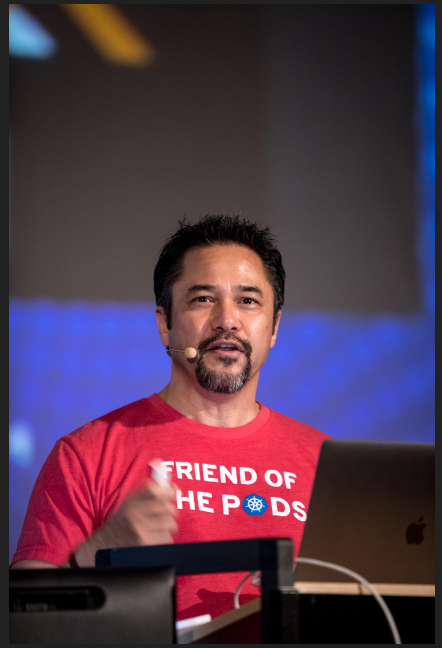

# 9 Steps to Awesome with Kubernetes

Burr Sutter (burrsutter.com)
burrsutter@gmail.com
https://github.com/redhat-scholars/kubernetes-tutorial

# Burr Sutter (burrsutter.com)




- Currently Red Hat's Global Director of Developer Experience
- Featured speaker at technology events around the globe
- A Java Champion since 2005
- Former President of the Atlanta Java User Group
- Founded the DevNexus conference
- Always looking for technologies that enable developers to deliver better software, faster

# Learning Resources

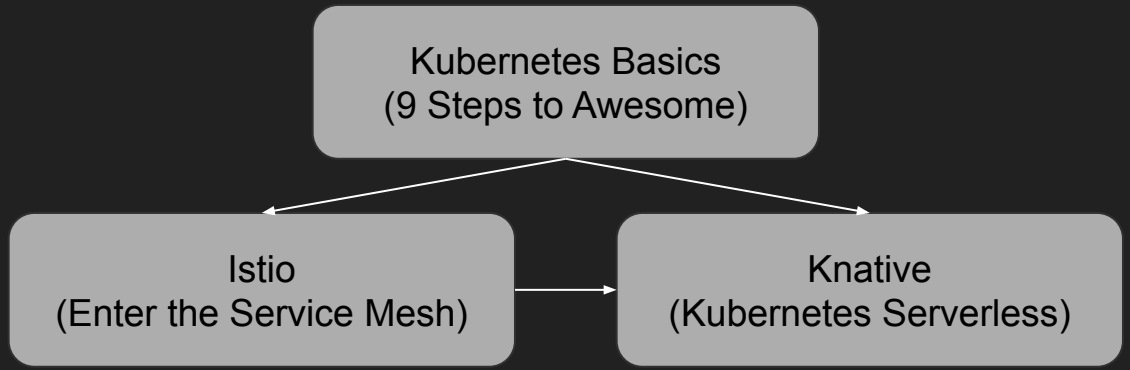
Tutorials

https://dn.dev/containers-tutorial
https://dn.dev/kube-tutorial
https://dn.dev/openshift-tutorial
https://dn.dev/istio-tutorial
https://dn.dev/knative-tutorial
https://dn.dev/quarkus-tutorial
https://dn.dev/kafka-tutorial
https://dn.dev/tekton-tutorial
https://dn.dev/argo-tutorial
https://dn.dev/helm-tutorial

# 3 classes on O'Reilly Live Training

https://www.oreilly.com/live-training/

Control-F and search for "burr"



# Setup

https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/installation.html

Tips on the Virtualization Drivers
https://docs.okd.io/latest/minishift/getting-started/setting-up-virtualization-environment.html

https://minikube.sigs.k8s.io/docs/drivers/

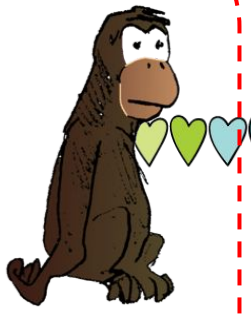
# Your Journey to Awesomeness

Re-Org to DevOps

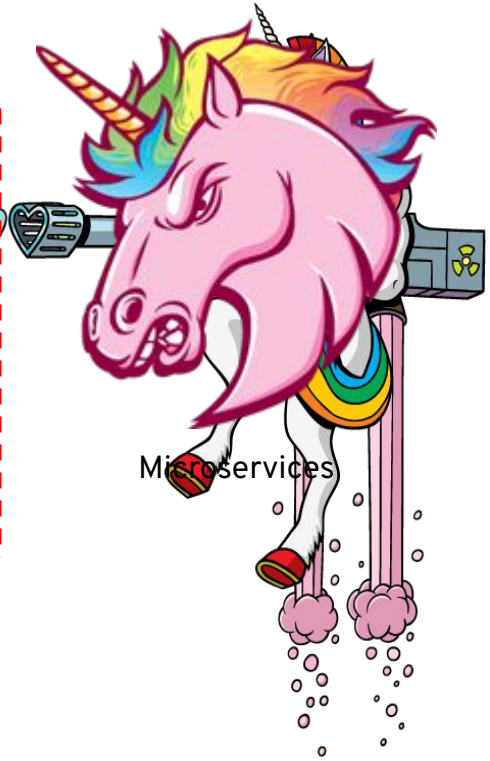
Self-Service, On-Demand, Elastic Infrastructure

Automation

CI & CD Deployment Pipeline



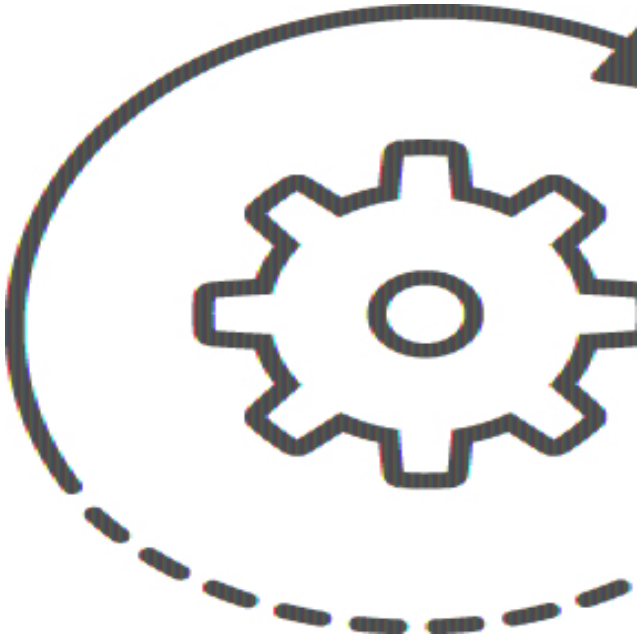
Advanced Deployment Techniques



Microservices

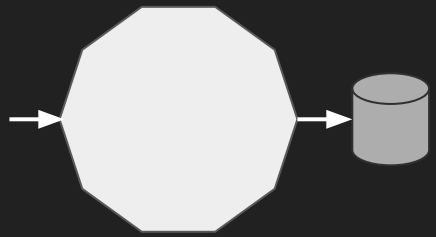
# Agility

Continuous Delivery, Deployment, Improvement



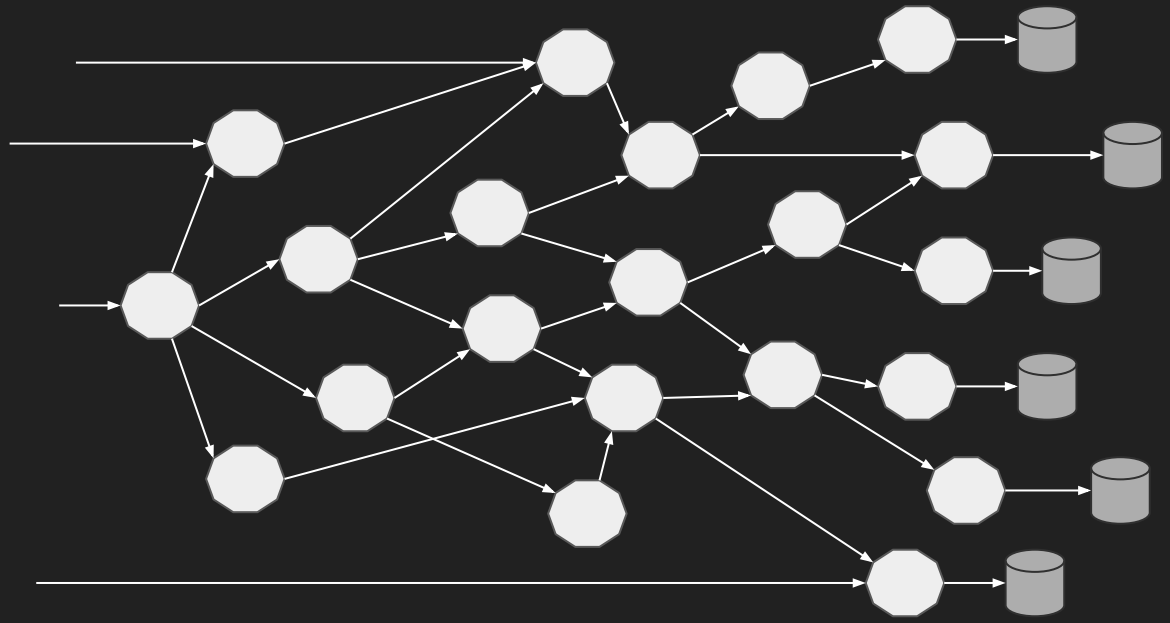
3 Month Deployment Cycle
3 Months BEFORE you gain Feedback and Learn

# Old School



Love Thy Mono

# New School



# 9 Steps

0 - Introduction
1 - Installation & Getting Started
2 - Building Images, Running Containers
3 - Logs
4 - oc/kubectl exec magic
5 - Configuration & Environment
6 - Service discovery & load-balancing
7 - Live & Ready
8 - Rolling updates, Canaries, Blue/Green
~~9 - Debugging~~ Databases
Bonus Items

# Step 0: Introduction

# Email

MyApp.war has been tested with the following

On my Windows 7 desktop

JDK 1.8.43

Wildfly 9

Configuration:

Datasource: MySQLDS

Tested with: mysql-connector-java-5.1.31-bin.jar

Production Environment

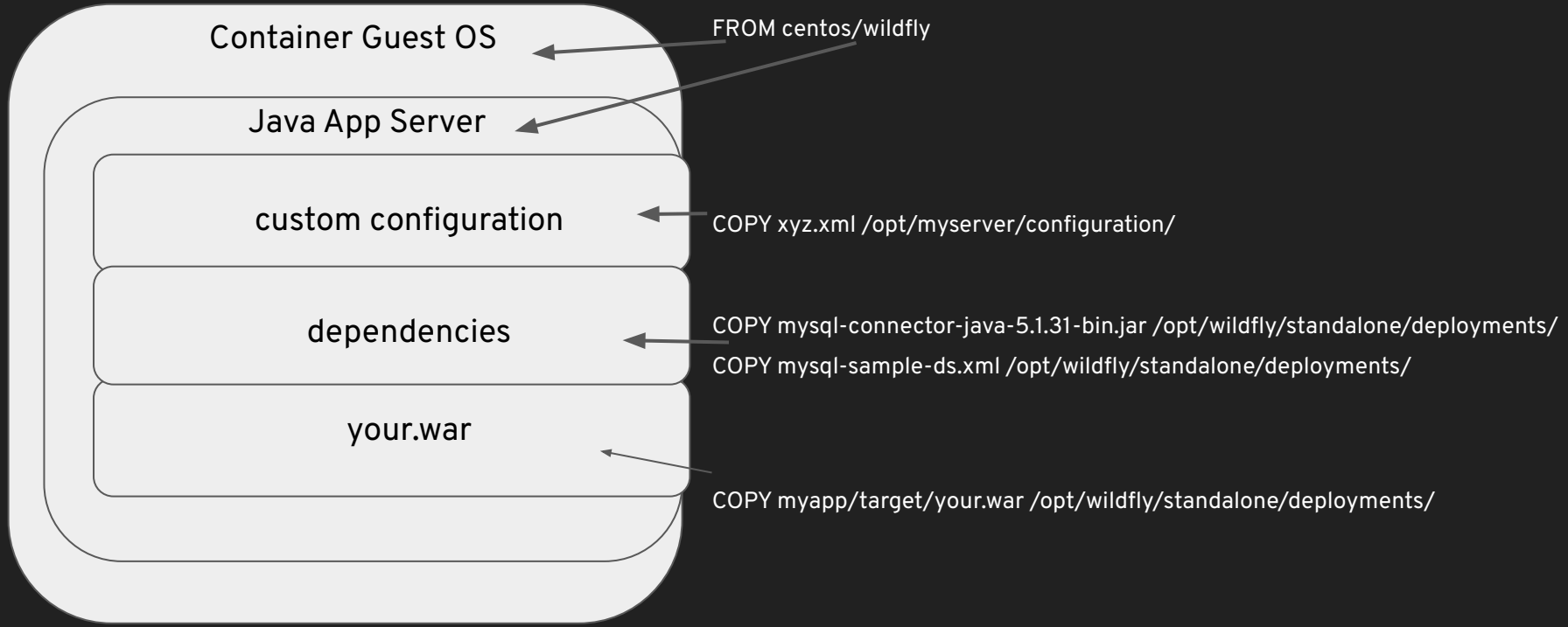
Red Hat Enterprise Linux 6.2

JRE 1.7.3

WebSphere 8.5.5

Oracle 9

# Dockerfile

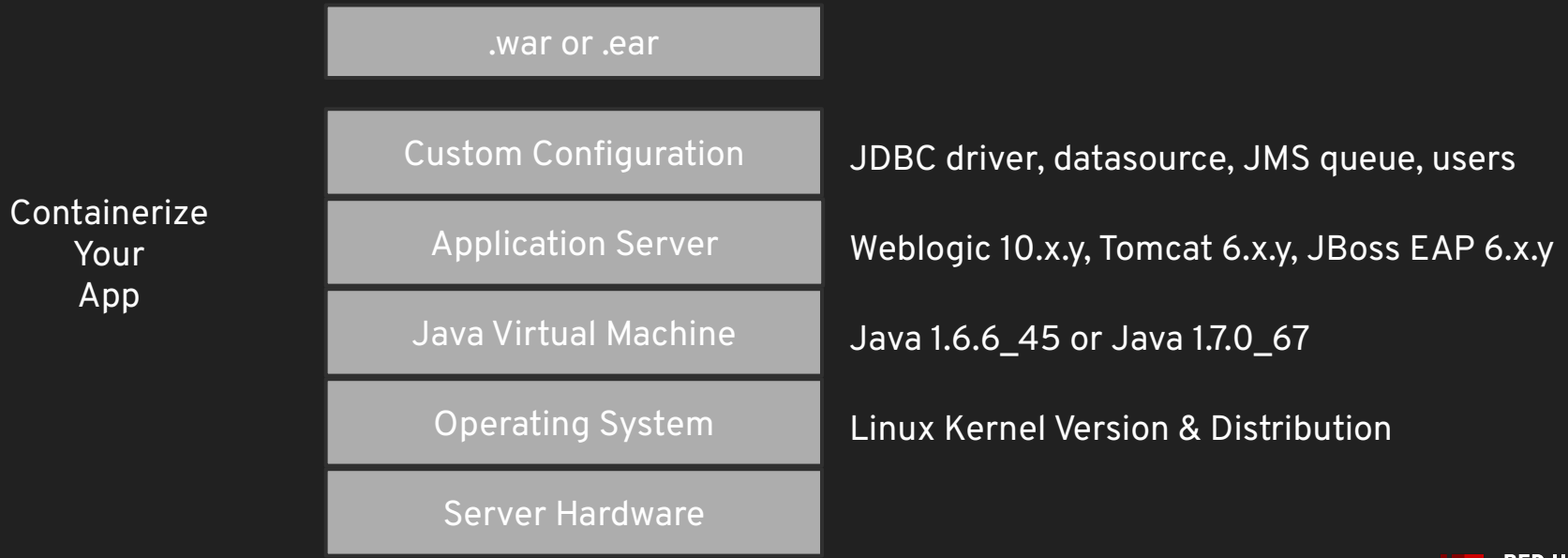


# DevOps Challenges for Multiple Containers

- How to scale?
- How to avoid port conflicts?
- How to manage them on multiple hosts?
- What happens if a host has trouble?
- How to keep them running?
- How to update them?
- Rebuild Container Images?

