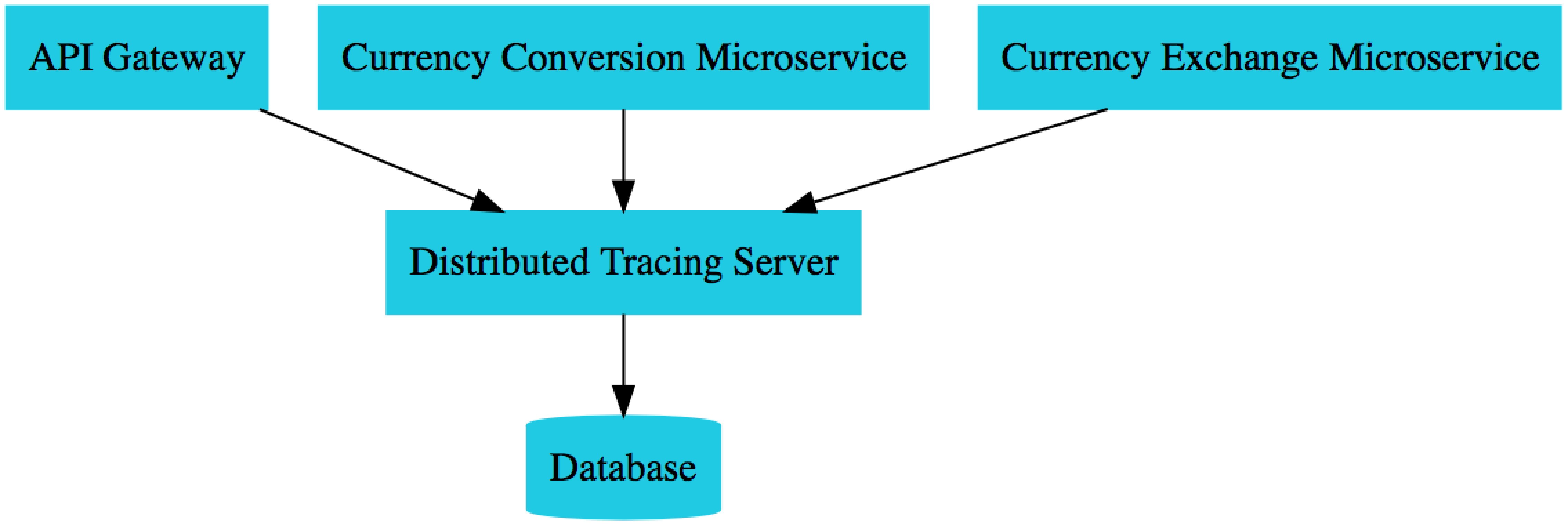


- What if one of the services is down or is slow?
  - Impacts entire chain!
- Questions:
  - Can we return a fallback response if a service is down?
  - Can we implement a Circuit Breaker pattern to reduce load?
  - Can we retry requests in case of temporary failures?
  - Can we implement rate limiting?
- Solution: Circuit Breaker Framework - Resilience4j

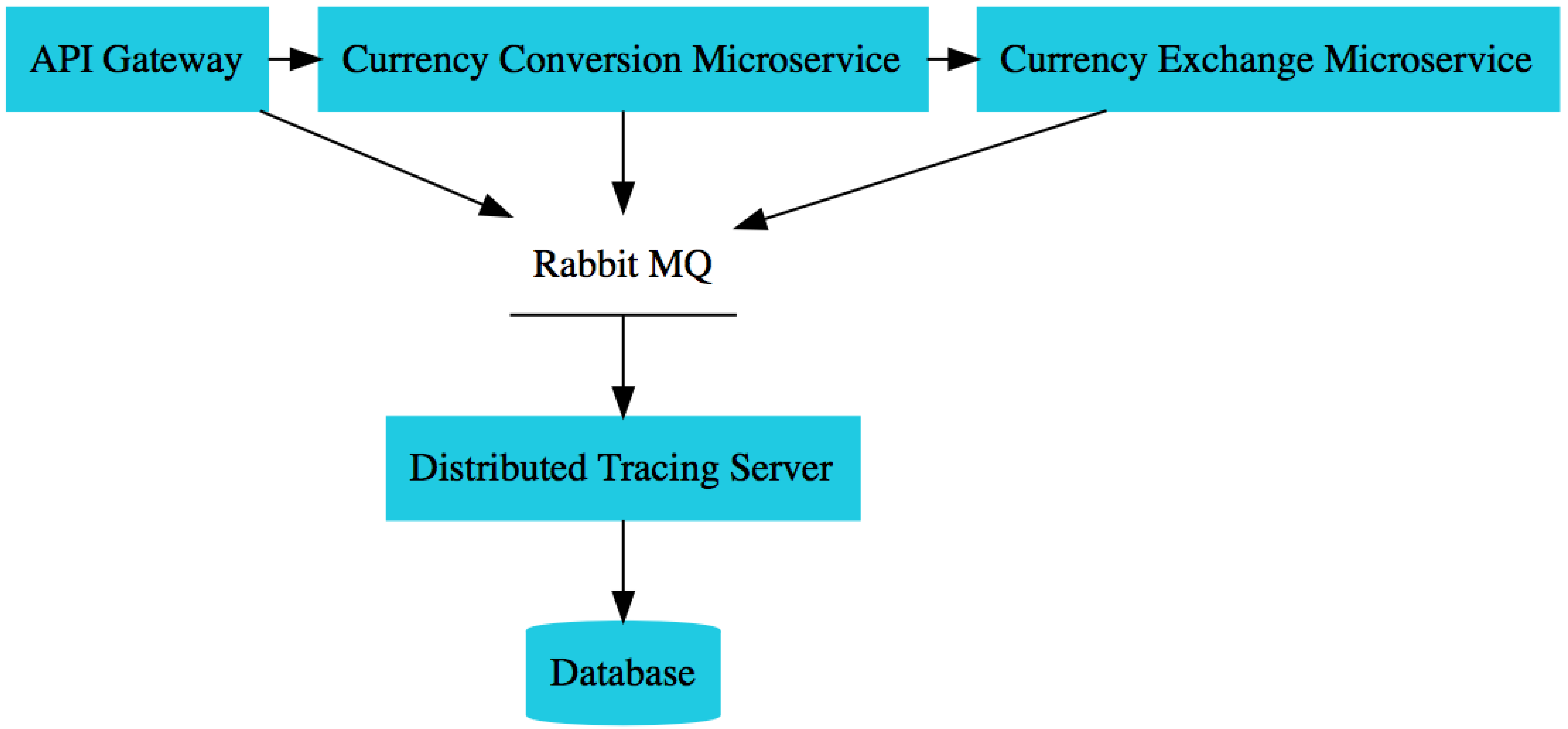
Distributed Tracing



- Complex call chain
- How do you debug problems?
- How do you trace requests across microservices?
- Enter Distributed Tracing