# A Challenge

Have you ever had “/” vs “\” break your app? Or perhaps needed a unique version of a JDBC driver? Or had a datasource with a slightly misspelled JNDI name? Or received a patch for the JVM or app server that broke your code?



https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

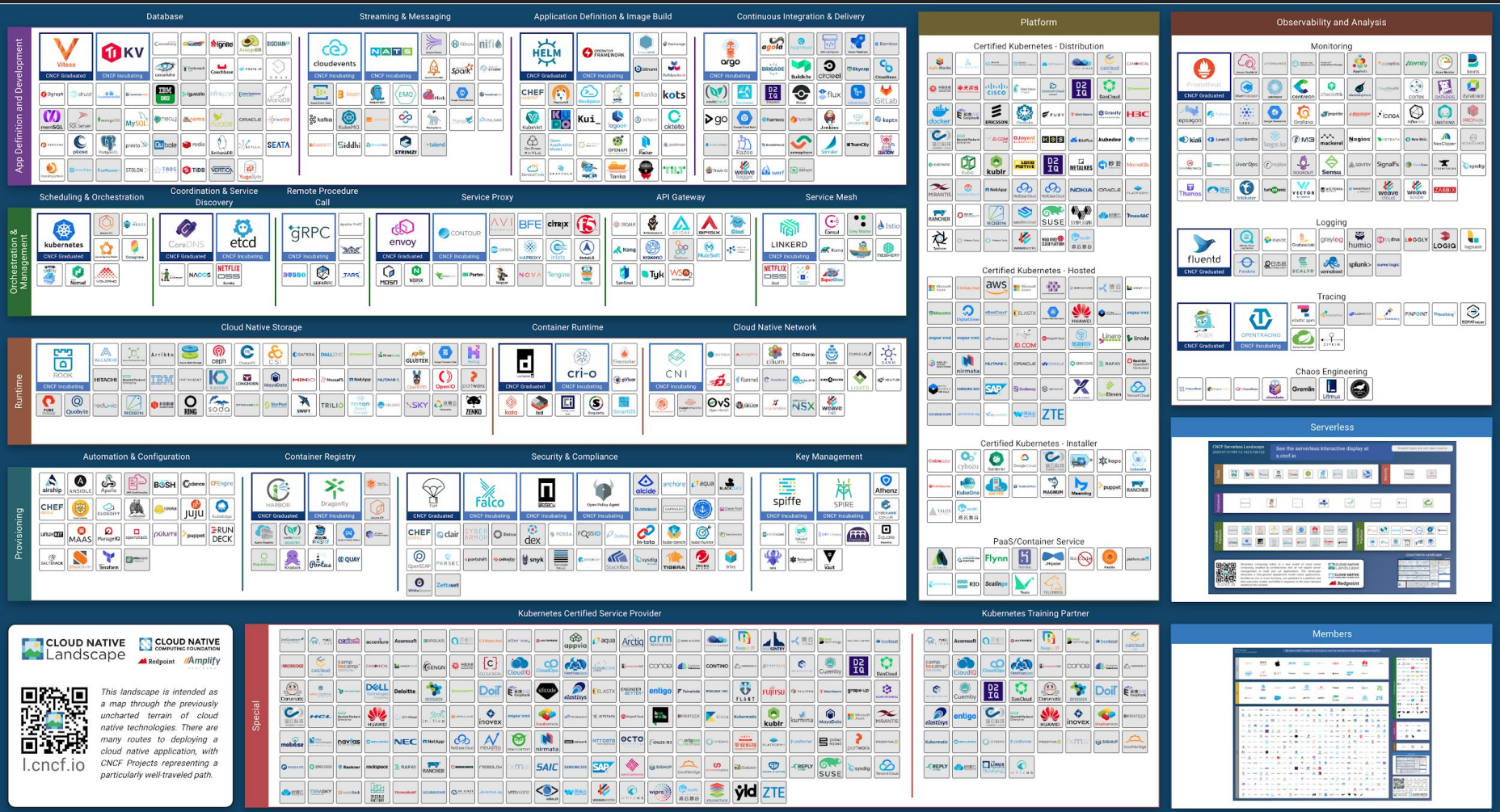
# Meet Kubernetes

- Greek for “Helmsman,” also the root of the word “Governor” (from latin: gubernator)
- Container orchestrator
- Supports multiple cloud and bare-metal environments
- Inspired by Google’s experience with containers
- Open source, written in Go
- Manage applications, not machines

OPENSHIFT

# Key Capabilities

- Self-healing
- Horizontal Manual & Auto Scaling
- Automatic Restarting
- Scheduled across hosts
- Built-in load-balancer
- Rolling upgrades

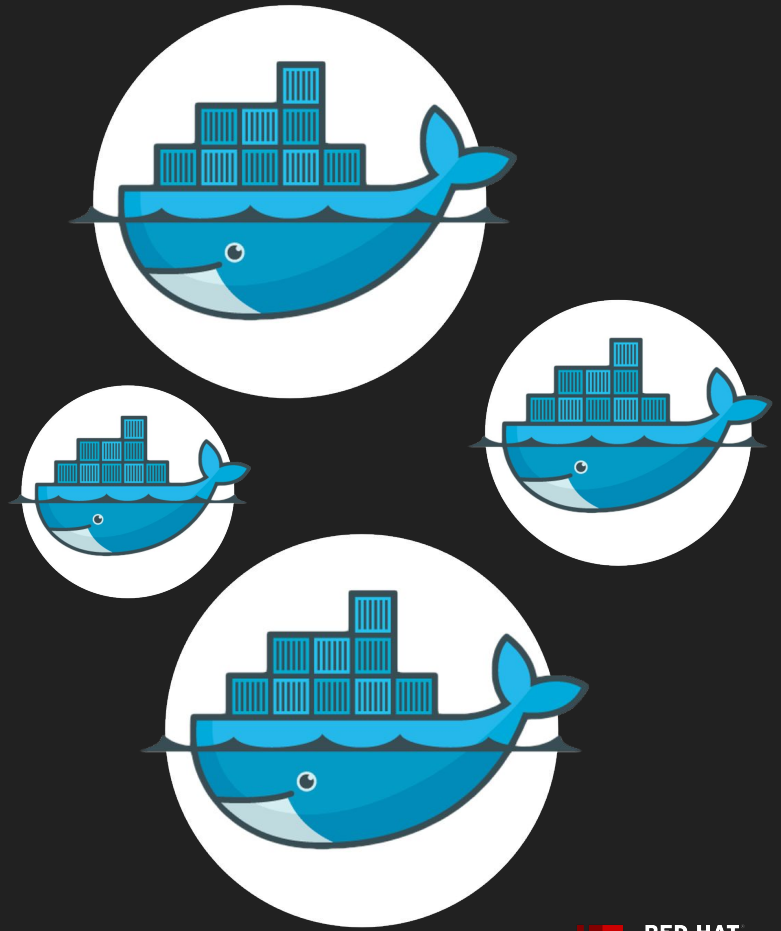

App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Serverless
Members
Kubernetes Certified Service Provider
Special
Kubernetes Training Partner
CLOUD NATIVE Landscape
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify
l.cncf.io
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

# Pods

A group of whales is commonly referred to as a pod and a pod usually consists a group of whales that have bonded together either because of biological reasons or through friendships developed between two or more whales.

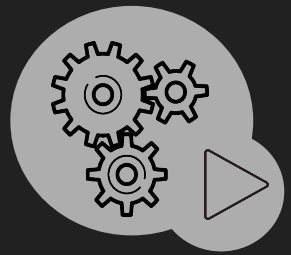
In many cases a typical whale pod consists of anywhere from 2 to 30 whales or more.*

*http://www.whalefacts.org/what-is-a-group-of-whales-called/



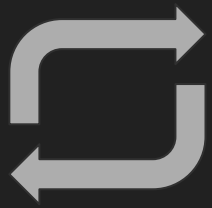
# Kubernetes Terms

## Pod



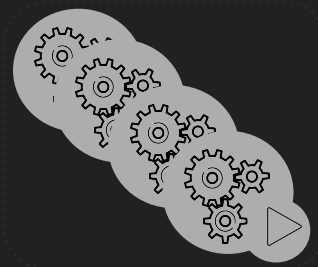
- ✓ 1+ containers
- ✓ Shared IP
- ✓ Shared storage (ephemeral)
- ✓ Shared resources
- ✓ Shared lifecycle

## Replicaset/ Deployment



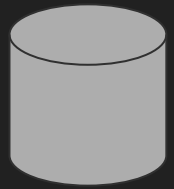
- ✓ The Desired State - replicas, pod template: health checks, resources, image

## Service



- ✓ Grouping of pods (acting as one) has stable virtual IP and DNS name

## Persistent Volume

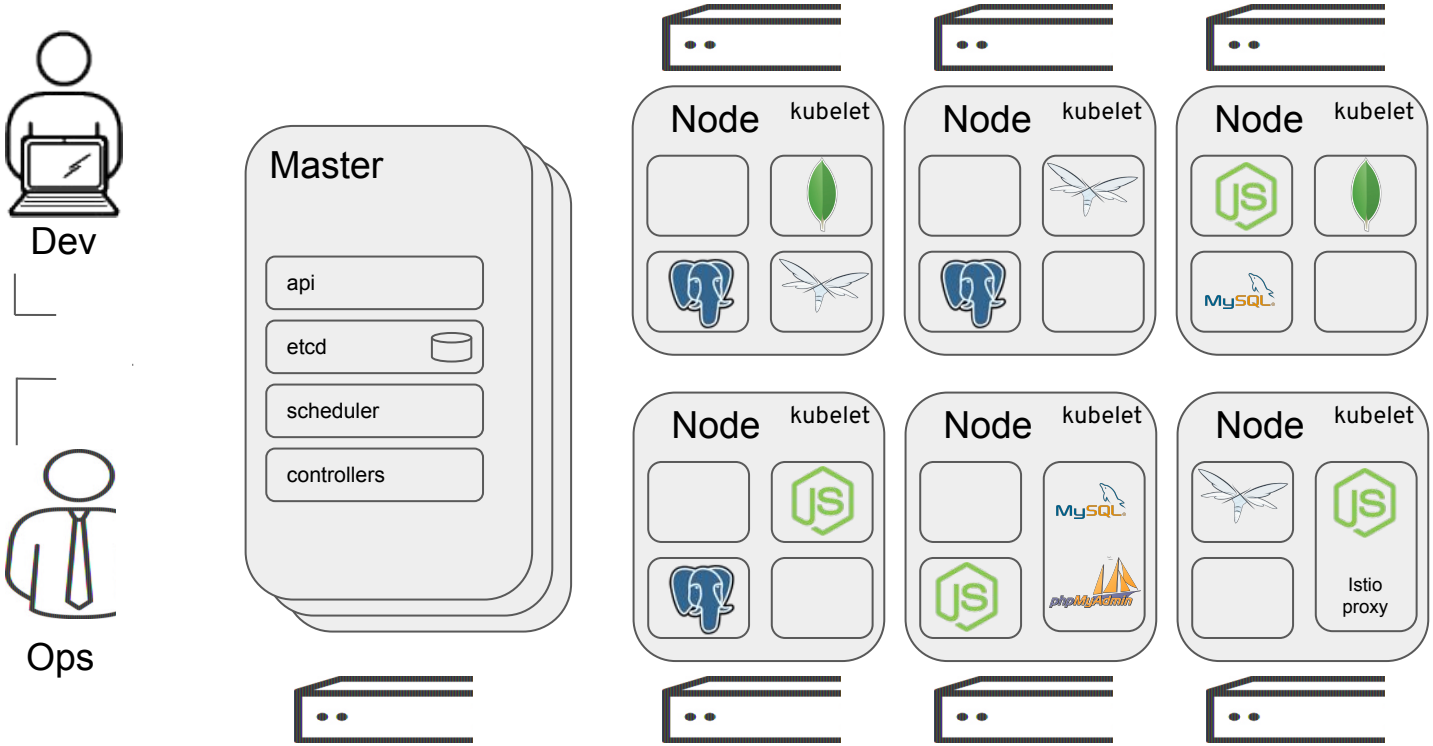


- ✓ Network available storage
- ✓ PVs and PVCs

## Label

- ✓ Key/Value pairs associated with Kubernetes objects (env=production)

# Kubernetes Cluster - Nodes

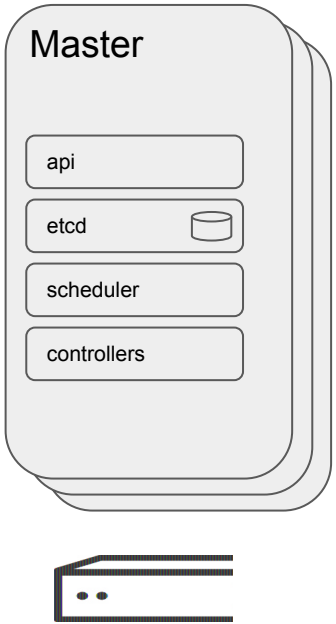


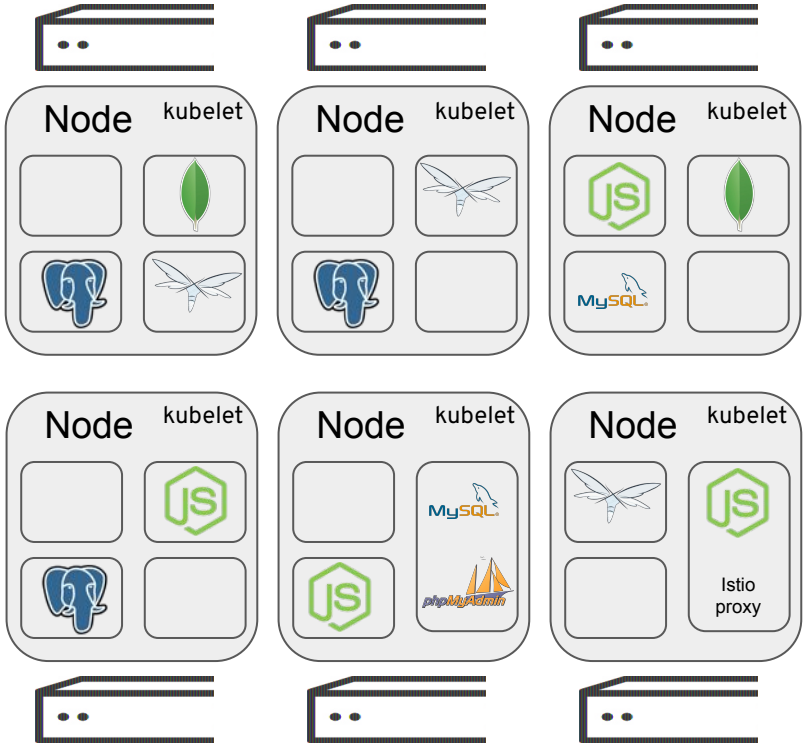
# Kubernetes Cluster - Declarative

Dev

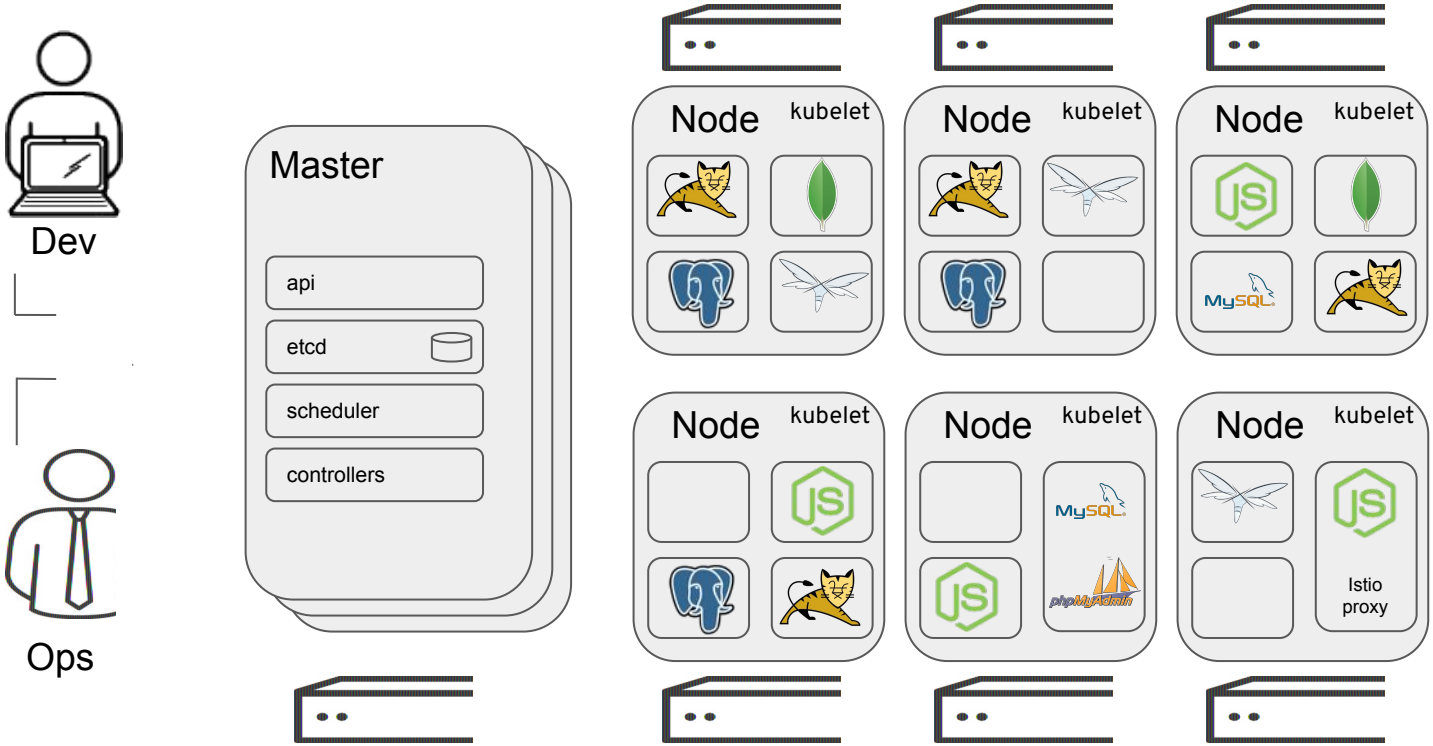
image: repo/mytomcat:v1
replicas: 4

Ops

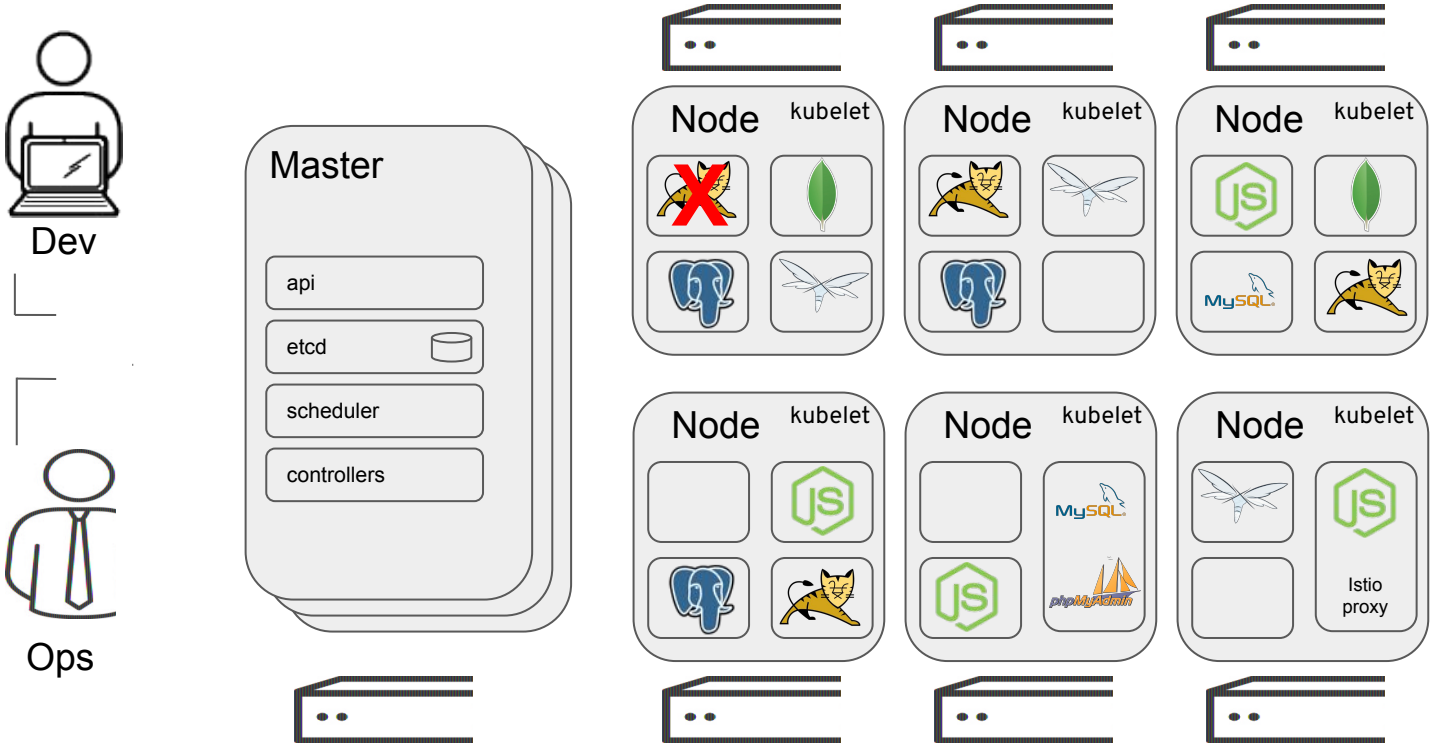


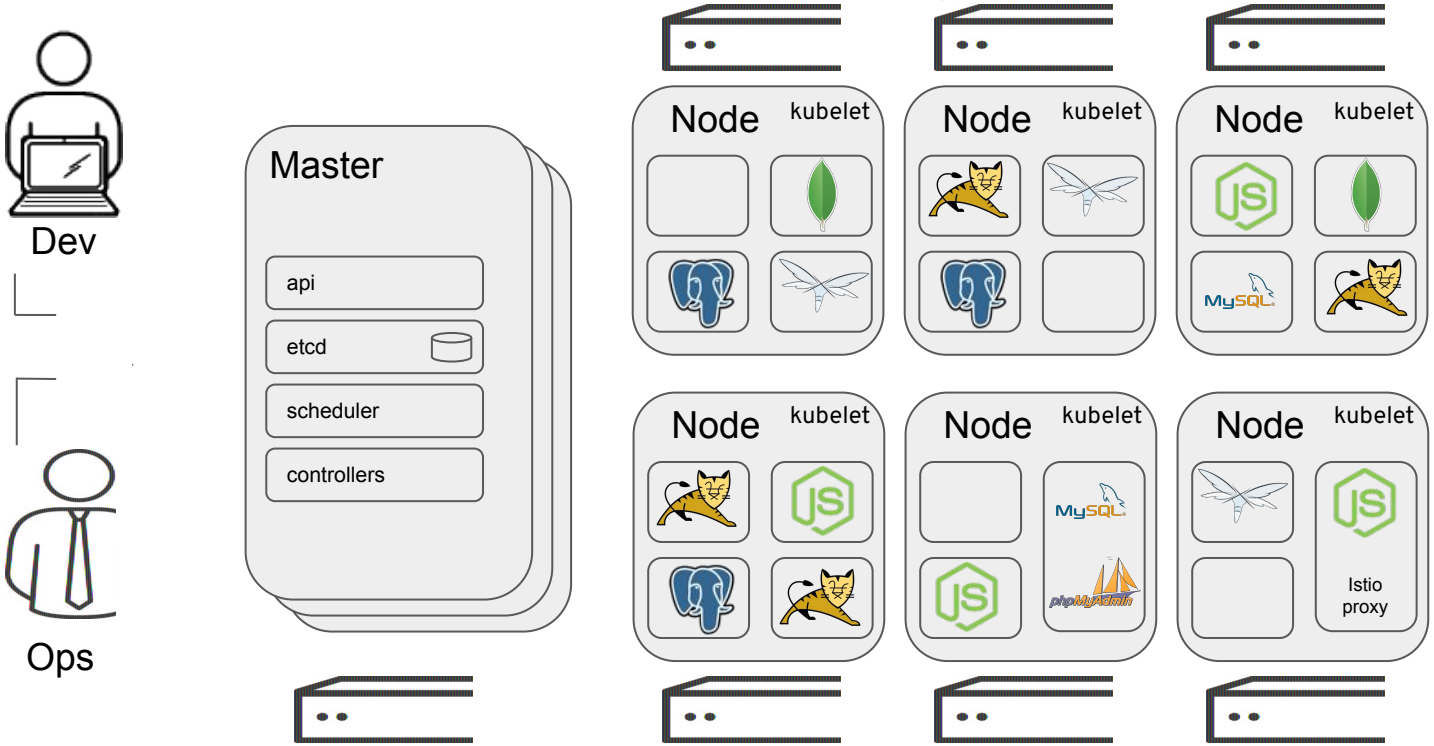
# Kubernetes Cluster - 4 Tomcats



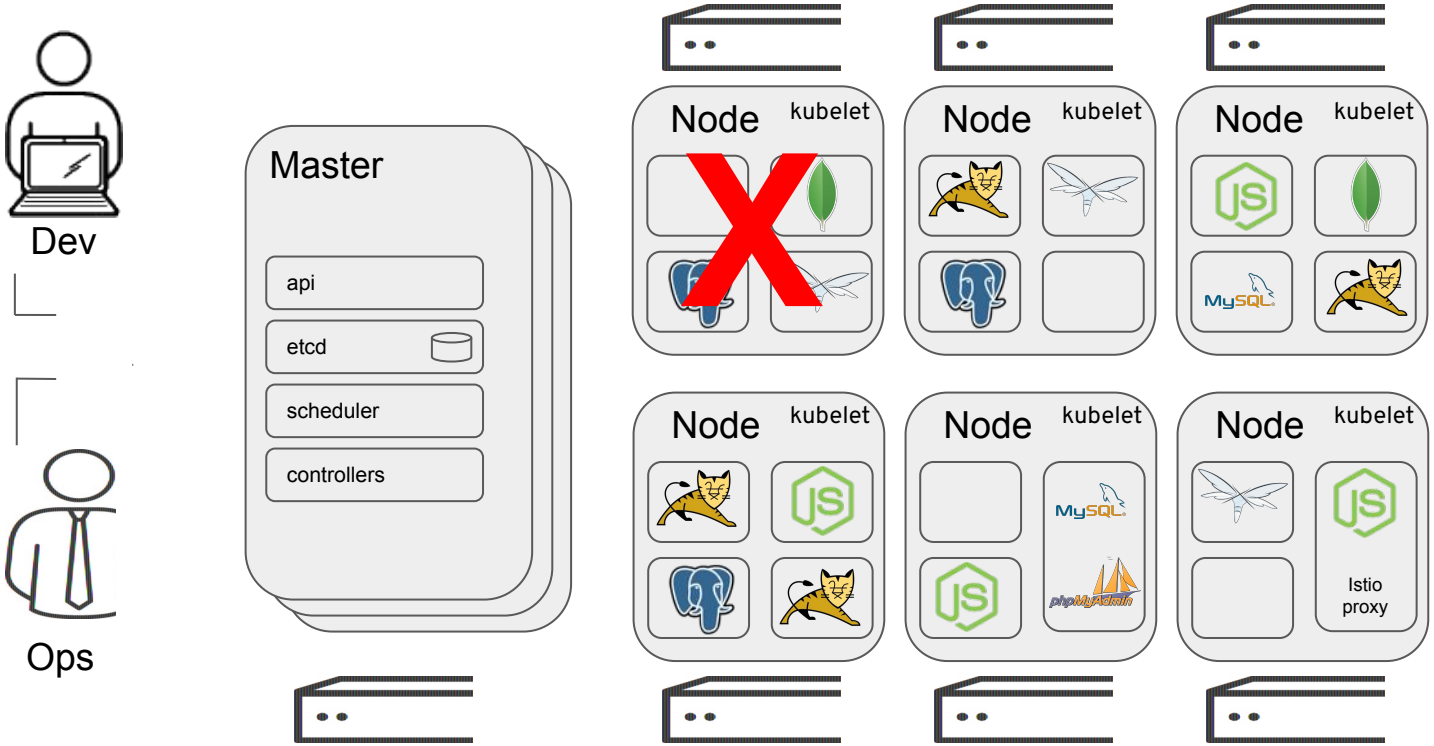
# Kubernetes Cluster - Pod Fail



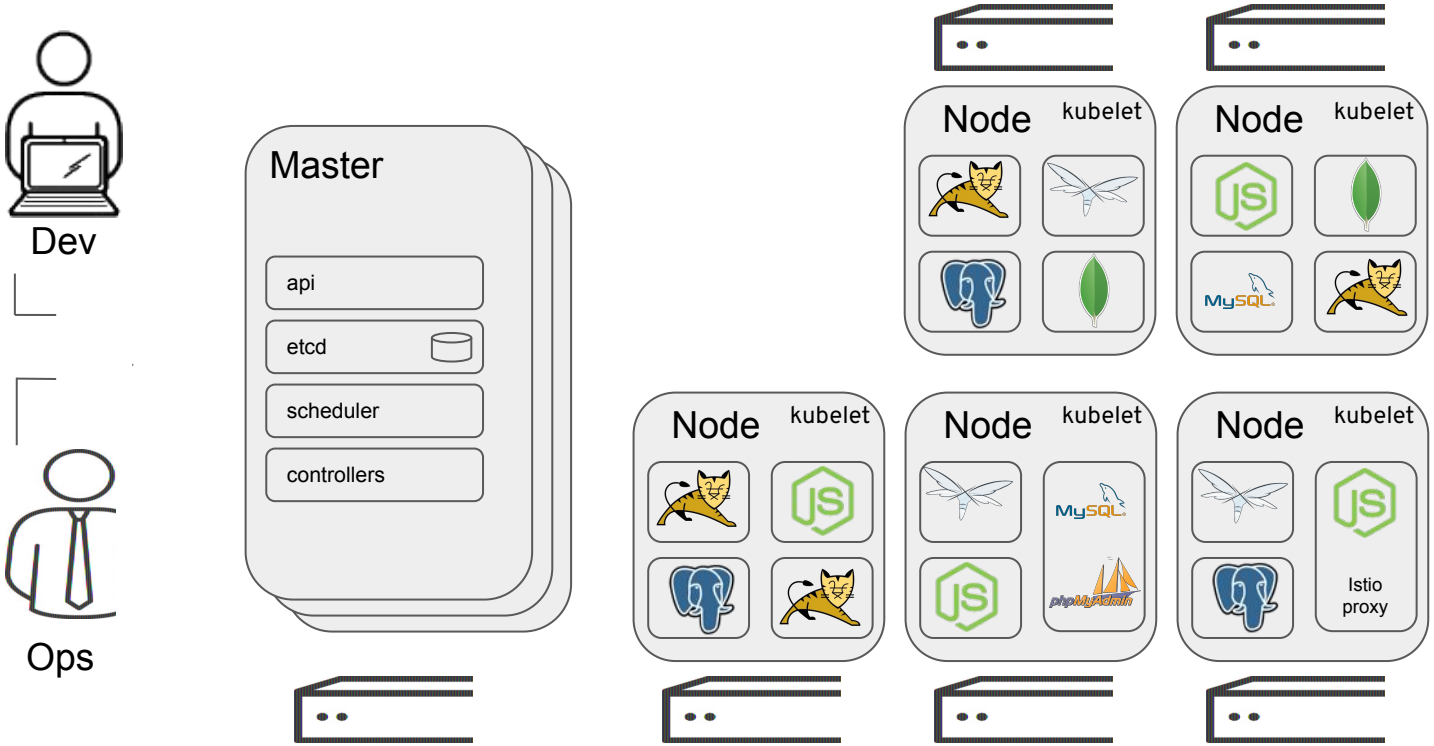
# Kubernetes Cluster - Correcting



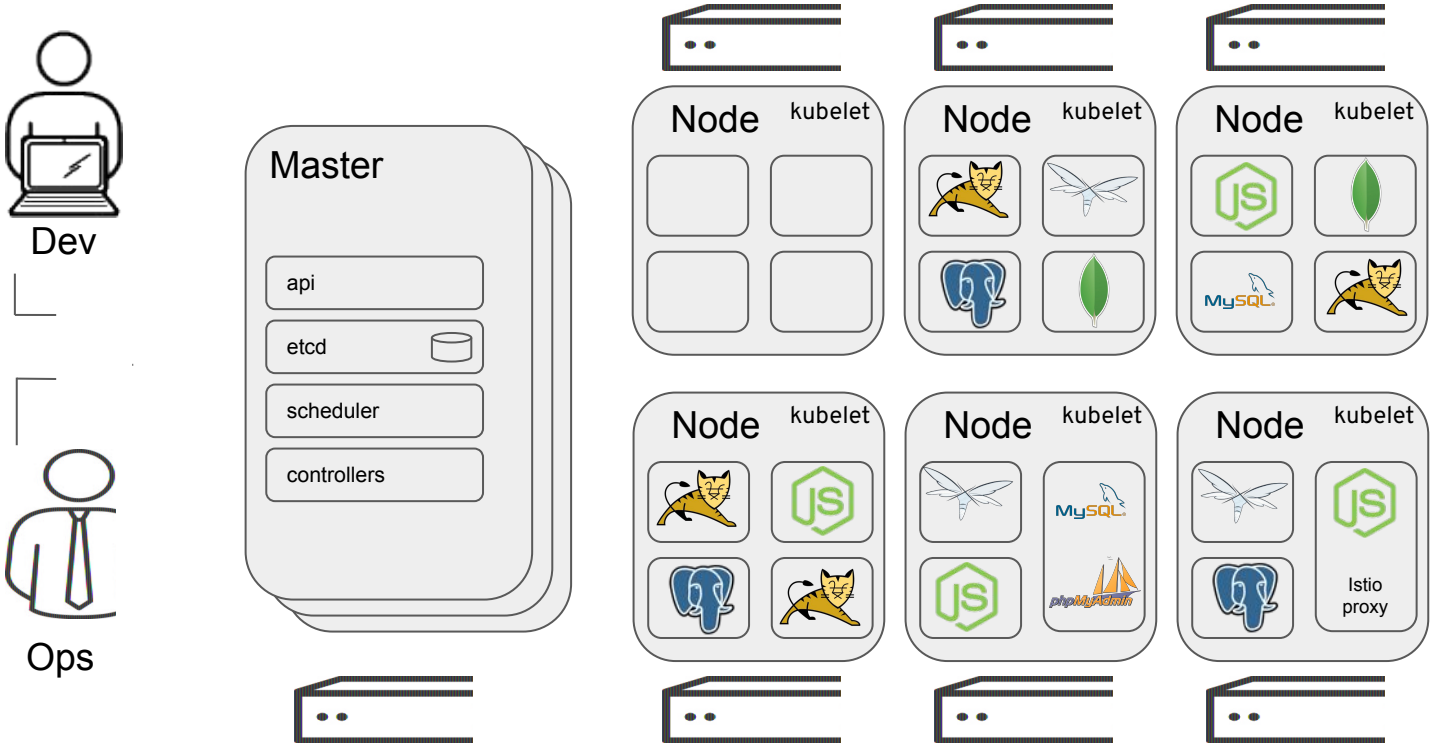
# Kubernetes Cluster - Node Fail



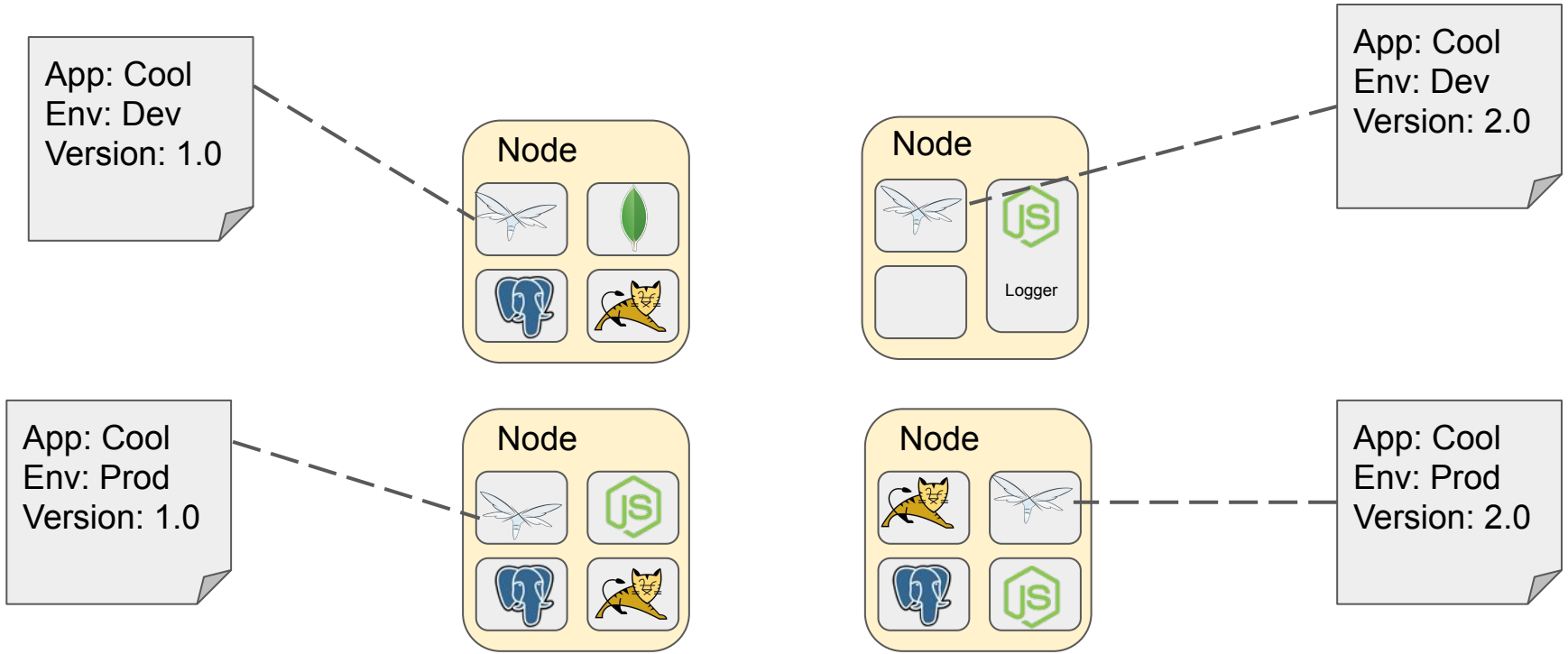
# Kubernetes Cluster



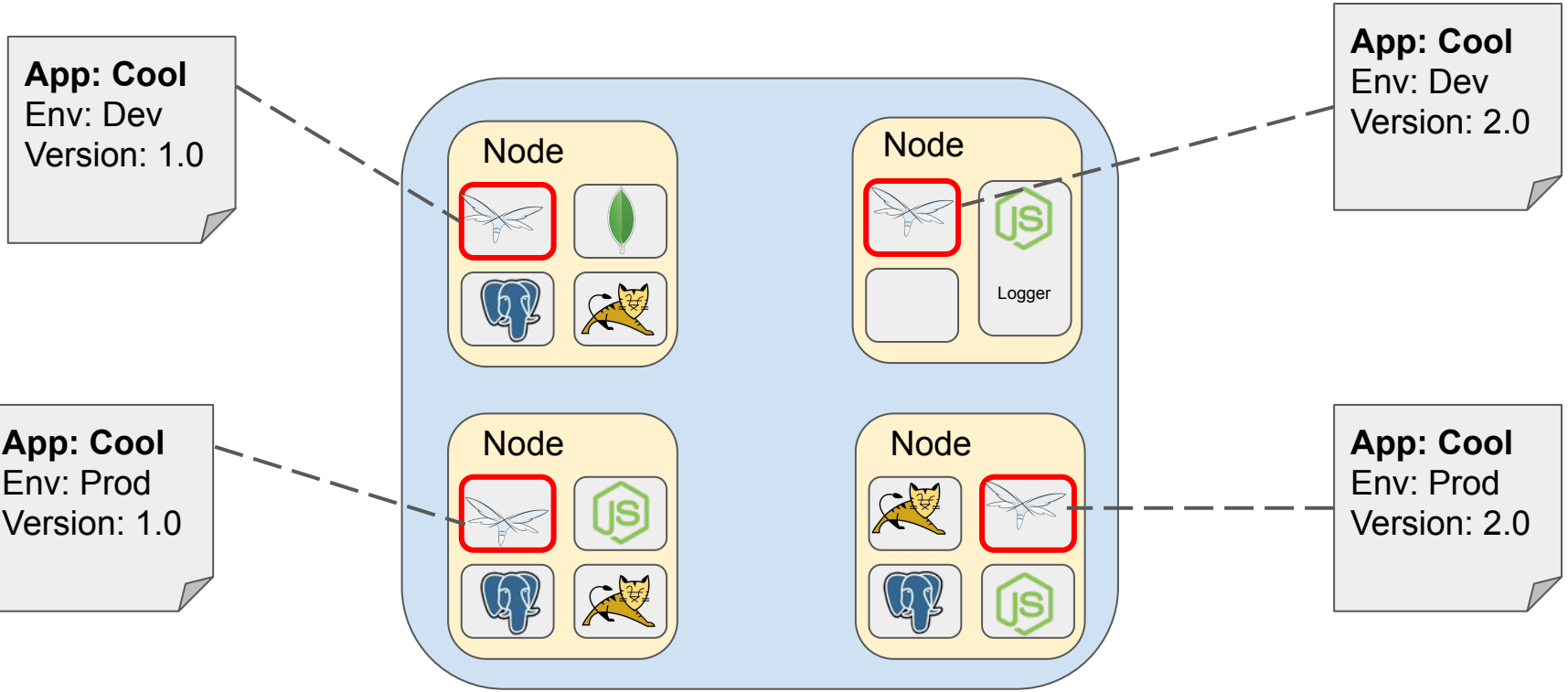