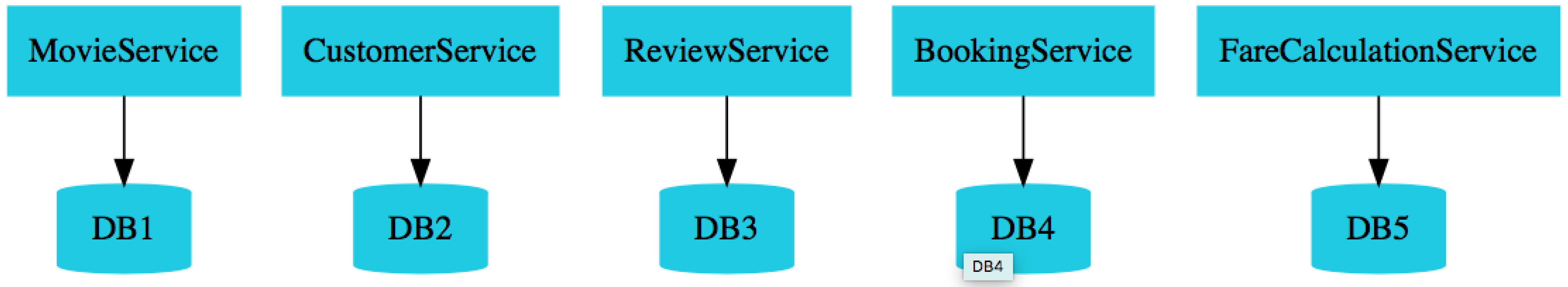

API Gateway
Currency Conversion Microservice
Currency Exchange Microservice
Distributed Tracing Server
Database

Distributed Tracing


API Gateway
Currency Conversion Microservice
Currency Exchange Microservice
Rabbit MQ
Distributed Tracing Server
Database

Distributed Tracing - Asynchronous

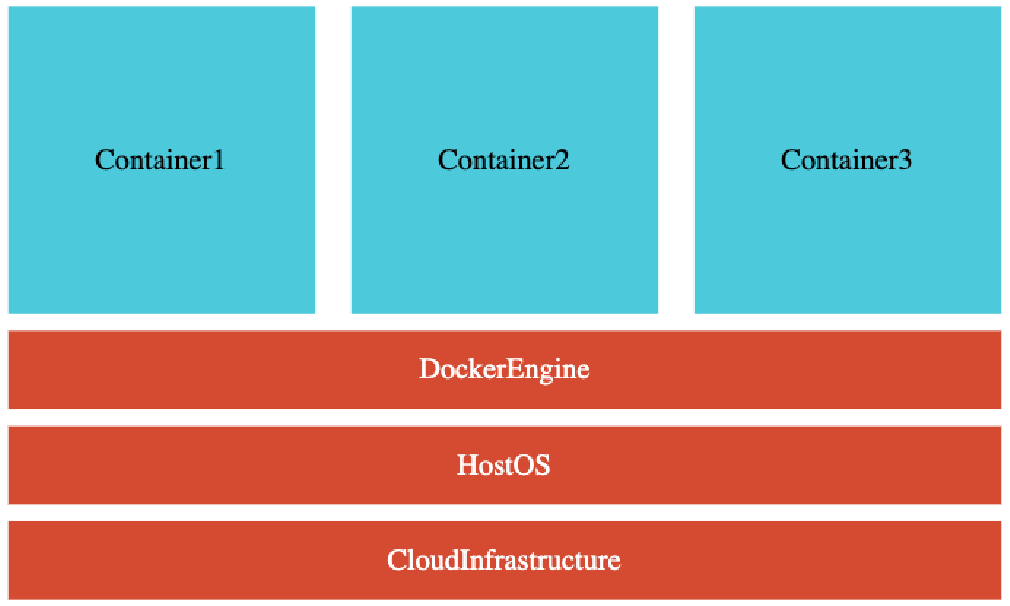
Microservices

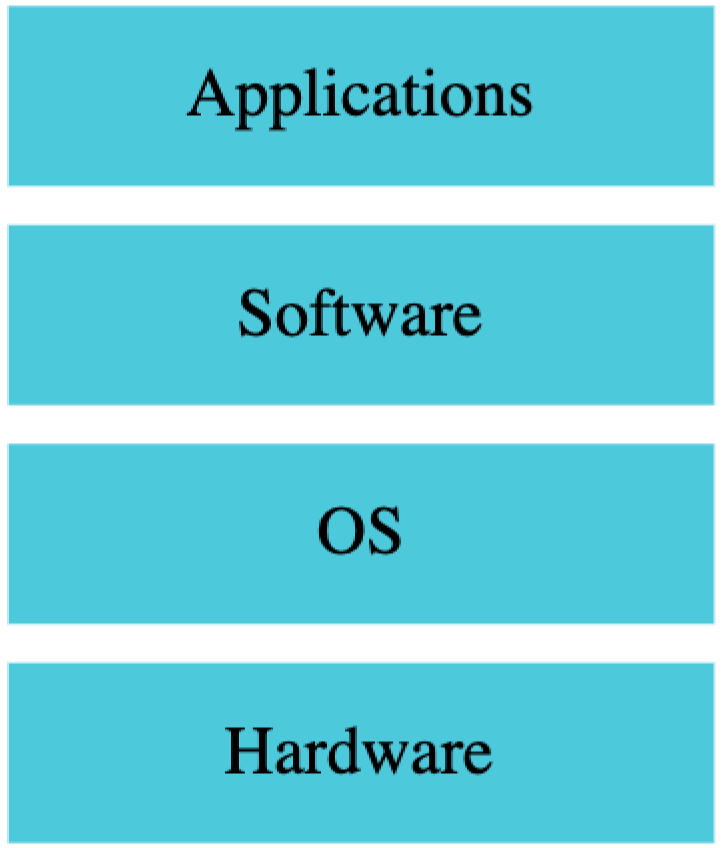


- Enterprises are heading towards microservices architectures
  - Build small focused microservices
  - **Flexibility to innovate** and build applications in different programming languages (Go, Java, Python, JavaScript, etc)
  - BUT **deployments become complex**!
  - How can we have **one way of deploying** Go, Java, Python or JavaScript .. microservices?
    - Enter **containers**!