# Kubernetes Cluster - New Node



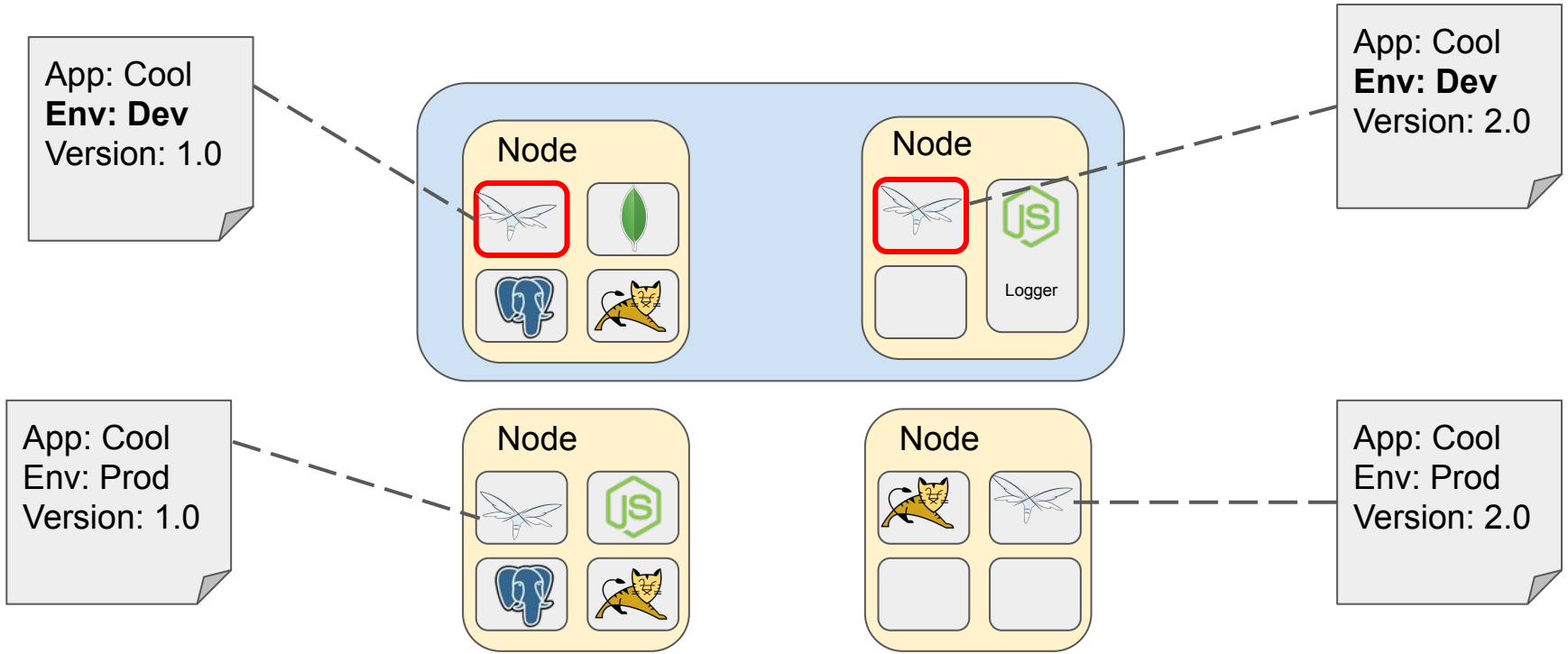
# Labels



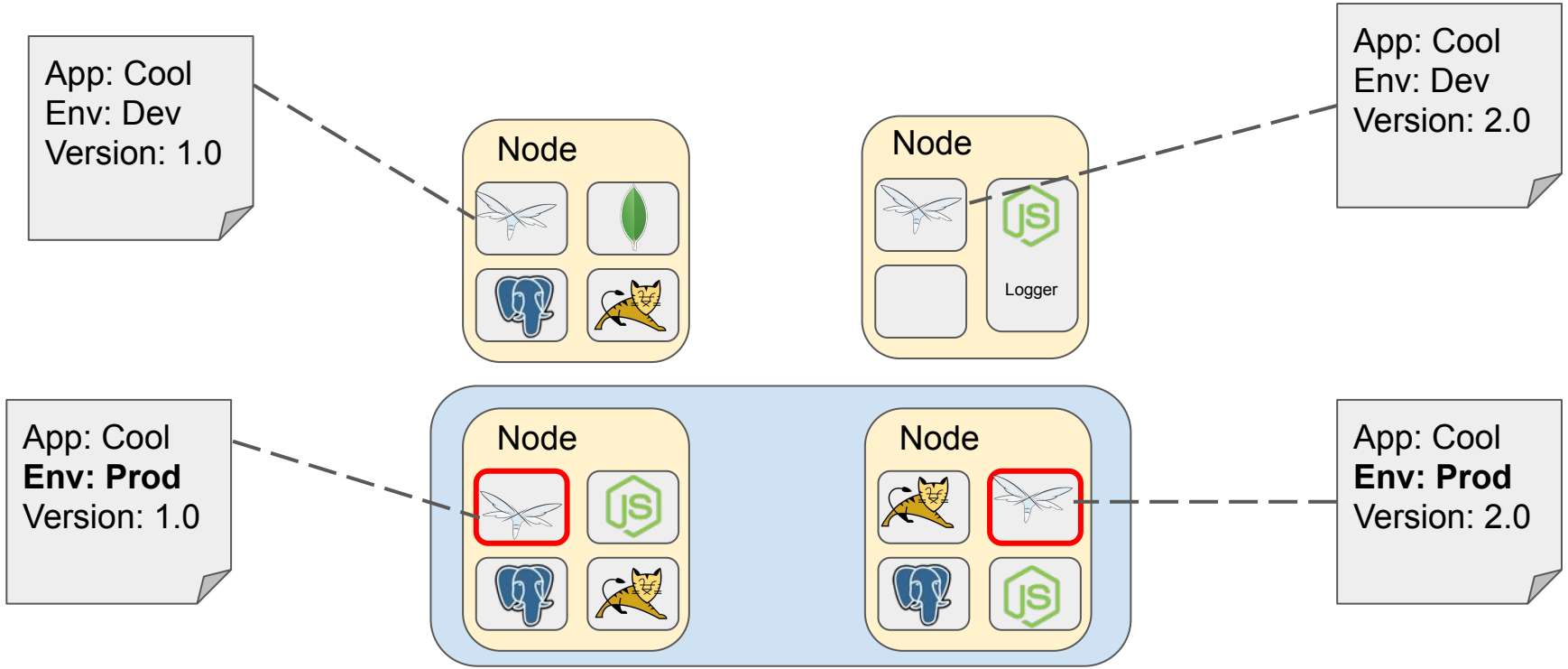
# Labels App:Cool



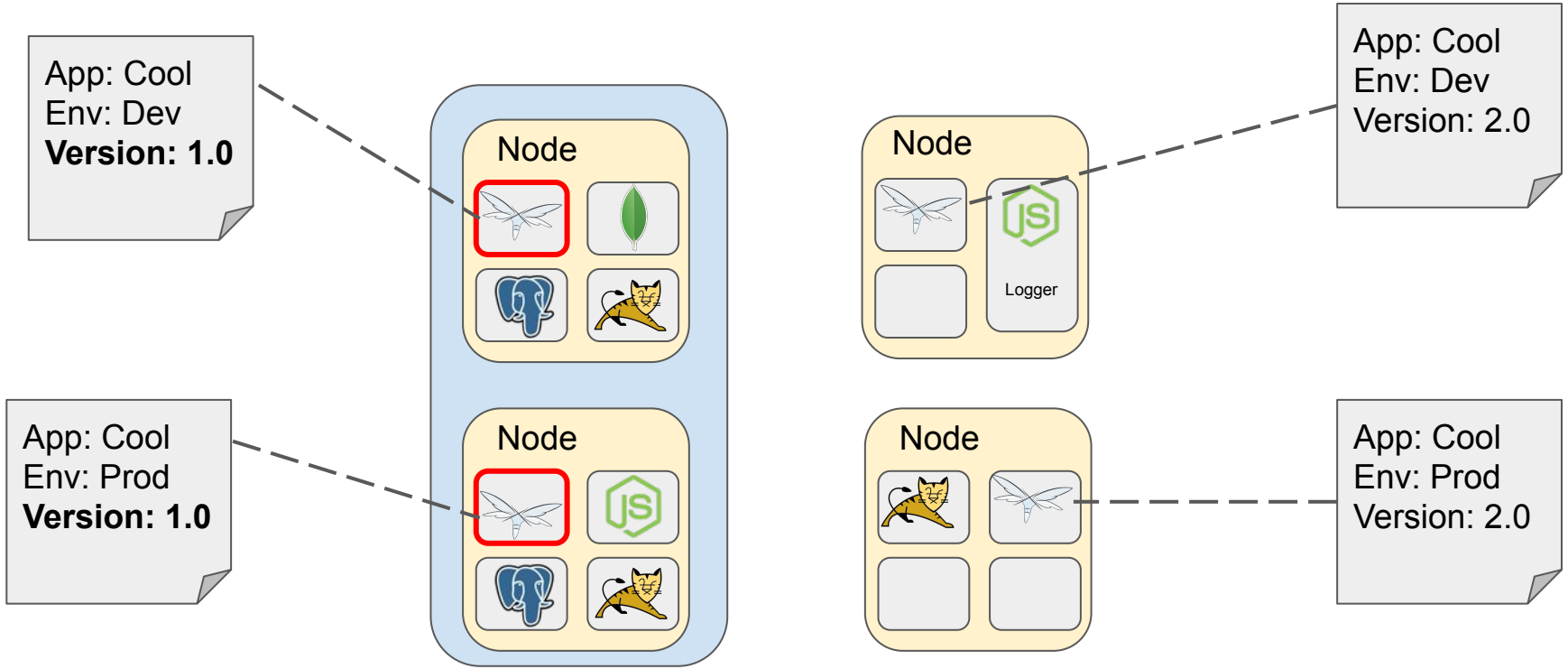
# Labels Env:Dev

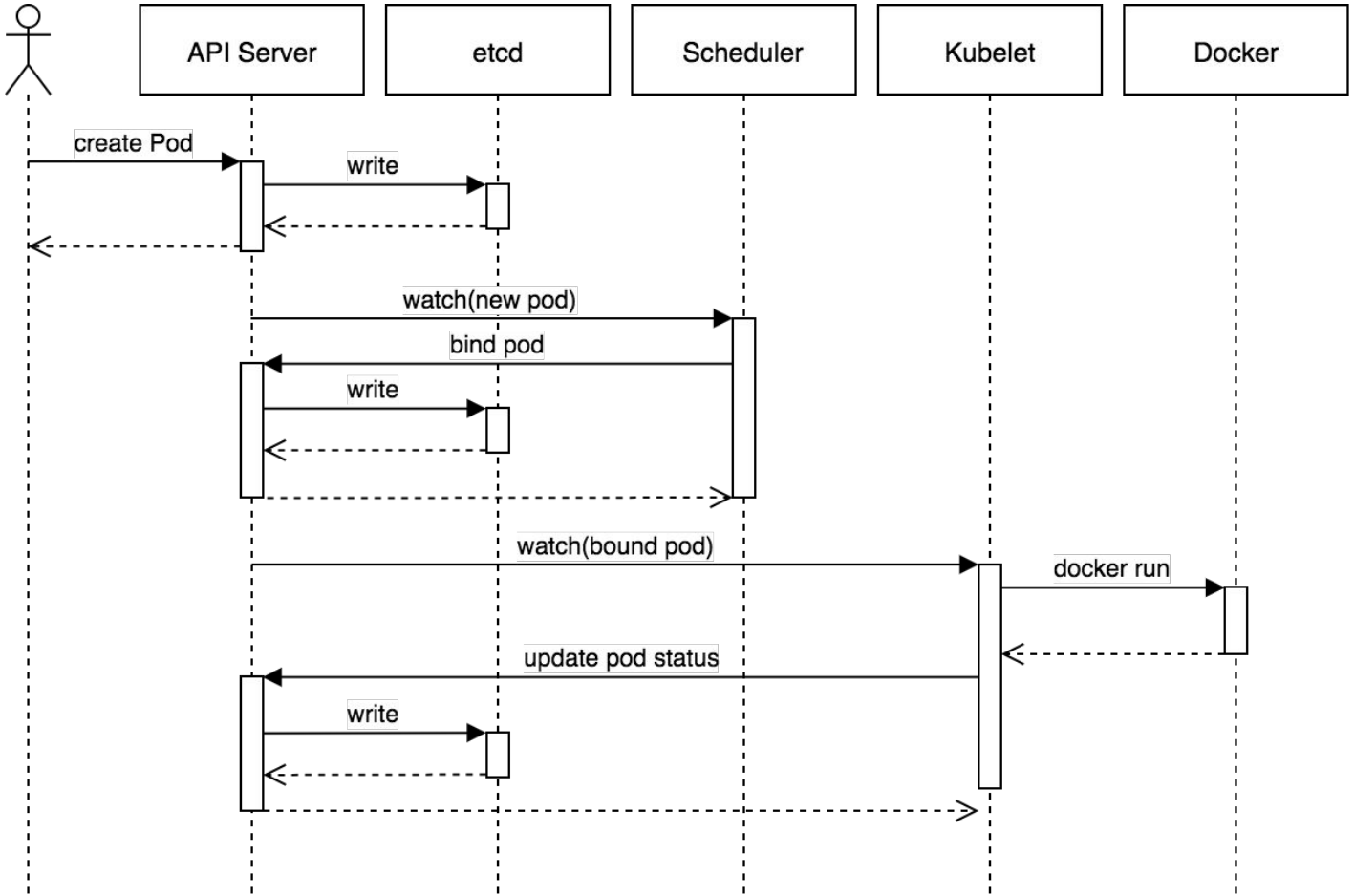


# Labels Env:Prod



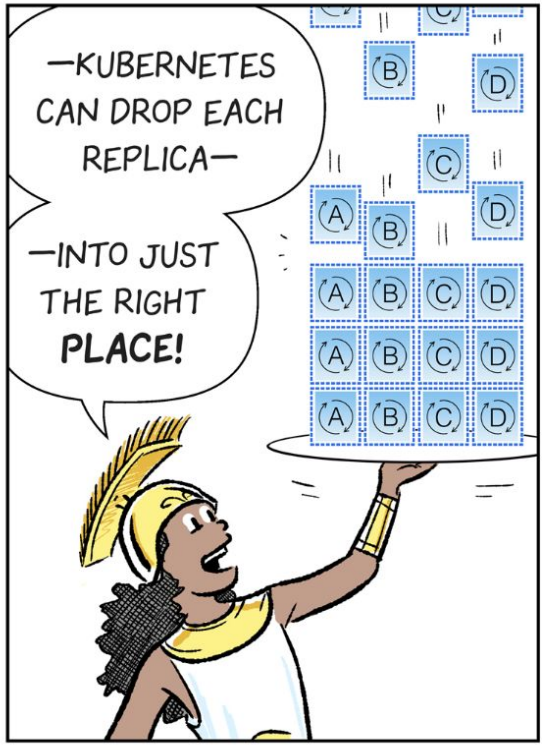
# Labels Version:1.0

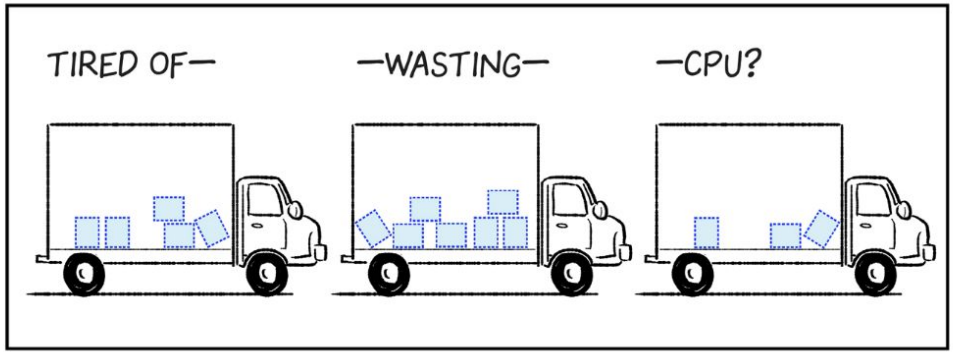




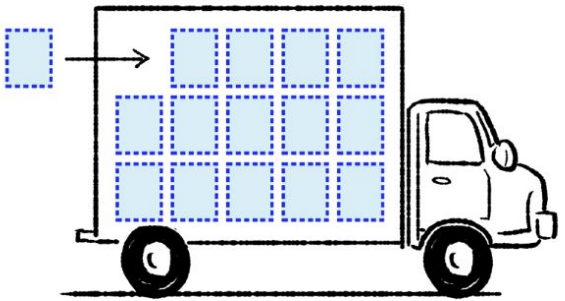

https://dzone.com/articles/kubernetes-lifecycle-of-a-pod
https://blog.heptio.com/core-kubernetes-jazz-improv-over-orchestration-a7903ea92ca

https://cloud.google.com/kubernetes-engine/kubernetes-comic/




KUBERNETES WILL BE ON THE LOOKOUT FOR MORE EFFICIENT **BIN PACKING** OPPORTUNITIES.

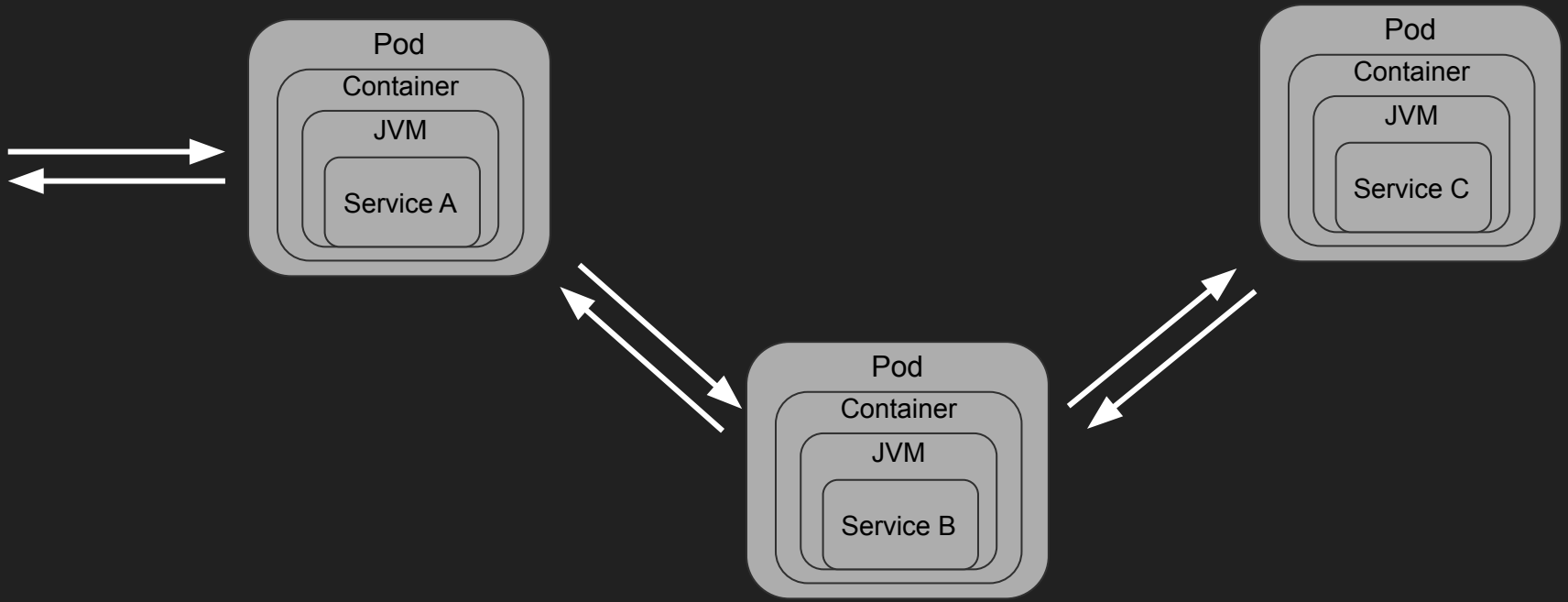


# kubectl commands

https://kubernetes.io/docs/user-guide/kubectl/
https://kubernetes.io/docs/reference/kubectl/cheatsheet

```
kubectl get namespaces

kubectl get pods -n mynamespace

kubectl run myvertx --image=burr/myvertx:v1 --port=8080

kubectl logs myvertx-kk605

kubectl expose deployment --port=8080 myvertx --type=LoadBalancer

kubectl scale deployment myvertx --replicas=3

kubectl set image deployment/myvertx myvertx=burr/myvertx:v2
```

# Microservices == Distributed Computing



# java -jar myapp.jar

## DropWizard

www.dropwizard.io

JAX-RS API

First to market

DropWizard Metrics

## Vert.x

vertx.io

Reactive
Async
non-blocking

RxJava

vertx run myhttp.java

VERT.X

## Spring Boot

spring.io/projects/spring-boot

Spring API
(@RestController)

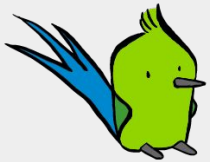
'Starter' POMs:
start.spring.io

## Thorntail

thorntail.io

MicroProfile.io

'Starter' POMs:
thorntail.io/generator



## Micronaut

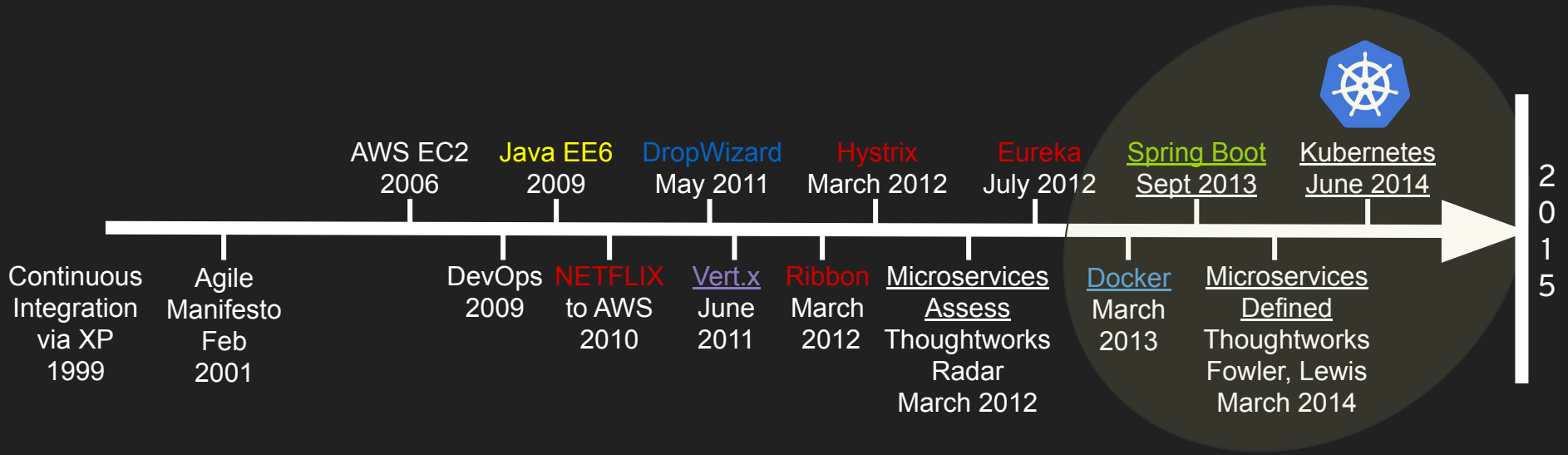
micronaut.io

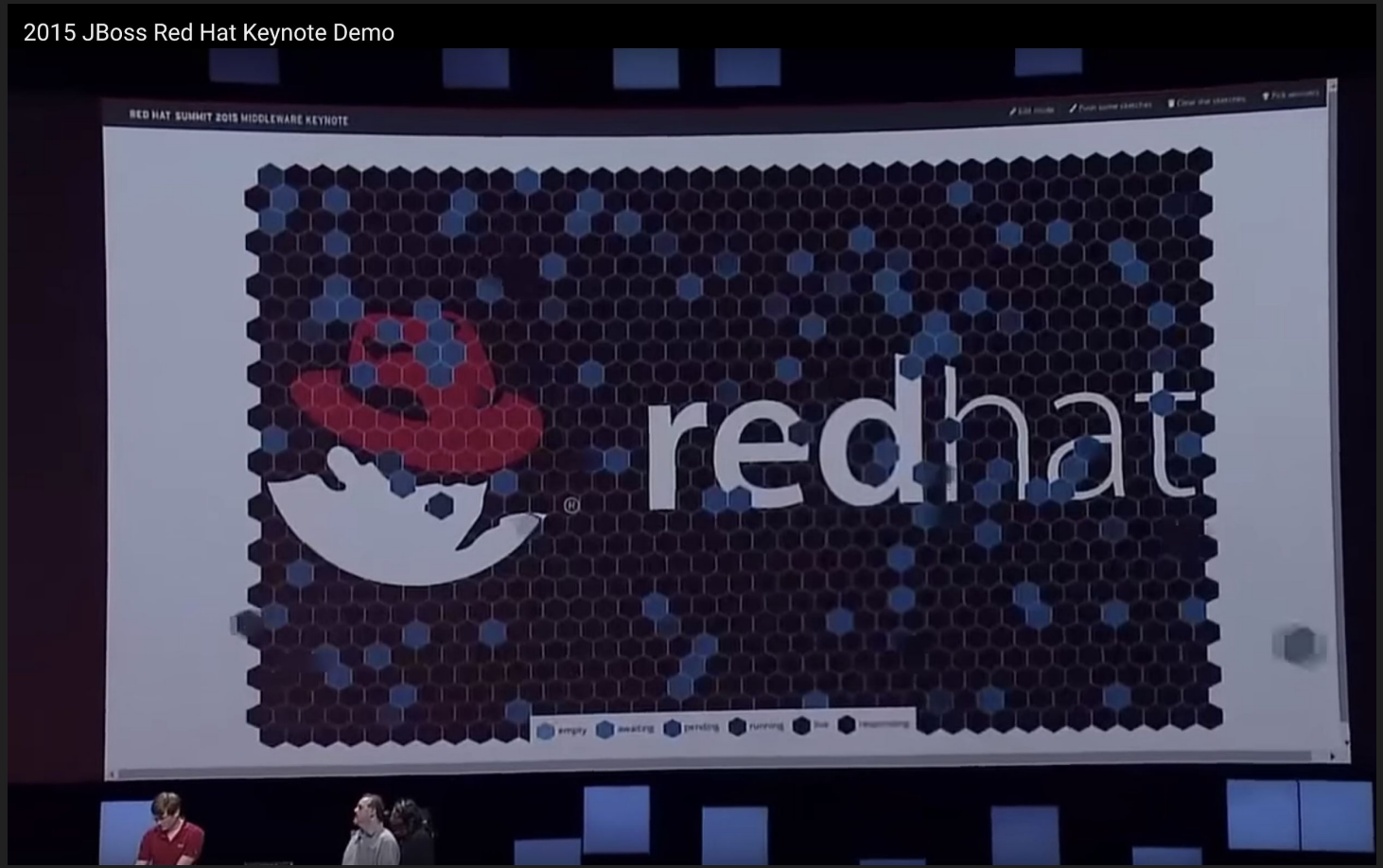
"Compile-time"
dependency
injection

# Quarkus: Supersonic, Subatomic Java

Kubernetes Native

# History of Microservices



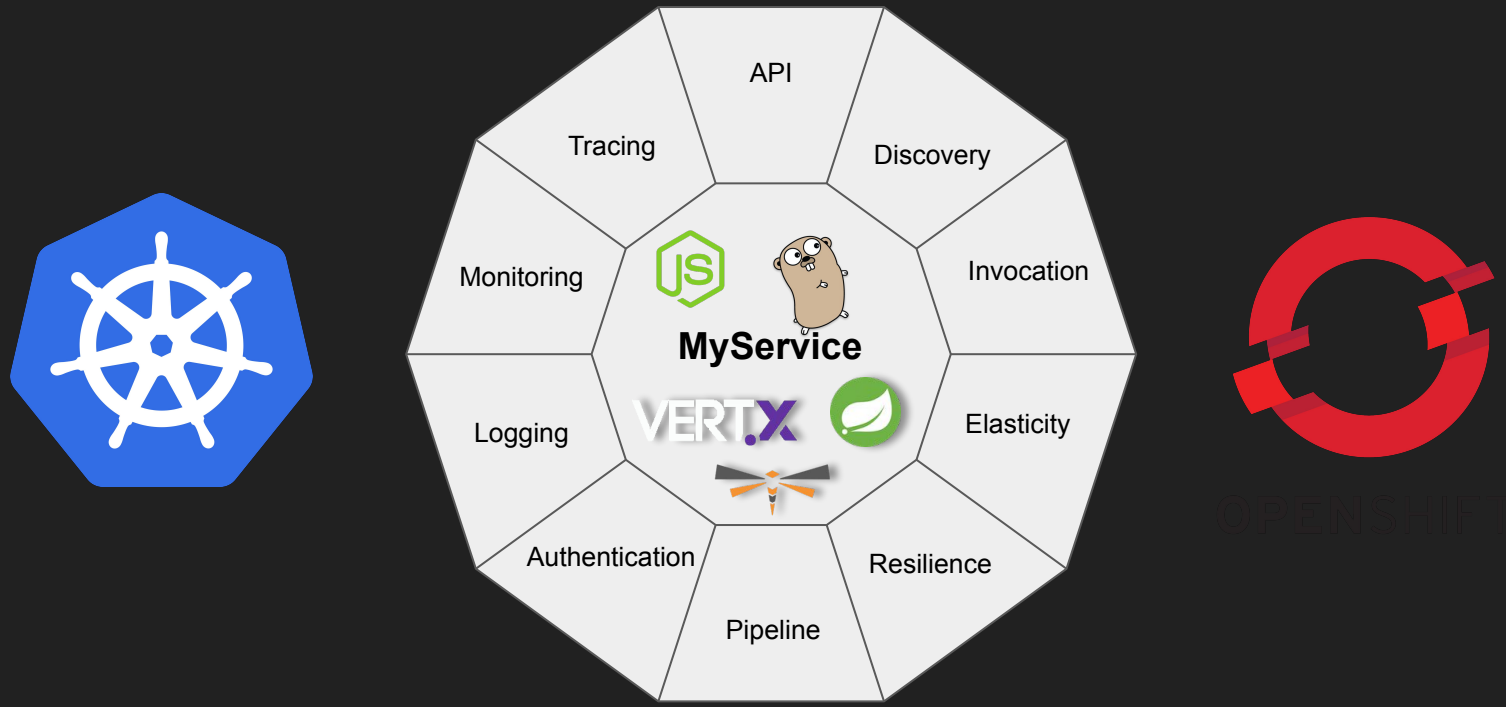



# 2015

Launch 1000+ Containers

Audience Claims a Container

https://www.youtube.com/watch?v=GCtpncA0Ea0&feature=youtu.be&t=1031




# Polyglot Microservices Platform

# 9 Steps

# Step 1: Installation

# Step 1: Installation

Lots of Options

1. Localhost Development
   a. minikube (kubectl) (docs)
   ~~b. minishift (oc) (docs)~~
2. Hosted Kubernetes Cluster
   a. GKE from Google Cloud Platform
   b. AKS from Microsoft Azure
   c. EKS from Amazon Web Services
3. Many more…
   https://kubernetes.io/docs/setup/pick-right-solution/#turnkey-cloud-solutions

# Tools

- bash: documented commands are based on bash
- docker and/or podman: build images
- git: pull resources from github
- JDK & Apache Maven: if you wish to compile Java
- curl, tar: downloading and unpacking installations
- Homebrew for Mac:  package installation tool
- kubectl: the primary tool you will be using
- minikube: on-laptop, single-node Kubernetes cluster

# Kubernetes Options

Localhost Development

a. minikube (kubectl) (docs)
b. Docker for Mac/Windows
c. CodeReady Containers
d. K3s
e. Micro8ks
f. KinD

Hosted Kubernetes Cluster

a. openshift.com/try from Red Hat
b. GKE from Google Cloud Platform
c. AKS from Microsoft Azure
d. EKS from Amazon Web Services
e. IKS from IBM Cloud
f. Digital Ocean

Roll-your-own with kubeadm

Many others

# Step 1 Demo

https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/installation.html

# Step 2: Building Images

# Running your App on Kubernetes

1. Find a base Image: Docker Hub, quay.io, gcr.io, access.redhat.com/containers
2. Craft your `Dockerfile`
3. Build your Image: `docker build -t mystuff/myimage:v1 .`
   a. If remote, do "docker push"
4. `kubectl apply -f myDeployment.yml`
5. `kubectl apply -f myService.yml`
6. `kubectl apply -f myIngress.yml`
7. Expose a URL via your Kubernetes distribution's load-balancer

# Step 2: Building Images

Options Include:

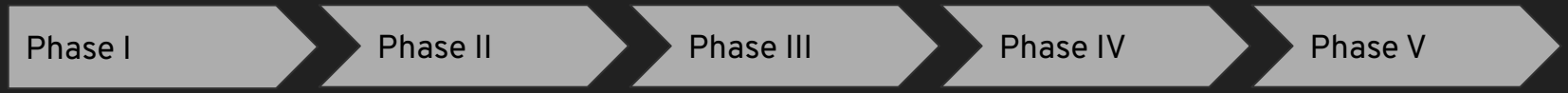
A. **docker build** then kubectl run or kubectl create -f deploy.yml
B. ~~Fabric8 maven plugin (fabric8.io)~~ (JShift.io - Eclipse JKube)
C. Jib - Maven/Gradle plugin
D. s2i - source to image
E. ~~Knative Build~~ - source to image, source to url (Tekton.dev)
F. No d-o-c-k-e-r
    a. Red Hat's podman, Google's kaniko, Uber's makisu
G. Buildpacks - similar to Heroku & Cloud Foundry

# Managing Yaml

Options Include:

A. Helm Charts - but for Tiller (Tiller going away in 3)
B. Kn (kn service create translator --image=<your_container_registry>/demo-app:v1 --autoscale-window 6s --revision-name=v1)
C. Kompose - converts docker-compose.yml to kubernetes yaml
D. jsonnet - https://jsonnet.org/articles/kubernetes.html
E. ~~Ksonnet~~ - templating for Kubernetes manifests (archived)
F. Kapitan - templating for Kubernetes manifests (helm, kapitan, kustomize)
G. Kustomize - templating for Kubernetes manifests (Helm vs Kustomize) - how built-in "kubectl apply -k"

# Beyond Managing Yaml - Operators




**Installation**
Provisioning
Configuration
Management

**Updates**
Patches and
Minors

**Full Lifecycle**
App and Storage
(backup, failure,
recovery)

**Deep Insights**
Metrics, alerts, log
processing,
workload analysis

**Auto Pilot**
Auto tuning, auto
scaling, abnormality
detection

Helm →

← Ansible →

← Go →

Operator SDK
https://github.com/operator-framework/operator-sdk

AWS Service Operator
provided by Amazon Web Services, Inc.