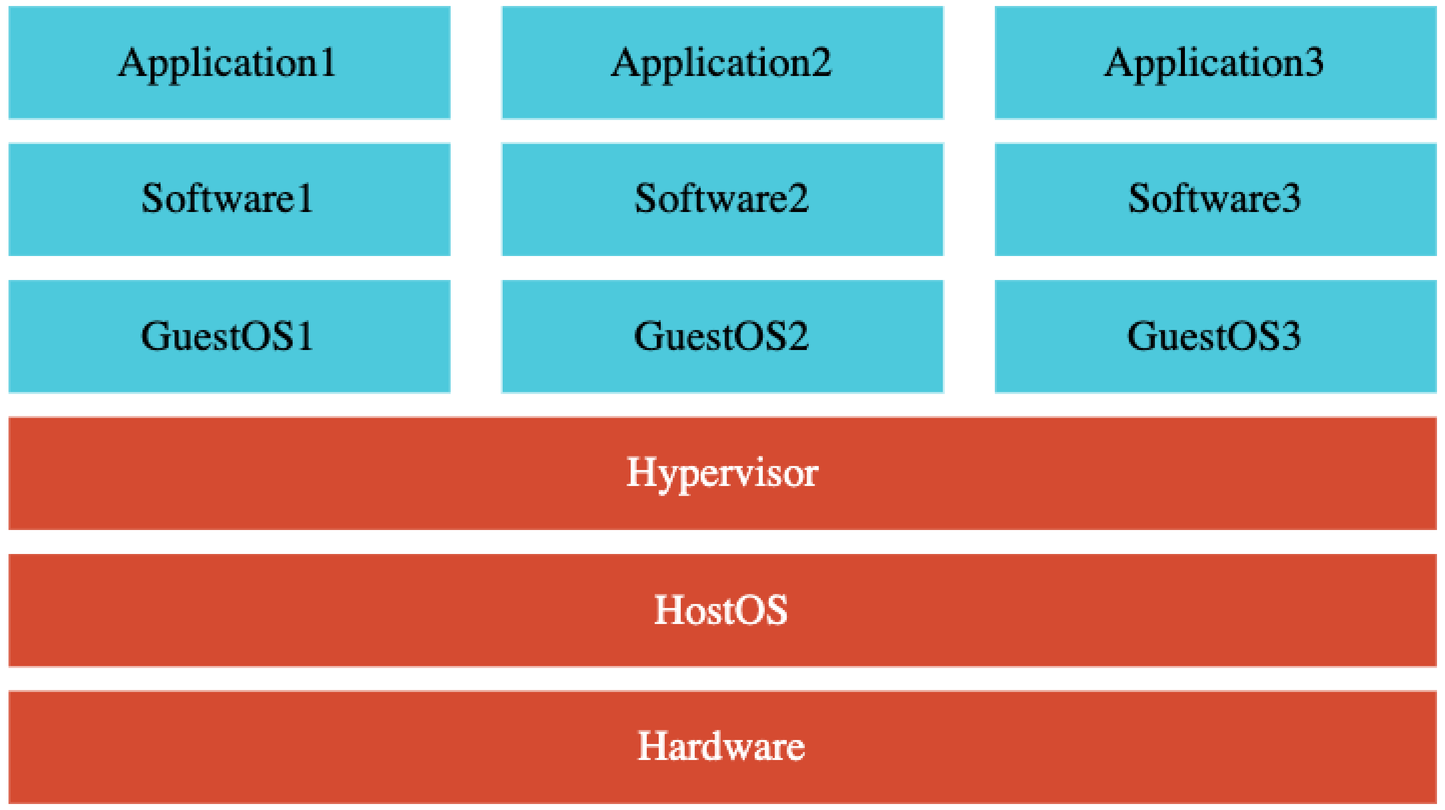
Docker

- Create **Docker images** for each microservice
- Docker image **contains everything a microservice needs** to run:
  - Application Runtime (JDK or Python or NodeJS)
  - Application code
  - Dependencies
- You can run these docker containers **the same way** on any infrastructure
  - Your local machine
  - Corporate data center
  - Cloud

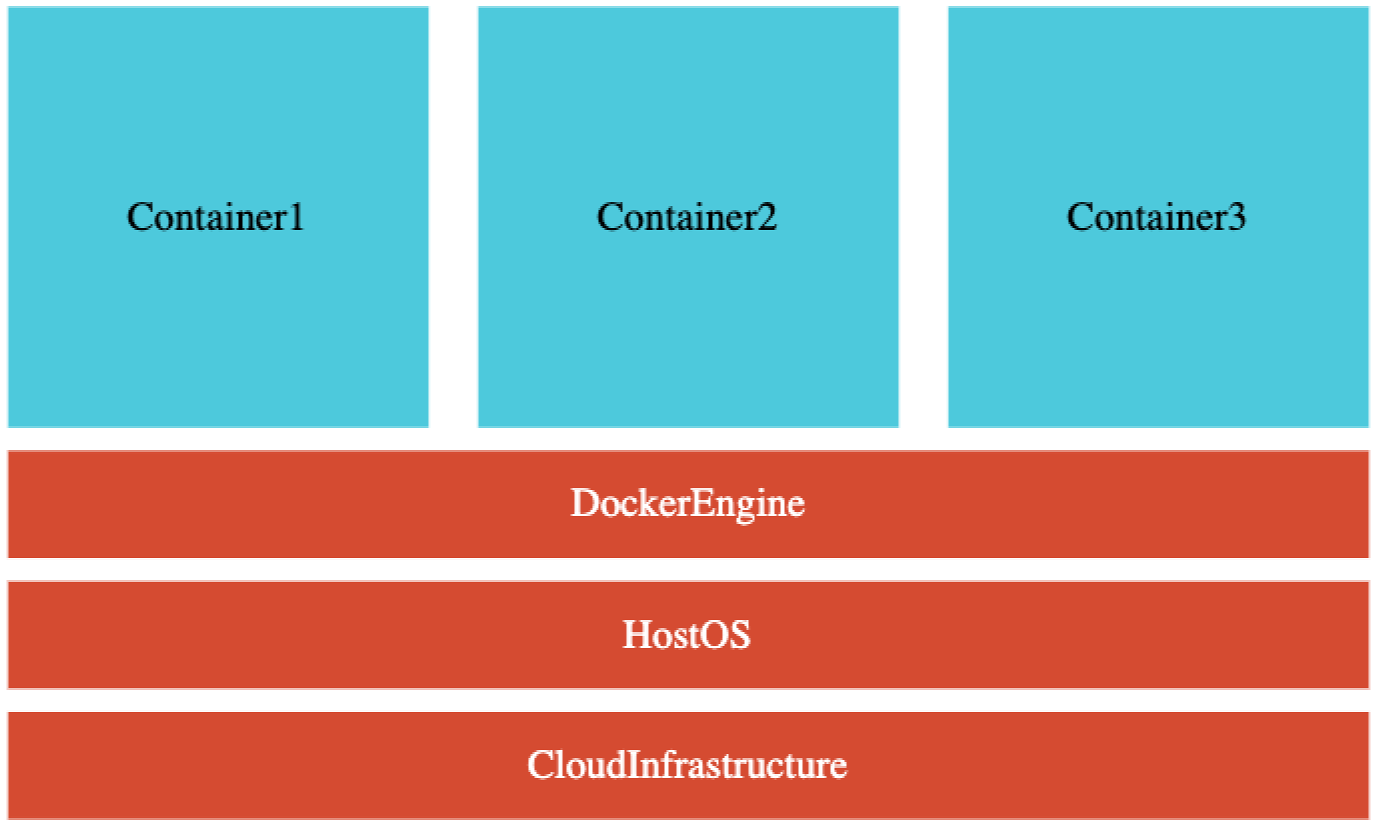




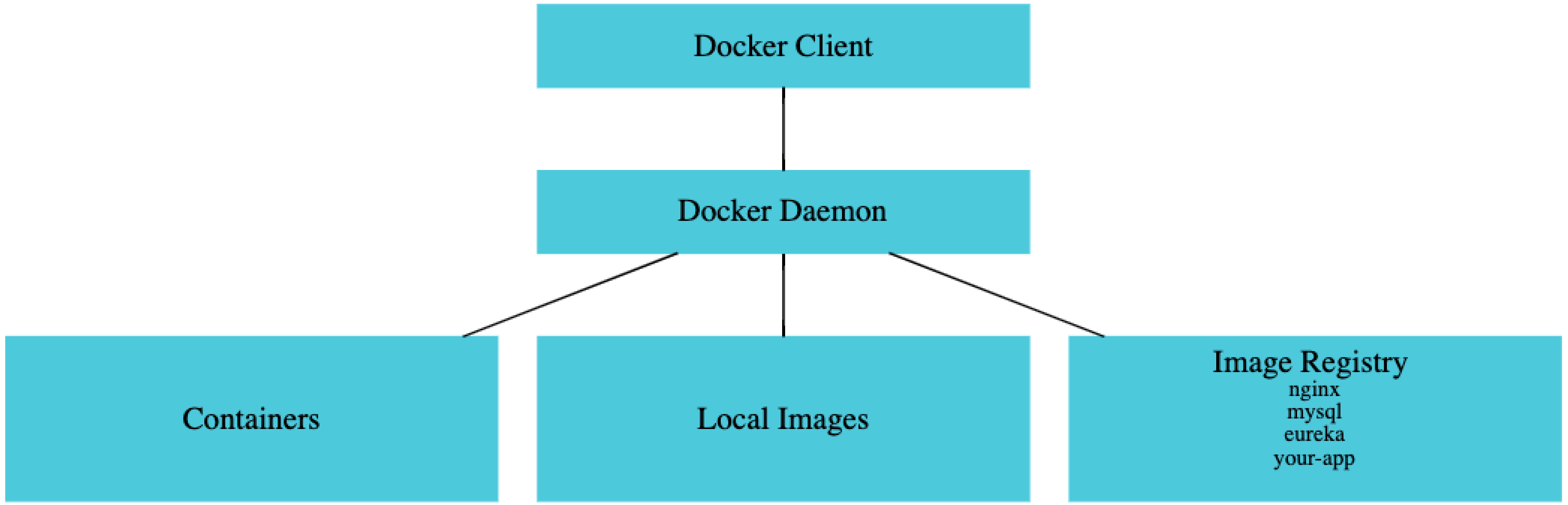
Traditional Deployment



Deployments using Virtual Machines



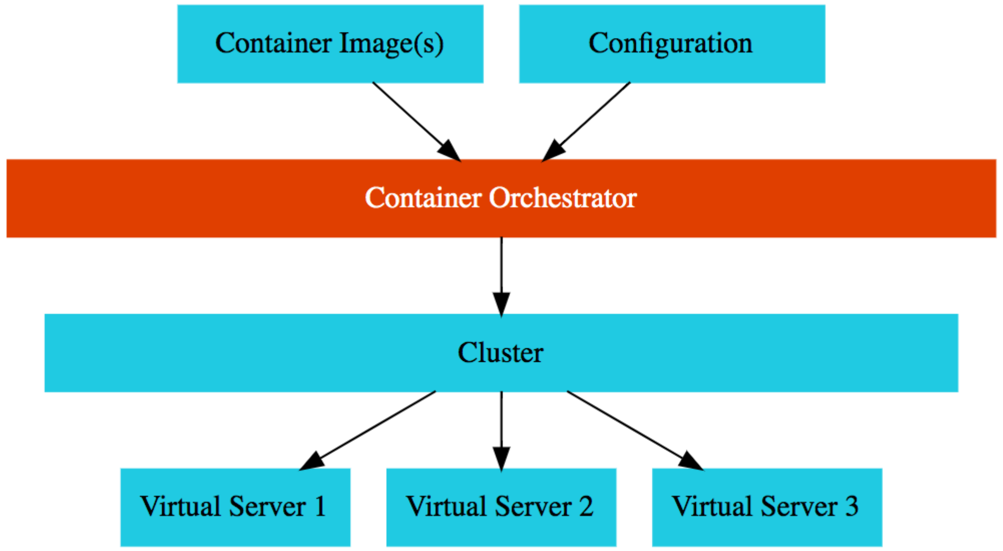
Deployments using Docker



Docker Architecture

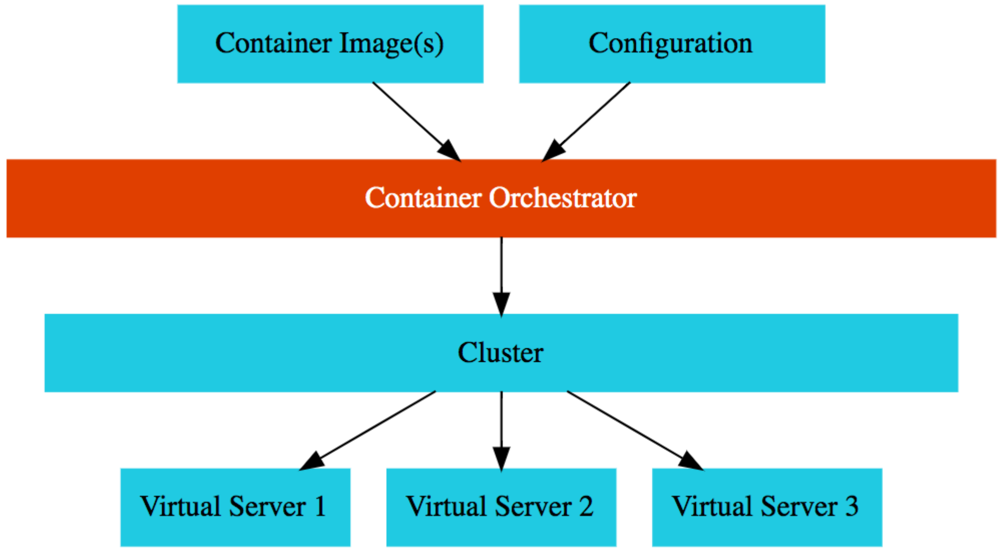
Container Orchestration

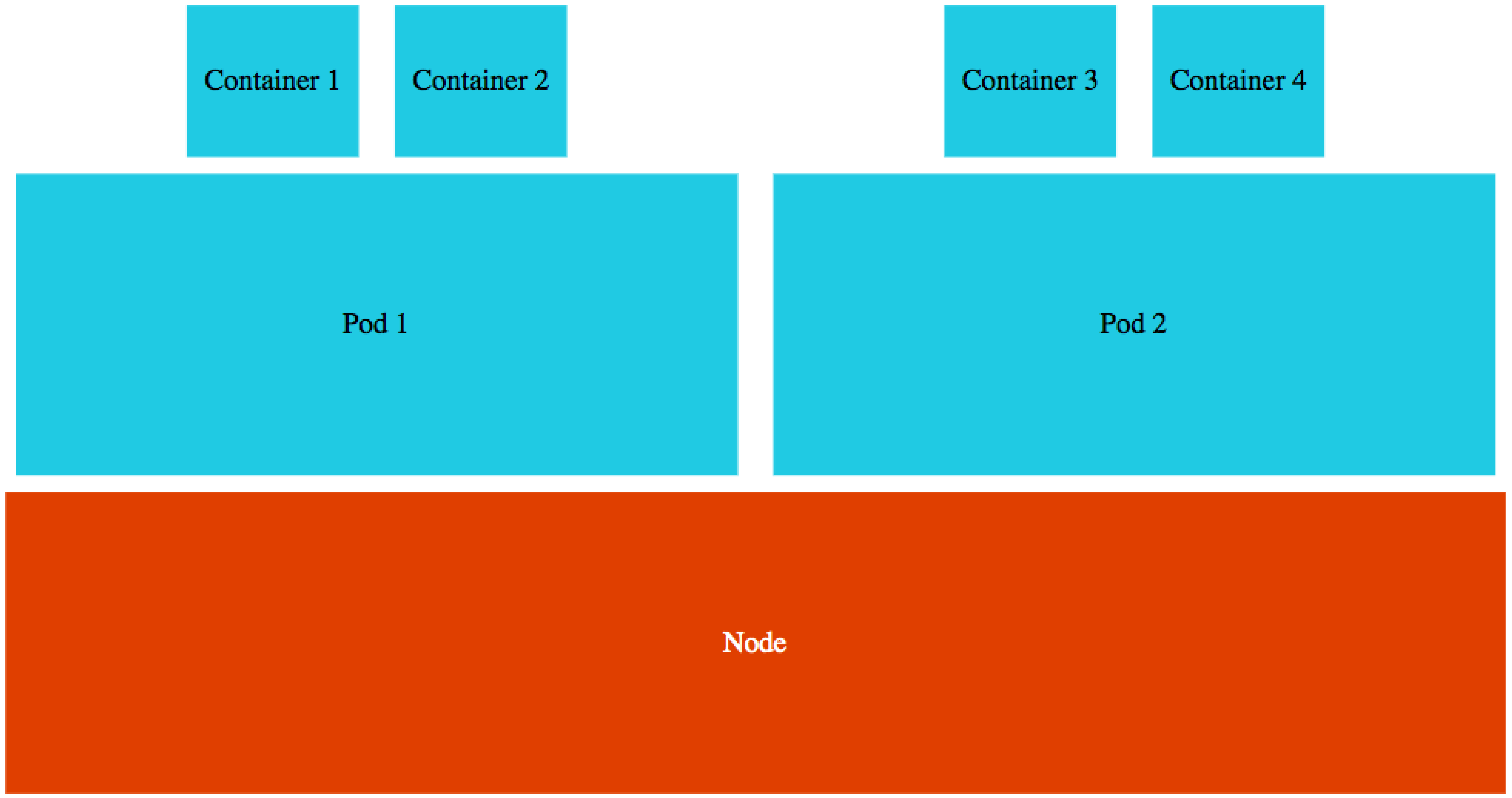
- **Requirement** : I want 10 instances of Microservice A container, 15 instances of Microservice B container and ....
- Typical Features:
  - **Auto Scaling** - Scale containers based on demand
  - **Service Discovery** - Help microservices find one another
  - **Load Balancer** - Distribute load among multiple instances of a microservice
  - **Self Healing** - Do health checks and replace failing instances
  - **Zero Downtime Deployments** - Release new versions without downtime