The AWS Service Operator allows you to manage AWS

Couchbase Operator
provided by Couchbase

The Couchbase Autonomous Operator allows users to easily deploy, manage, and maint

Crunchy PostgreSQL
provided by Crunchy Data Solutions, Inc.

PostgreSQL is a powerful, open source object-relational

Dynatrace OneAgent
provided by Dynatrace LLC

Install full-stack monitoring of Kubernetes clusters with the Dynatrace OneAgent.

etcd
provided by CNCF

Creates and maintain highly-available etcd clusters on Kubernetes

Federation
provided by Red Hat

Gain Hybrid Cloud capabilities between your clusters with Kubernetes Federation.

Jaeger Tracing
provided by Jaeger

Provides tracing, monitoring and troubleshooting microservices-based

MongoDB
provided by MongoDB, Inc

The MongoDB Enterprise Kubernetes Operator enables easy deploys of MongoDB

Percona Xtradb Cluster Operator
provided by Percona

Percona MySQL Operator manages the lifecycle of

PlanetScale Operator for Vitess
provided by PlanetScale

PlanetScale's operator for Vitess deploys and manage

Prometheus Operator
provided by Red Hat

The Prometheus Operator for Kubernetes provides easy monitoring definitions for


Redis Enterprise
provided by Redis Labs, Inc

An operator to run Redis Enterprise Clusters

Strimzi Kafka
provided by Red Hat

Run an Apache Kafka cluster, including Kafka Connect, ZooKeeper and more.

[Operatorhub.io](Operatorhub.io)

RED HAT DEVELOPER

# Dockerfile for Java projects

```
FROM fabric8/java-jboss-openjdk8-jdk:1.4.0
ENV JAVA_APP_DIR=/deployments
EXPOSE 8080 8778 9779
COPY target/my.jar /deployments/
```

https://github.com/fabric8io-images/java/tree/master/images/jboss/openjdk8/jdk

# docker build, kubectl run

```
minikube(docker-env) or minishift(docker-env)
docker build -t burr/myimage:v1 .
docker run -it -p 8080:8080 burr/myimage:v1
curl $(minishift ip):8080
# now run it on Kubernetes
kubectl run myapp --image burr/myimage:v1 --port 8080
kubectl expose deployment --port=8080 myapp --type=LoadBalancer
oc expose service myapp
curl myapp-stuff.$(minishift ip).nip.io
# scale up
kubectl scale --replicas=2 deploy/myapp
# create an updated image
docker build -t burr/myimage:v2 .
# rollout update
kubectl set image deployment/myapp myapp=burr/myimage:v2
```

# ~~Fabric8 Maven Plugin~~ http://jshift.io/

https://maven.fabric8.io/#fabric8:setup (no **Dockerfile, Deployment.yml**)

```
oc new-project stuff (or kubectl create namespace)

mvn clean compile package

mvn io.fabric8:fabric8-maven-plugin:3.5.40:setup

mvn fabric8:deploy
```

# Do NOT Java + Docker == FAIL

Slides: bit.ly/javadockerfail

Recording from JBCNConf 2017

```
docker run -m 100MB openjdk:8u121 java
-XshowSettings:vm -version

docker run -m 100MB openjdk:8u131 java
-XX:+UnlockExperimentalVMOptions
-XX:+UseCGroupMemoryLimitForHeap -XshowSettings:vm
-version
```

# Step 2 Demo

https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/building-images.html

# Step 3: logs

# logs

```
System.out.println("Java, where am I?");
Or console.log("Node logs");

kubectl get pods
kubectl logs microspringboot1-2-nz8f8
kubectl logs microspringboot1-2-nz8f8 -p # last failed pod

OR stern (brew install stern)
https://github.com/wercker/stern
stern myboot

OR kail (https://github.com/boz/kail)
kail -d myboot
```

# Step 3 Demo

https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/logs.html

# Step 4: oc or kubectl exec

# Step 4: oc or kubectl exec

"ssh" into your containers and explore

```
kubectl get pods --namespace=microworld
kubectl exec -it --namespace=microworld $POD cat /sys/fs/cgroup/memory/memory.limit_in_bytes
Or
kubectl exec -it --namespace=microworld microspringboot1-2-nz8f8 /bin/bash
ps -ef | grep java
Note: the following apply if using the fabric8 generated image, otherwise consult your Dockerfile
java -version
javac -version
# now find that fat jar
find / -name *.jar
cd /deployments (based on use of the fabric8 maven plugin)
ls
exit
```

# Step 4 Demo

https://github.com/burrsutter/9stepsawesome/blob/master/4_kubectl_exec.adoc
Video on Resource Limits/Constraints
https://www.youtube.com/watch?v=xjpHggHKm78

# Step 5: env and configmaps

# Step 5: env vars & configmaps

*An app's config is everything that is likely to vary between deploys (staging, production, developer environments, etc).* [*12 Factor Apps*]

```
kubectl set env deployment/myboot DBCONN="jdbc:sqlserver://45.91.12.123:1443;user=MyUserName;password=*****;"

kubectl create cm my-config --from-env-file=config/some.properties
```

# ConfigMaps

```
kubectl create cm my-config --from-env-file=config/some.properties
```

some.properties

```
GREETING=jambo

LOVE=Amour
```

Partial deployment.yml

```
    spec:
      containers:
      - name: myboot
        image: 9stepsawesome/myboot:v1
        ports:
          - containerPort: 8080
        envFrom:
        - configMapRef:
            name: my-config
```

Partial .java

```
@Autowired

private Environment environment;

String greeting =
environment.getProperty("GREETING","Default");

String love =
environment.getProperty("LOVE","Default");
```

# Step 5 Demo

https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/configmap.html

# Step 6: service discovery & load-balancing

# Step 6: Service Discovery

1. Services are internal to the cluster and can be mapped to pods via a label selector
2. Just refer to a Service by its name, it is just DNS

```
String url = "http://producer:8080/";

ResponseEntity<String> response =
restTemplate.getForEntity(url, String.class);
```

# Step 6 Demo

https://github.com/burrsutter/9stepsawesome/blob/master/6_discovery.adoc

# Step 7: Live and Ready

# Step 7: Live and Ready

```
kubectl create -f Deployment.yml
```

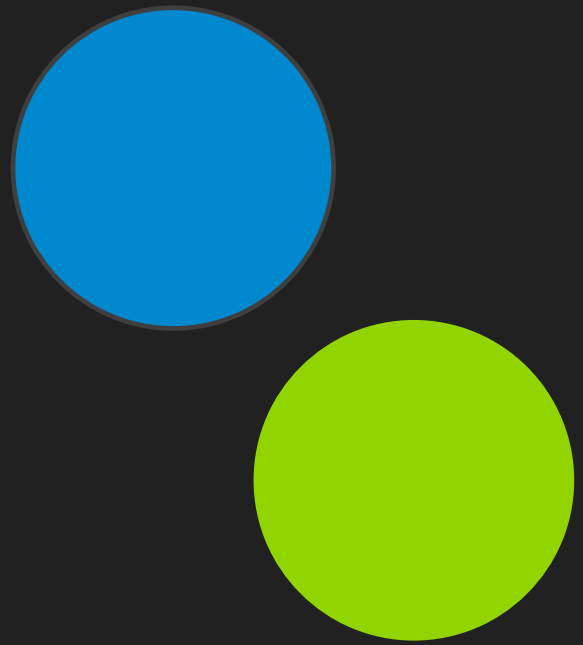
```yaml
resources:
  requests:
    memory: "300Mi"
    cpu: "250m" # 1/4 core
  limits:
    memory: "400Mi"
    cpu: "1000m" # 1 core
livenessProbe:
  httpGet:
      port: 8080
      path: /
  initialDelaySeconds: 10
  periodSeconds: 5
  timeoutSeconds: 2
readinessProbe:
  httpGet:
    path: /health
    port: 8080
  initialDelaySeconds: 10
  periodSeconds: 3
```

# Step 7 Demo

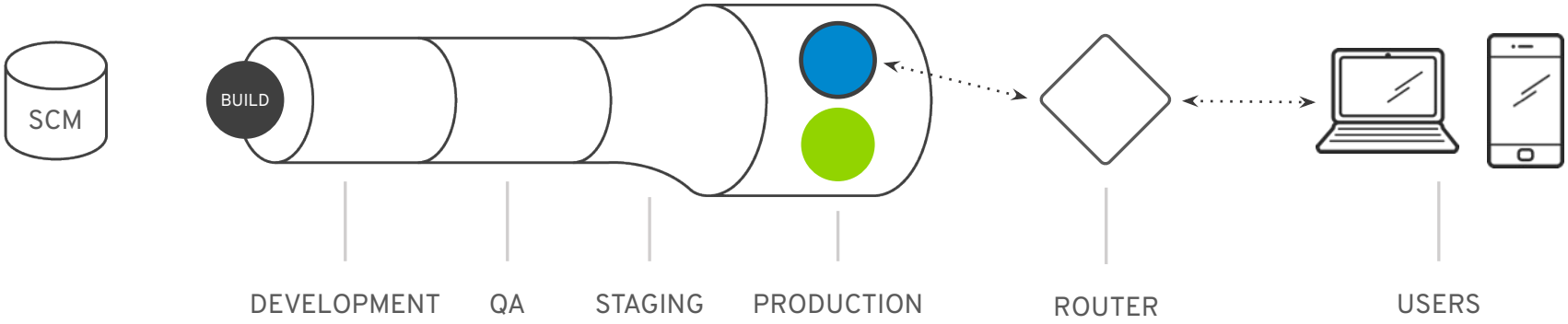
https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/live-ready.html