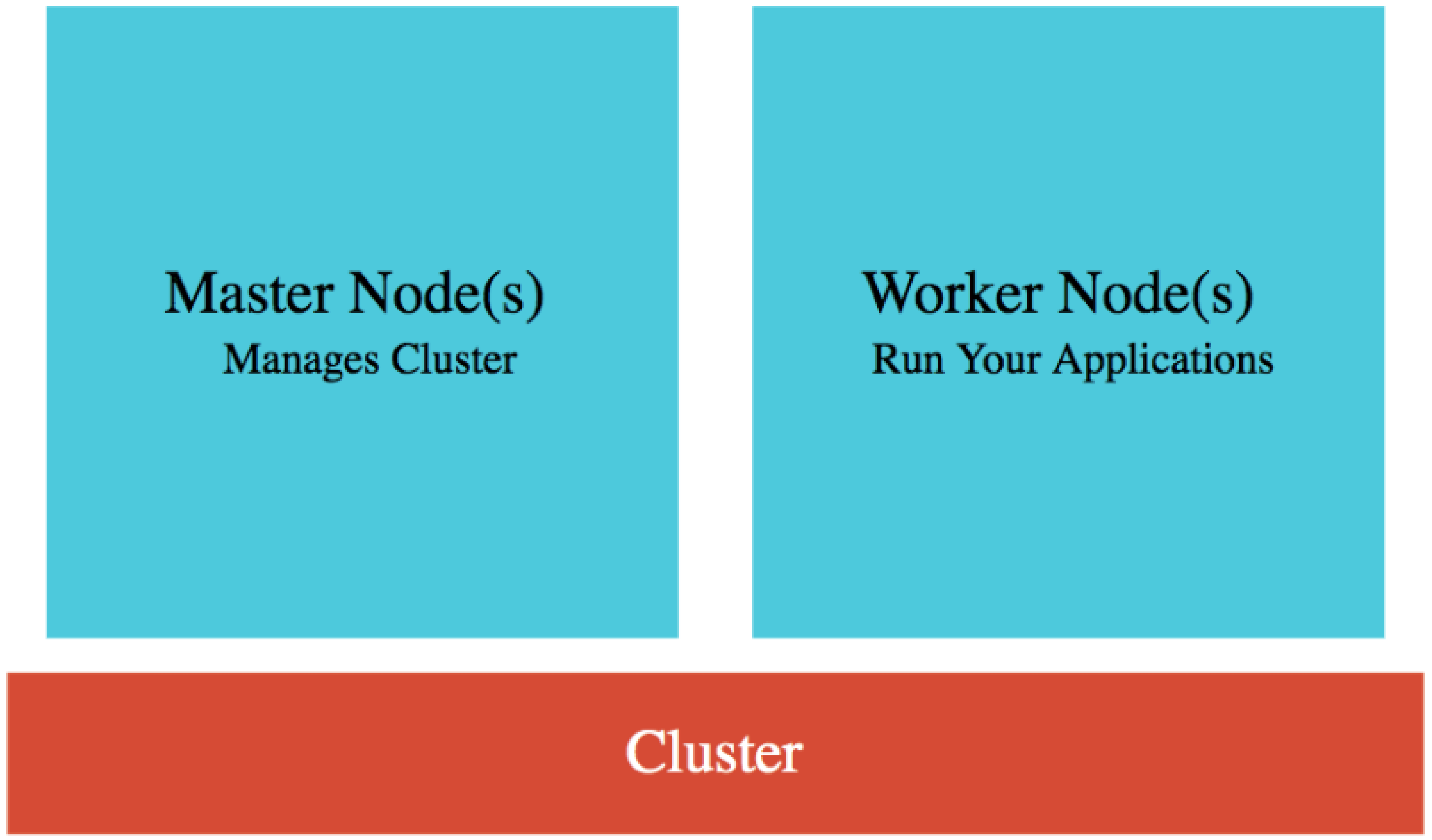
Container Orchestration Options

- **AWS Specific**
  - AWS Elastic Container Service (ECS)
  - AWS Fargate : Serverless version of AWS ECS
- **Cloud Neutral** - Kubernetes
  - AWS - Elastic Kubernetes Service (EKS)
  - Azure - Azure Kubernetes Service (AKS)
  - GCP - Google Kubernetes Engine (GKE)
  - EKS/AKS does not have a free tier!
    - We use GCP and GKE!

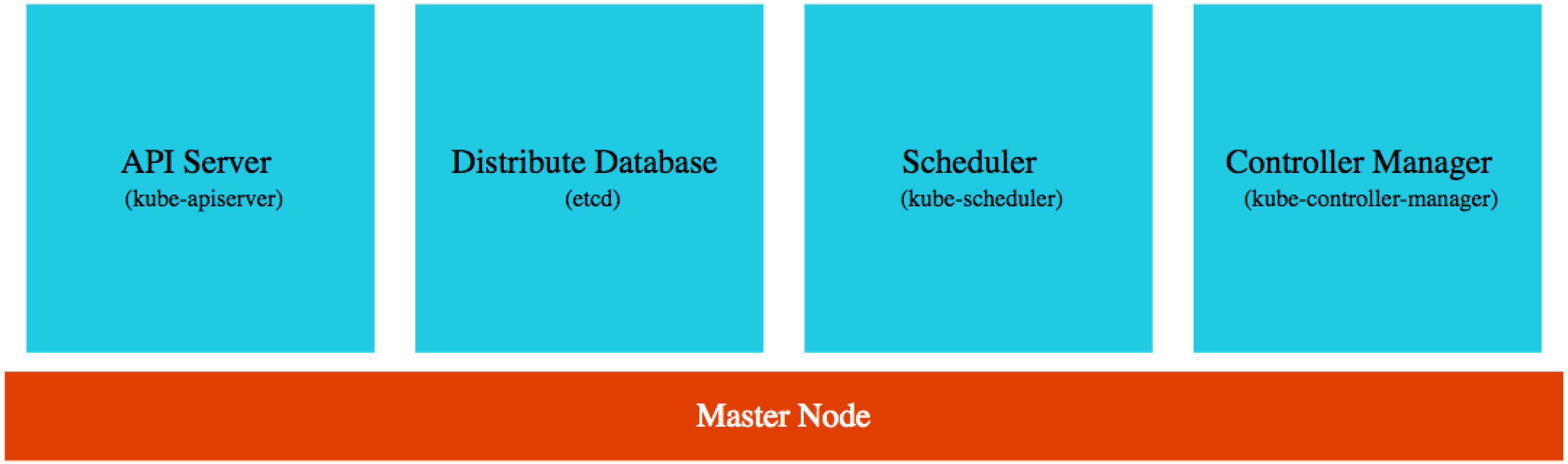



Container 1
Container 2
Container 3
Container 4
Pod 1
Pod 2
Node

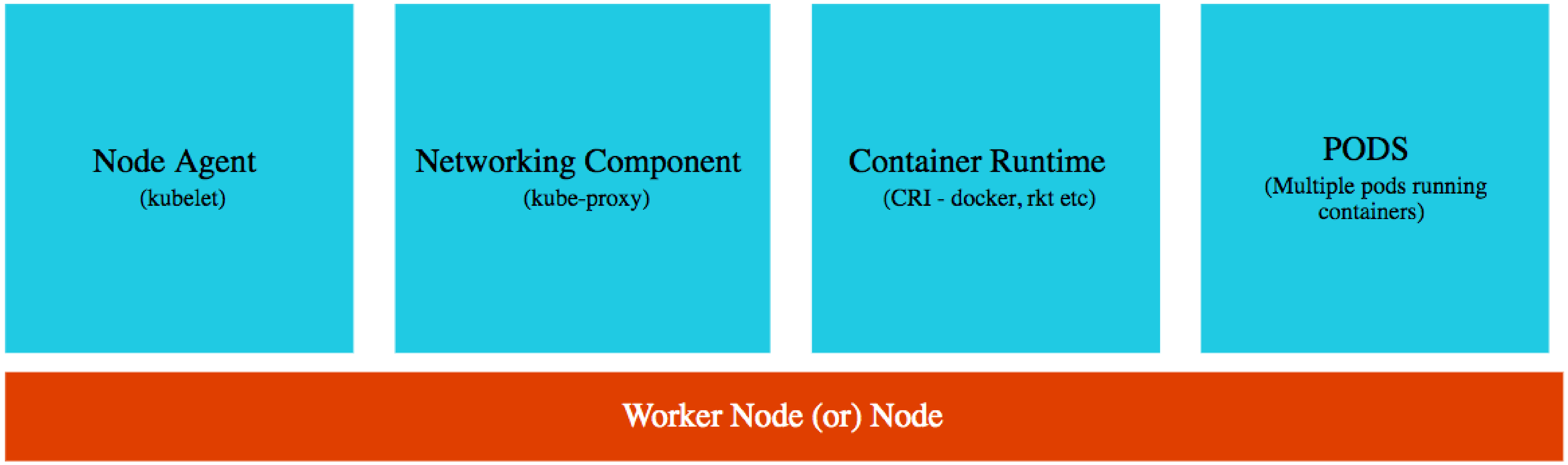
Kubernetes Architecture



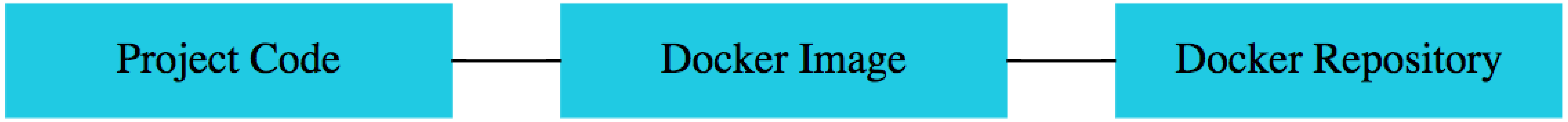
Kubernetes Architecture



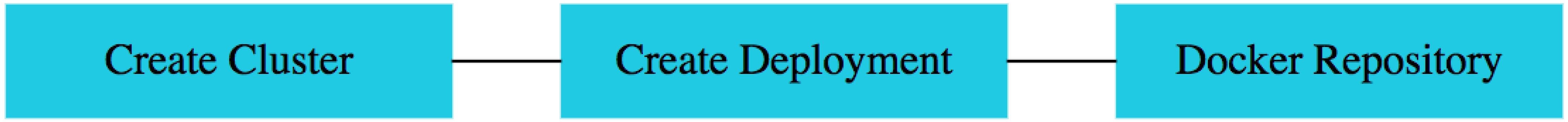
Kubernetes Architecture



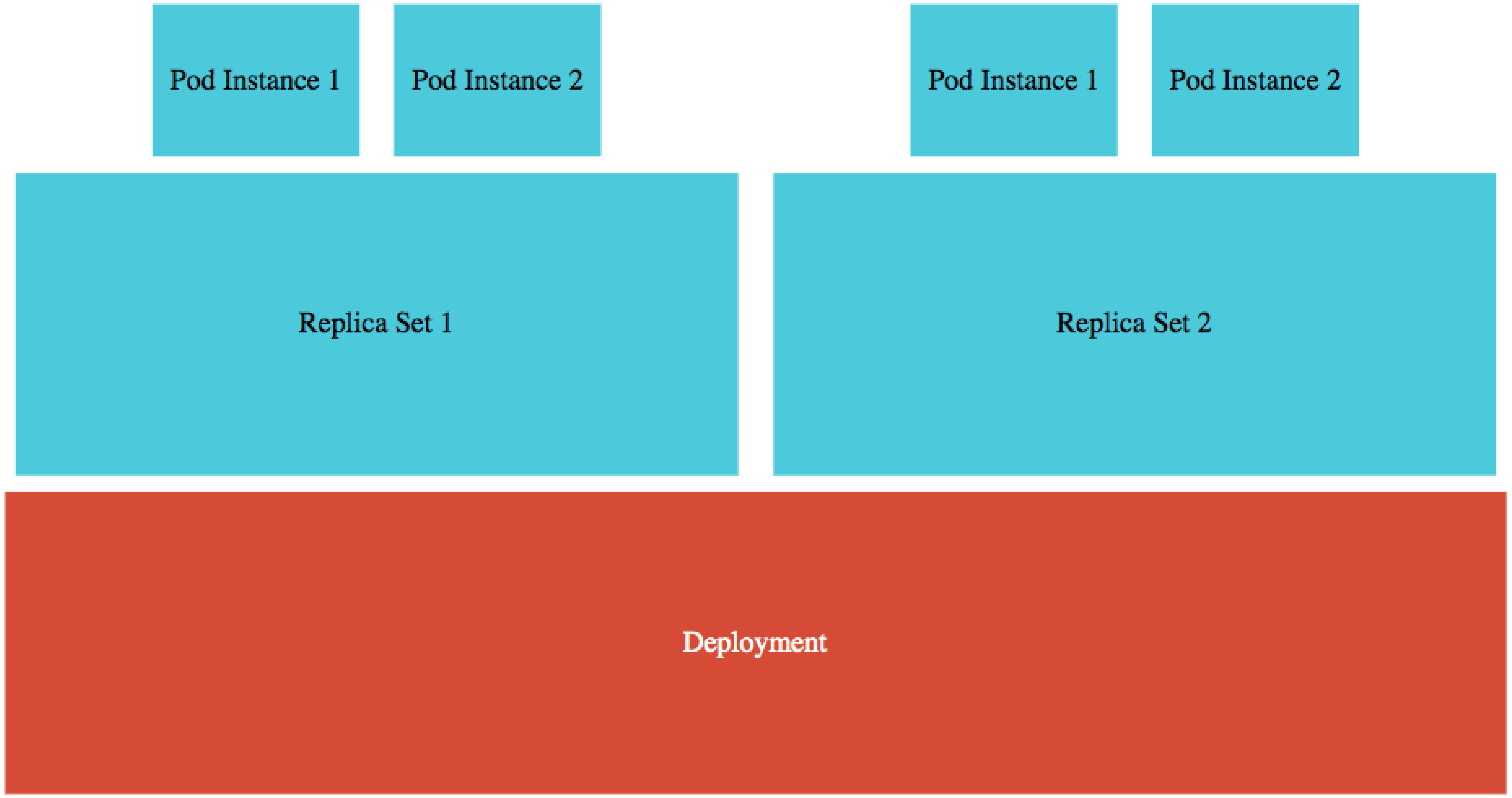
Kubernetes Architecture



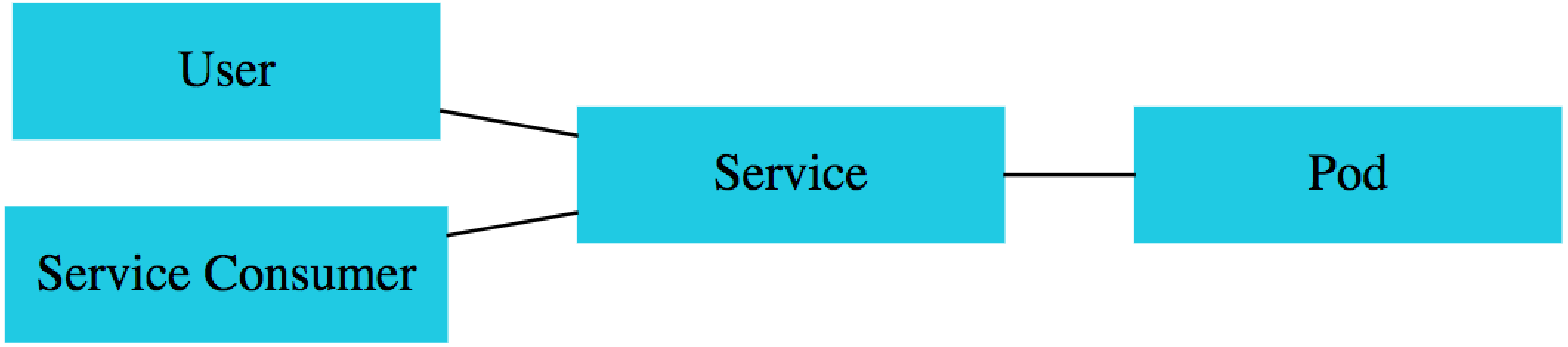
Kubernete Deployments



Kubernete Deployments



Kubernete Deployments



Kubernete Service

Kubernetes - Liveness and Readiness Probes

Microservice1
Microservice2
Microservice3
Microservice4
Microservice5

- Kubernetes uses probes to check the health of a microservice:
  - If readiness probe is not successful, no traffic is sent
  - If liveness probe is not successful, pod is restarted
- Spring Boot Actuator (>=2.3) provides inbuilt readiness and liveness probes:
  - /health/readiness
  - /health/liveness

What Next?

Docker & Kubernetes in Depth

Master Docker with Java - DevOps for Spring Microservices

Create Docker Containers for Java Spring Boot Microservices. DevOps with Docker and Docker Compose for Java Developers.

in28Minutes Official

4.6 ★★★★☆ (970)

6.5 total hours • 62 lectures • All Levels

Master Kubernetes with Docker on Google Cloud, AWS & Azure

Learn Kubernetes and Docker on Google Cloud GKE, AWS EKS & Azure AKS deploying Microservices (Spring Boot + Java)