# Step 8: Rolling Updates, Blue/Green Canary

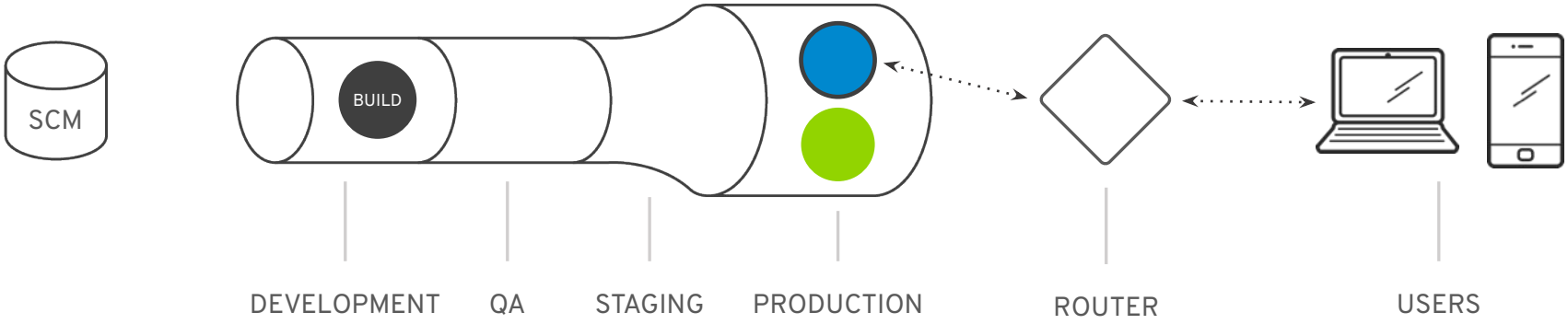
# Blue/Green Deployment



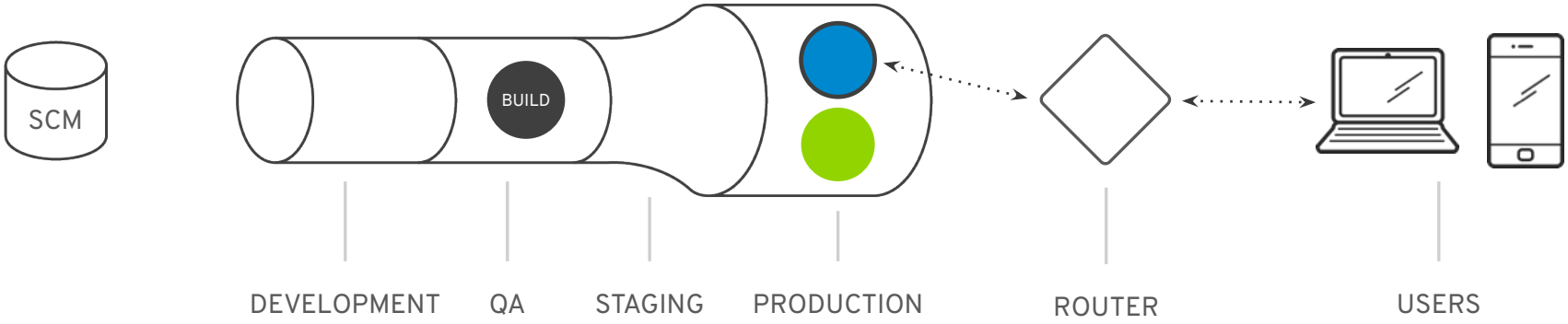
# Blue/Green Deployment



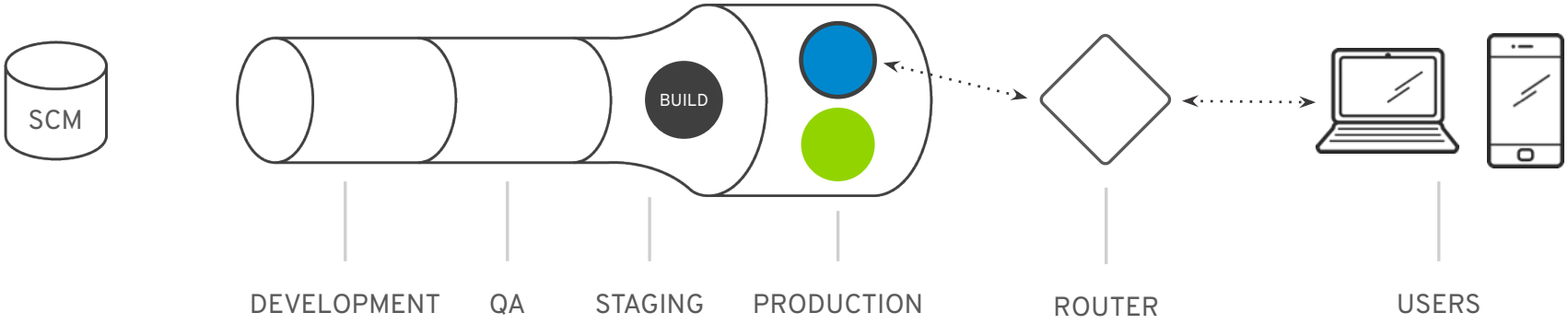
# Blue/Green Deployment



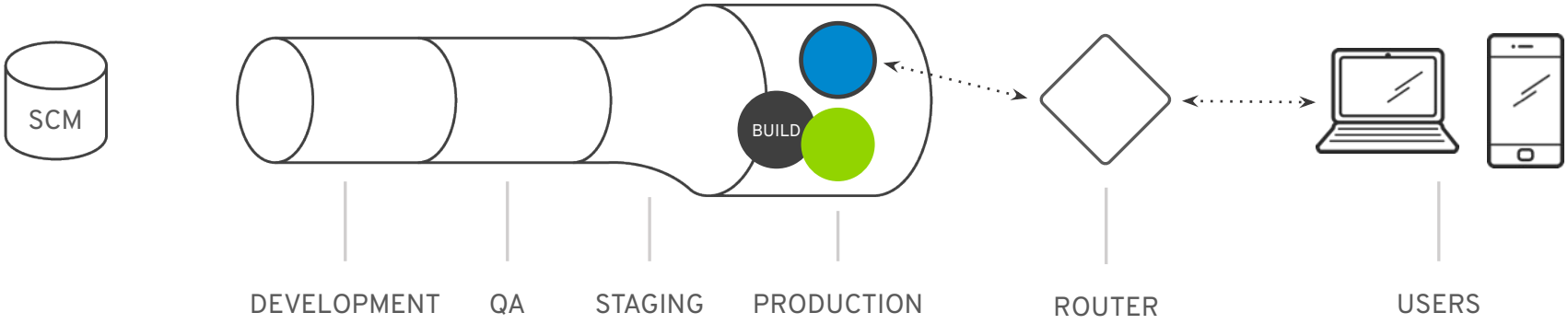
# Blue/Green Deployment



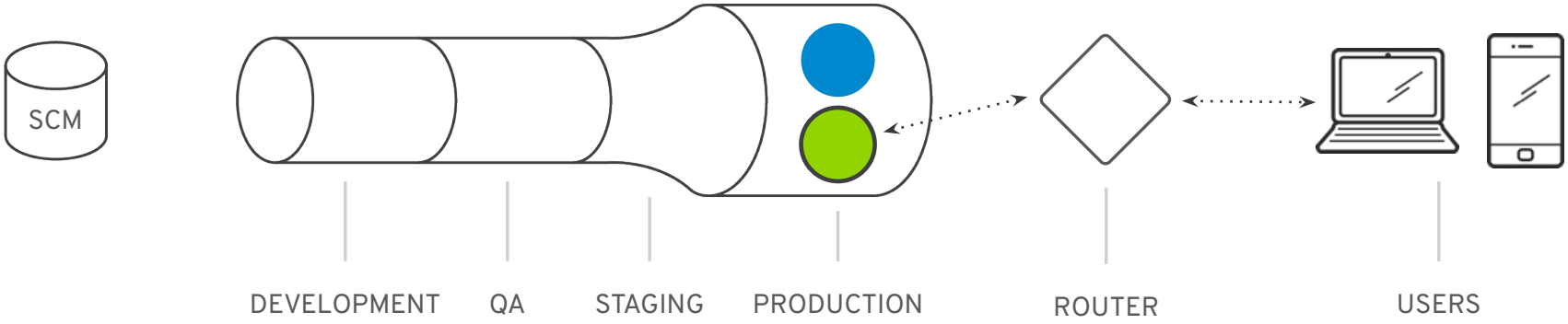
# Blue/Green Deployment



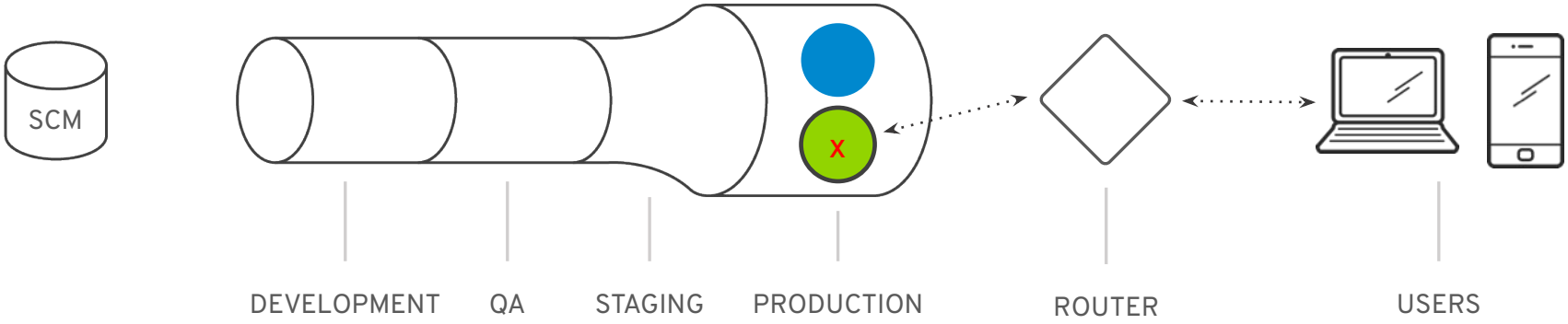
# Blue/Green Deployment



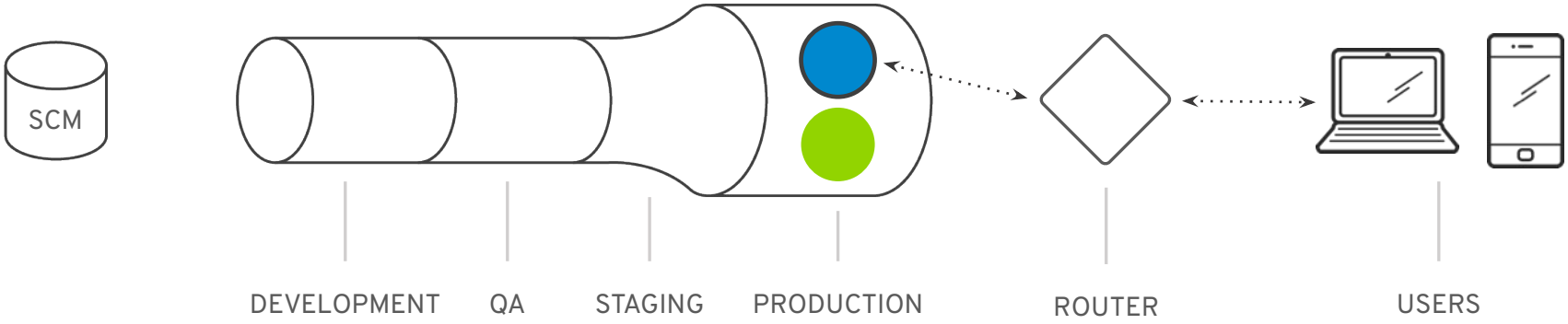
# Blue/Green Deployment



# Blue/Green Deployment



# Blue/Green Deployment



# Step 8 Demo

https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/blue-green.html

# New Step 9: Databases

# Step 9: Databases

1. Persistent Volume
2. Persistent Volume Claim
3. Deployment
4. Service

https://redhat-scholars.github.io/kubernetes-tutorial/kubernetes-tutorial/volumes-persistentvolumes.html

https://github.com/burrsutter/9stepsawesome/blob/master/9_databases.adoc

# Bonus: Istio

# Istio - Sail

(Kubernetes - Helmsman or ship's pilot)

https://dn.dev/istio-tutorial

# Next Generation - Service Mesh

Code Independent

- Intelligent Routing and Load-Balancing
  - Canary Releases
  - Dark Launches
- Distributed Tracing
- Circuit Breakers
- Fine grained Access Control
- Telemetry, metrics and Logs
- Fleet wide policy enforcement

# What is Knative?


"Kubernetes-based platform to build, deploy, and manage modern serverless workloads."

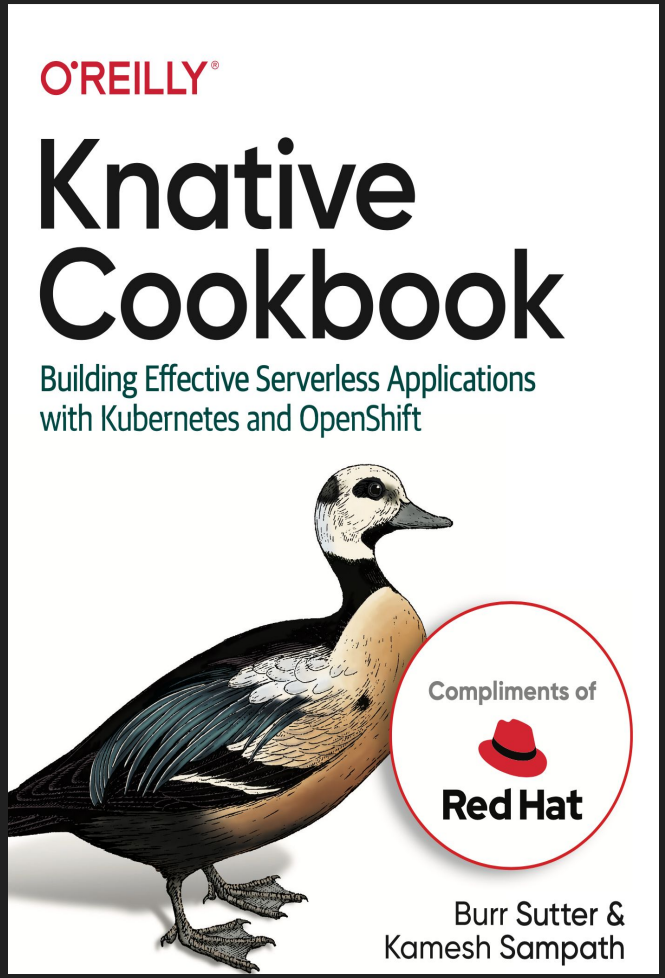
"Essential base primitives for all"

"Knative provides a set of middleware components that are essential to build modern, source-centric, and container-based applications that can run anywhere: on premises, in the cloud, or even in a third-party data center"


http://strimzi.io/

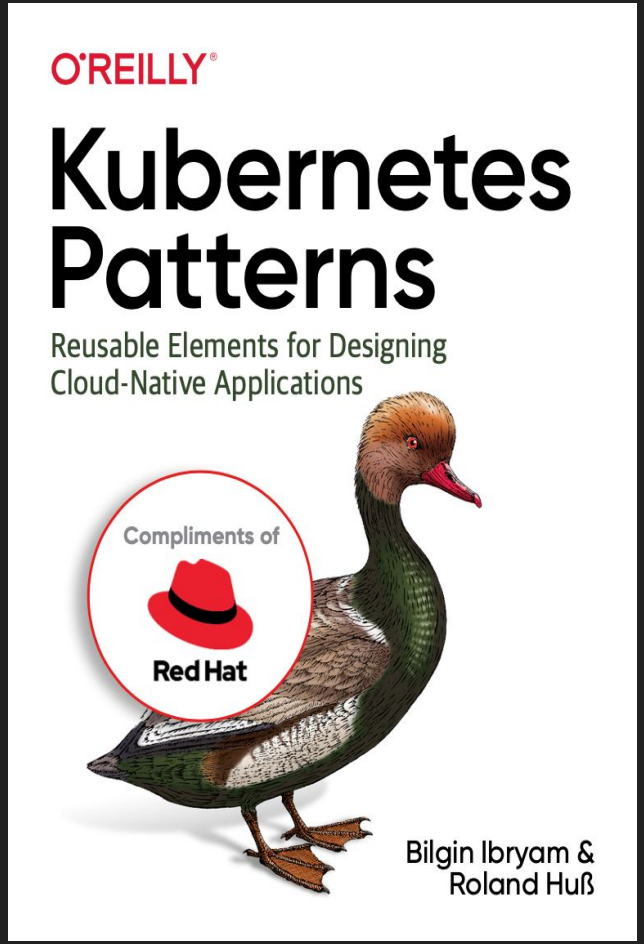
# Apache Kafka on Kubernetes & OpenShift

# Free eBooks/ Resources



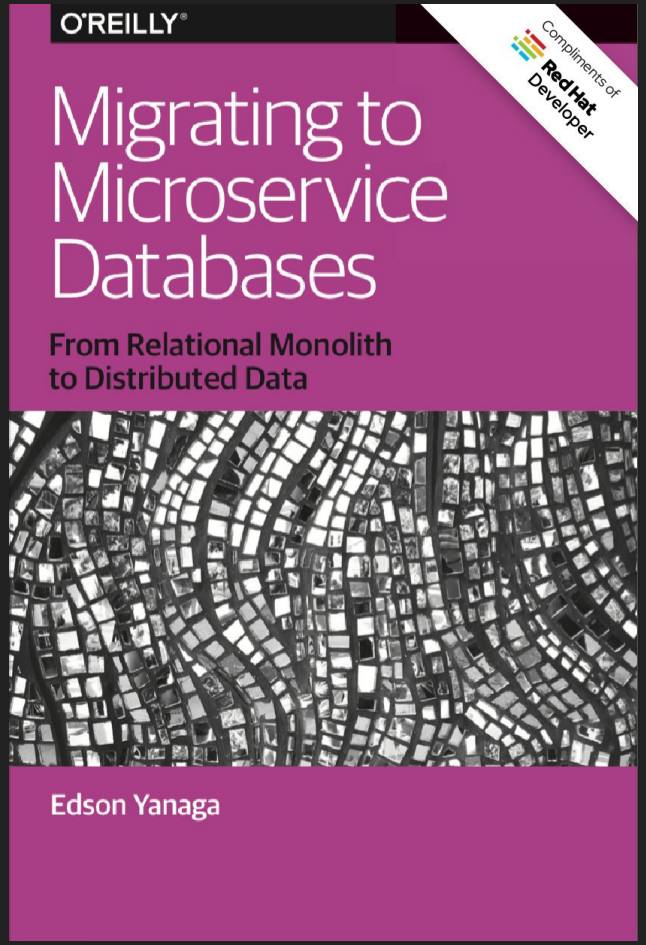

Download

dn.dev/knative-cookbook



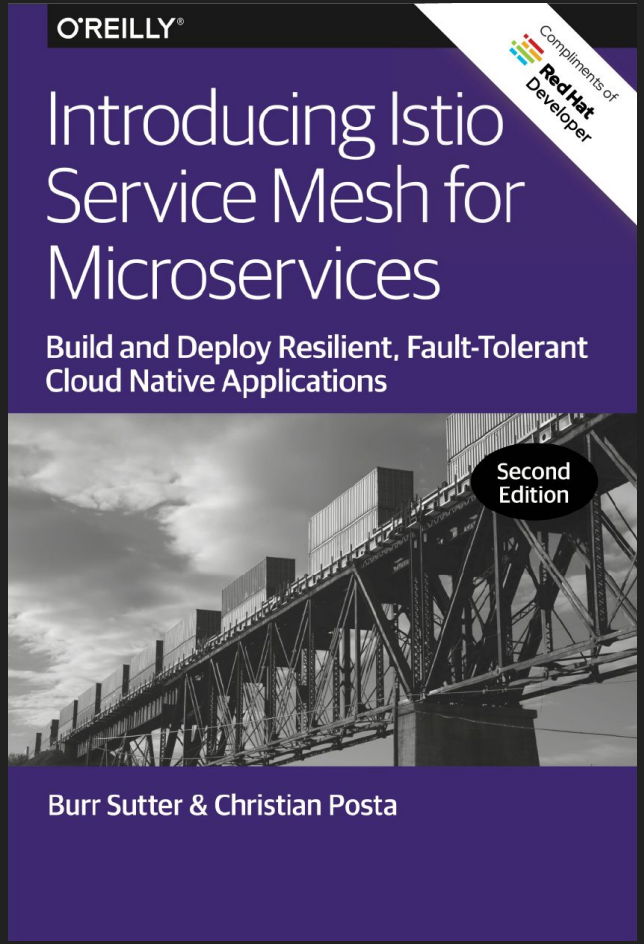

Download

https://dn.dev/k8s-patterns




Download

bit.ly/mono2microdb




Download

https://dn.dev/istiobook



RED HAT DEVELOPER

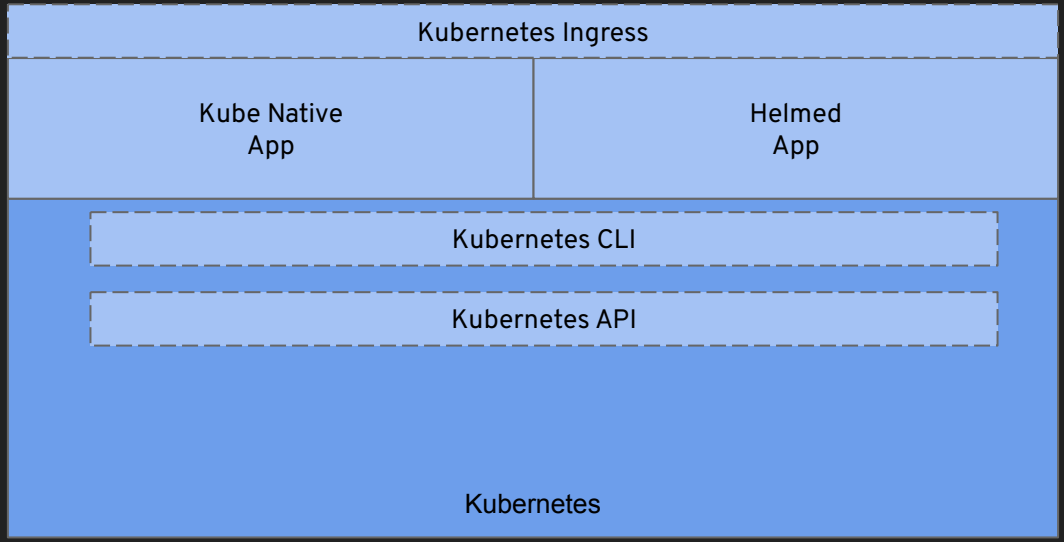
# Backup Content

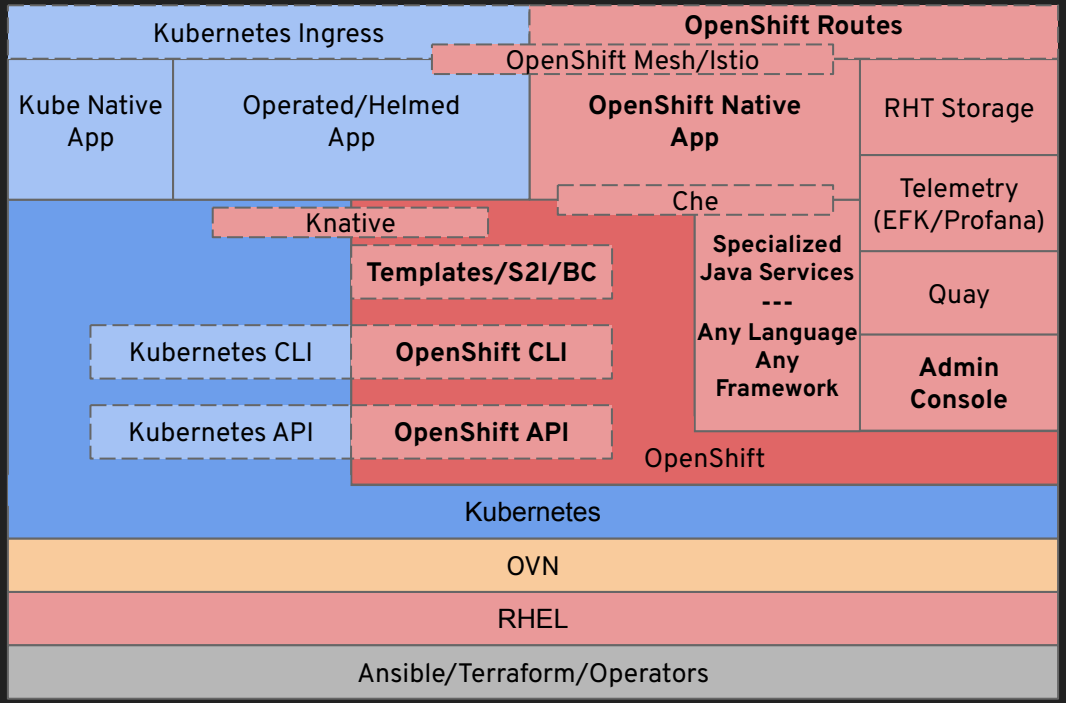
# The End

(Istio is Next)

bit.ly/istio-intro

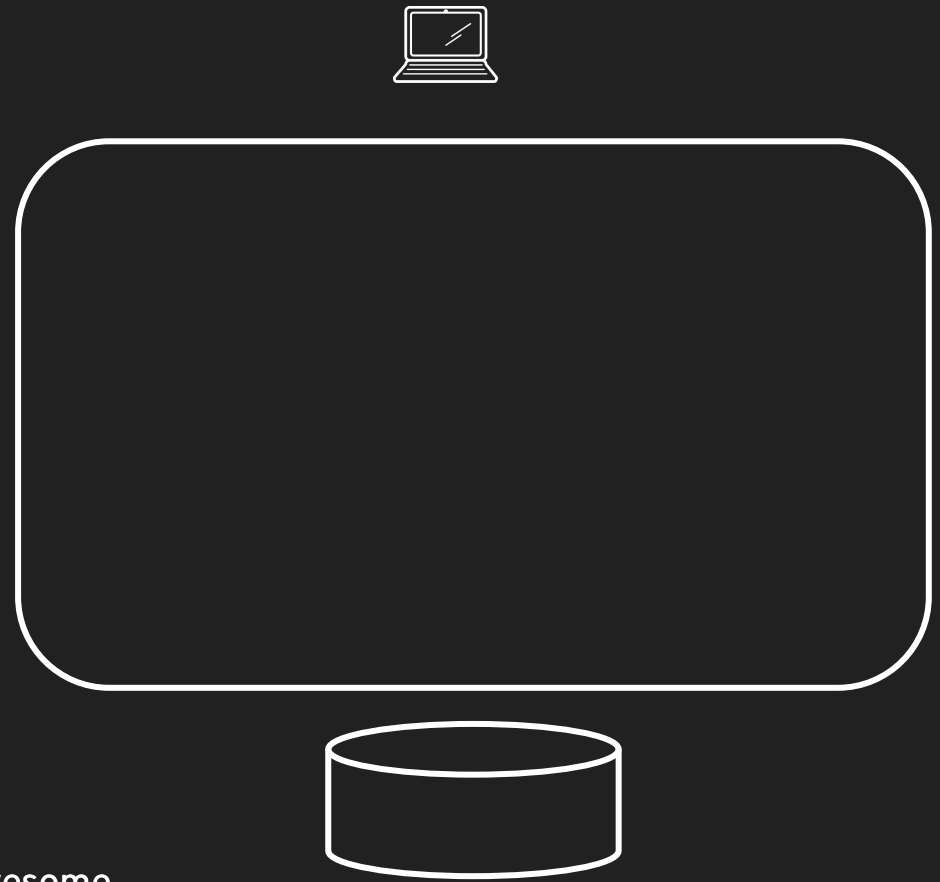
| | Kubernetes | OpenShift |
|---|---|---|
| Multi-container, multi-host scheduling | ✅ | ✅ |
| Self-service provisioning | ✅ | ✅ |
| Service discovery | ✅ | ✅ |
| Persistent Storage | ✅ | ✅ |
| Multi-tenancy | | ✅ |
| Networking (SDN) | | ✅ |
| Image Registry | | ✅ |
| Image Build Tools | | ✅ |
| Metrics | | ✅ |
| Log Aggregation | | ✅ |
| Ingress | | ✅ |
| UX: Console, Service Catalog, "oc" | | ✅ |
| Secured by Default | | ✅ |