in28Minutes Official

4.6 ★★★★☆ (708)

13 total hours • 136 lectures • All Levels

Full Stack

Go Java Full Stack with Spring Boot and React

Build Your First Java Full Stack Application with React & Spring Boot. Become a Java Full Stack Java Web Developer Now!

in28Minutes Official

4.4 ★★★★☆ (1,594)

12 total hours • 135 lectures • Beginner

Go Java Full Stack with Spring Boot and Angular

Become a Full Stack Java Developer. Build Your First Java Full Stack Application with Angular and Spring Boot.

in28Minutes Official

4.5 ★★★★☆ (4,012)

11 total hours • 124 lectures • All Levels

AWS Certifications

[NEW] AWS Certified Developer Associate - Step by Step

AWS Certified Developer Associate - Get **AWS** Certified.

in28Minutes Official

4.6 ★★★★☆ (85)

33.5 total hours • 518 lectures • All Levels

Hot & new

AWS Certified Solutions Architect Associate - Step by Step

Become **AWS Certified** Solutions Architect Associate (SAA-C02 - **AWS Certified** Solutions Architect Associate

in28Minutes Official

4.5 ★★★★☆ (1,118)

27.5 total hours • 419 lectures • All Levels

Serverless

Go Serverless with AWS Lambda and Azure Functions

Go Serverless with AWS Lambda & Azure Functions. Build Serverless Apps with SAM & Serverless Framework.

in28Minutes Official

4.9 ★★★★★ (5)

12 total hours • 138 lectures • All Levels

Hot & new