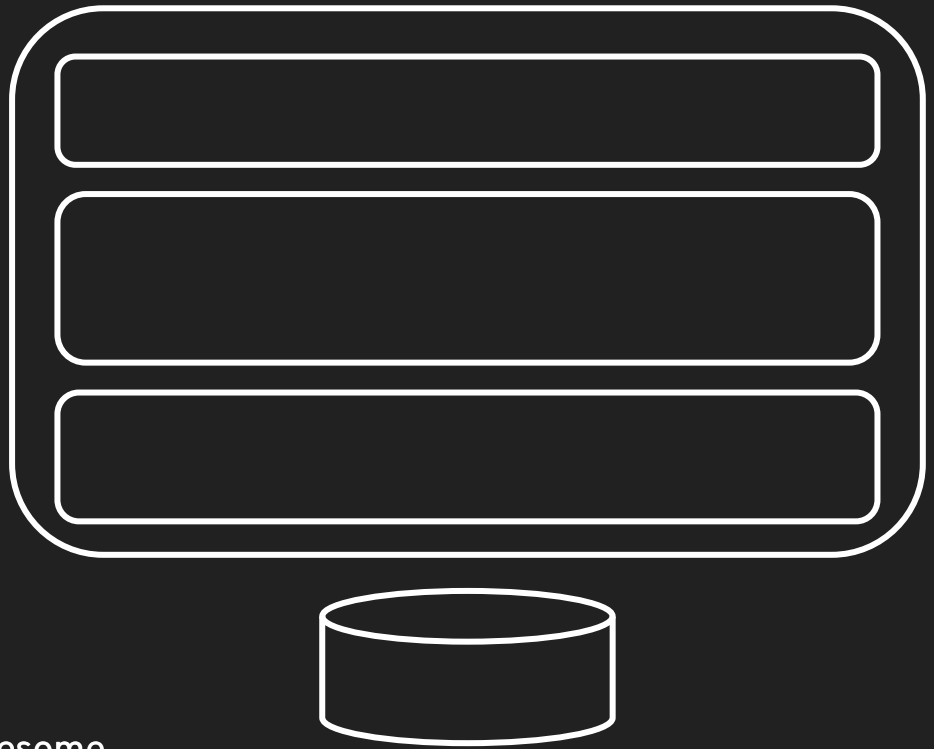


# Strangler Strategy

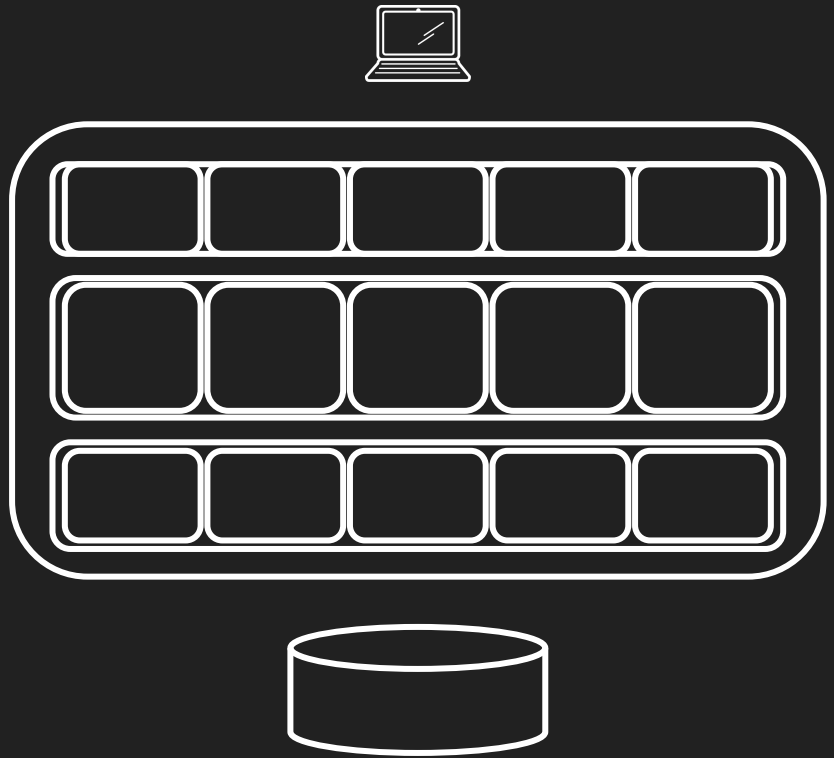
# Application on Whiteboard



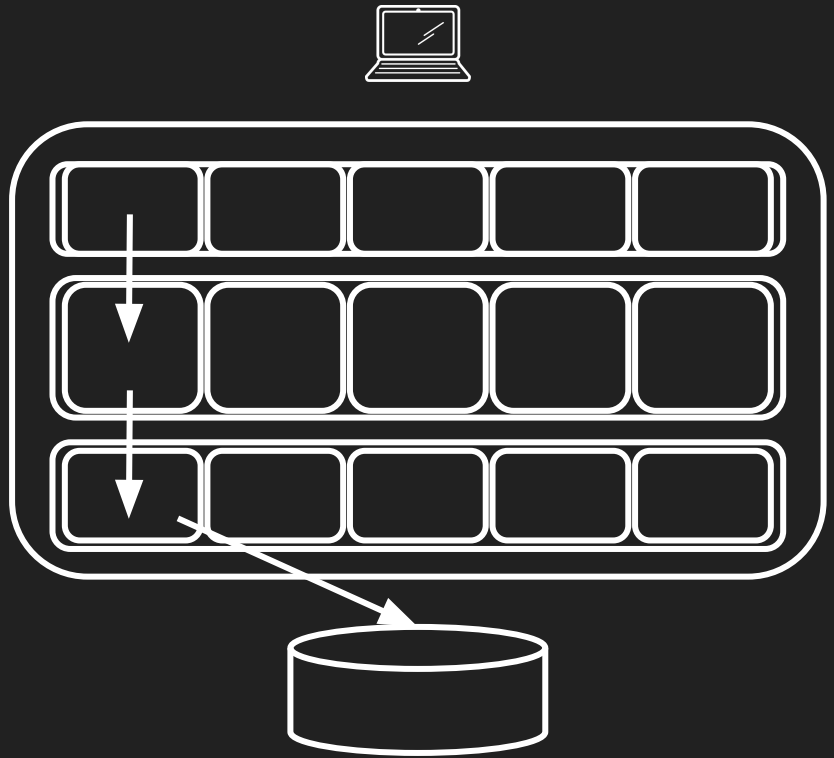
# Whiteboard had 3 Tiers



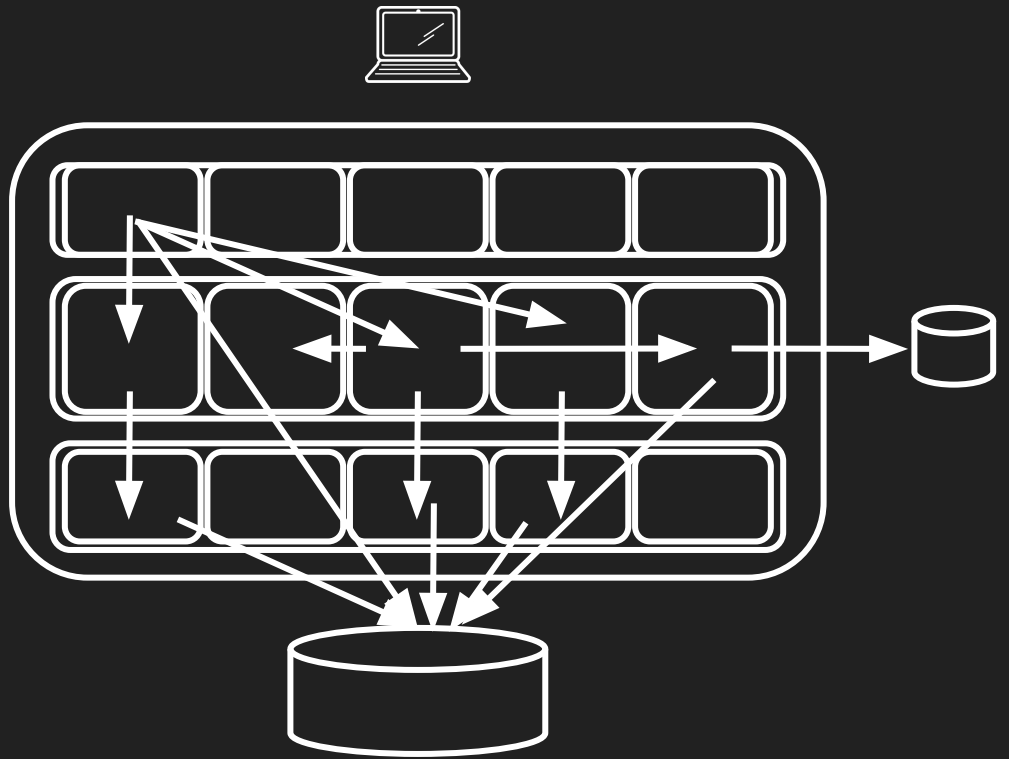
# UI, Logic, Data



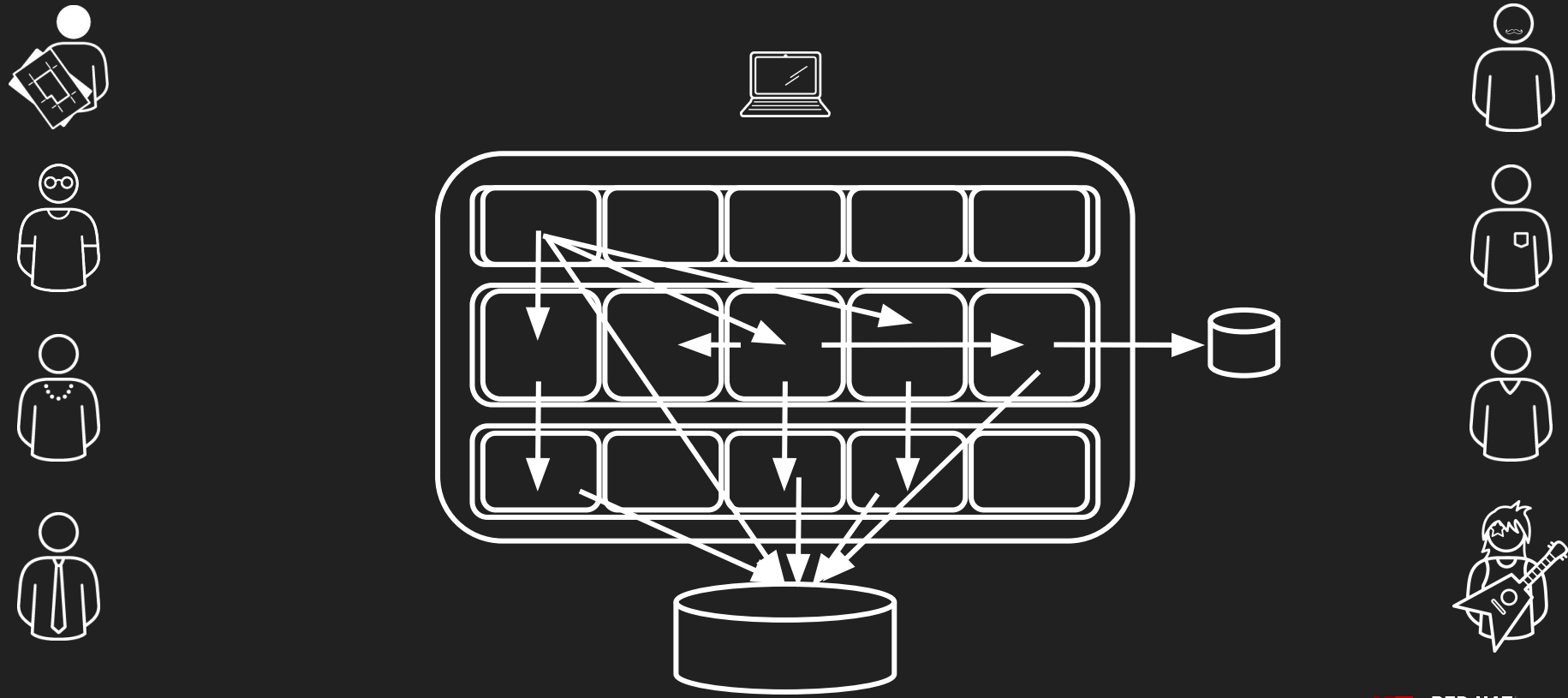
# Clean Architecture



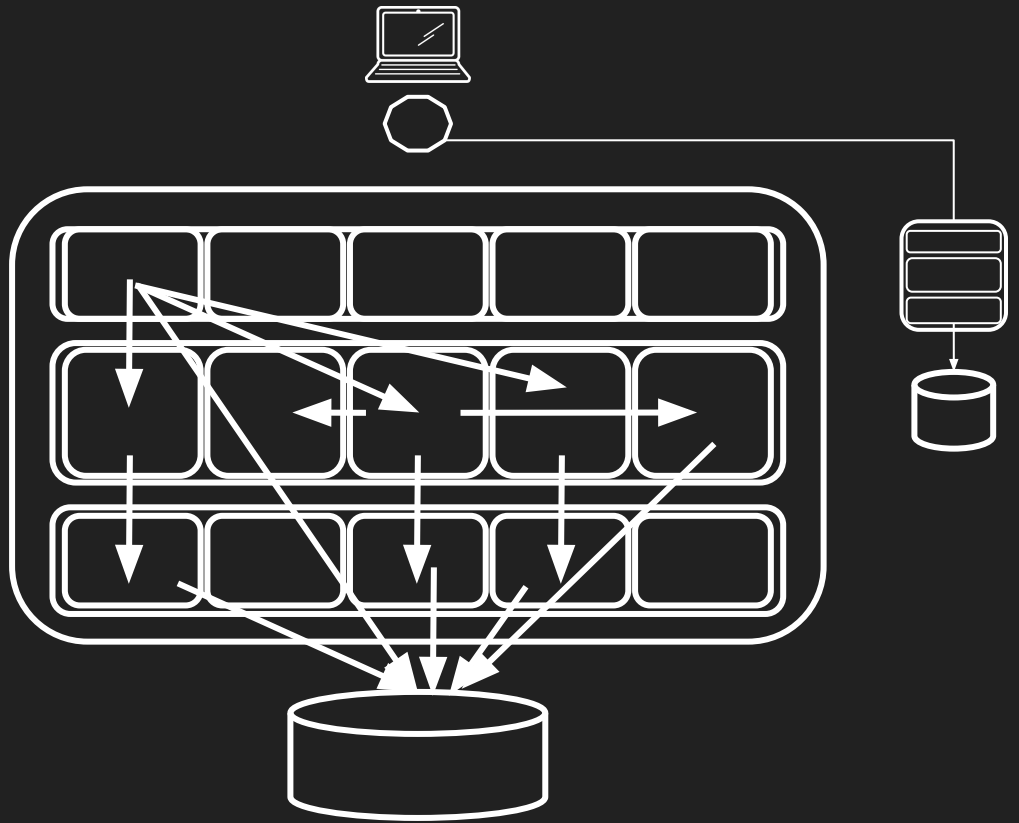
# Real Life: Non-Majestic Monolith



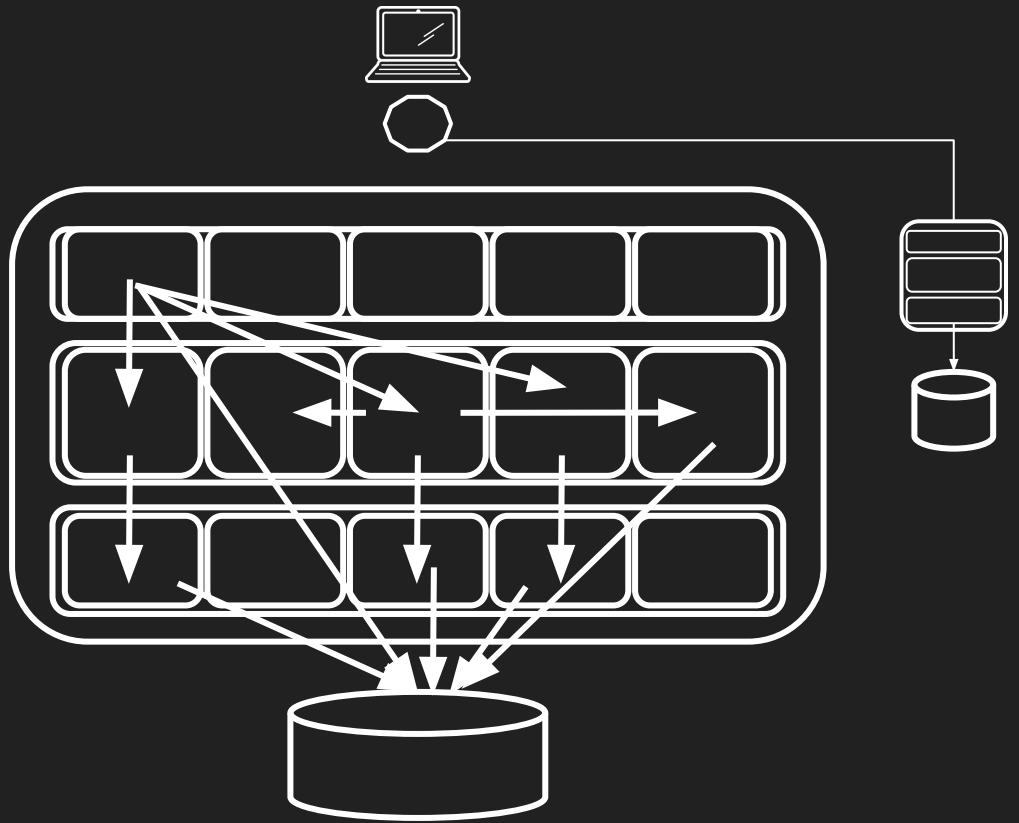
# Large Team



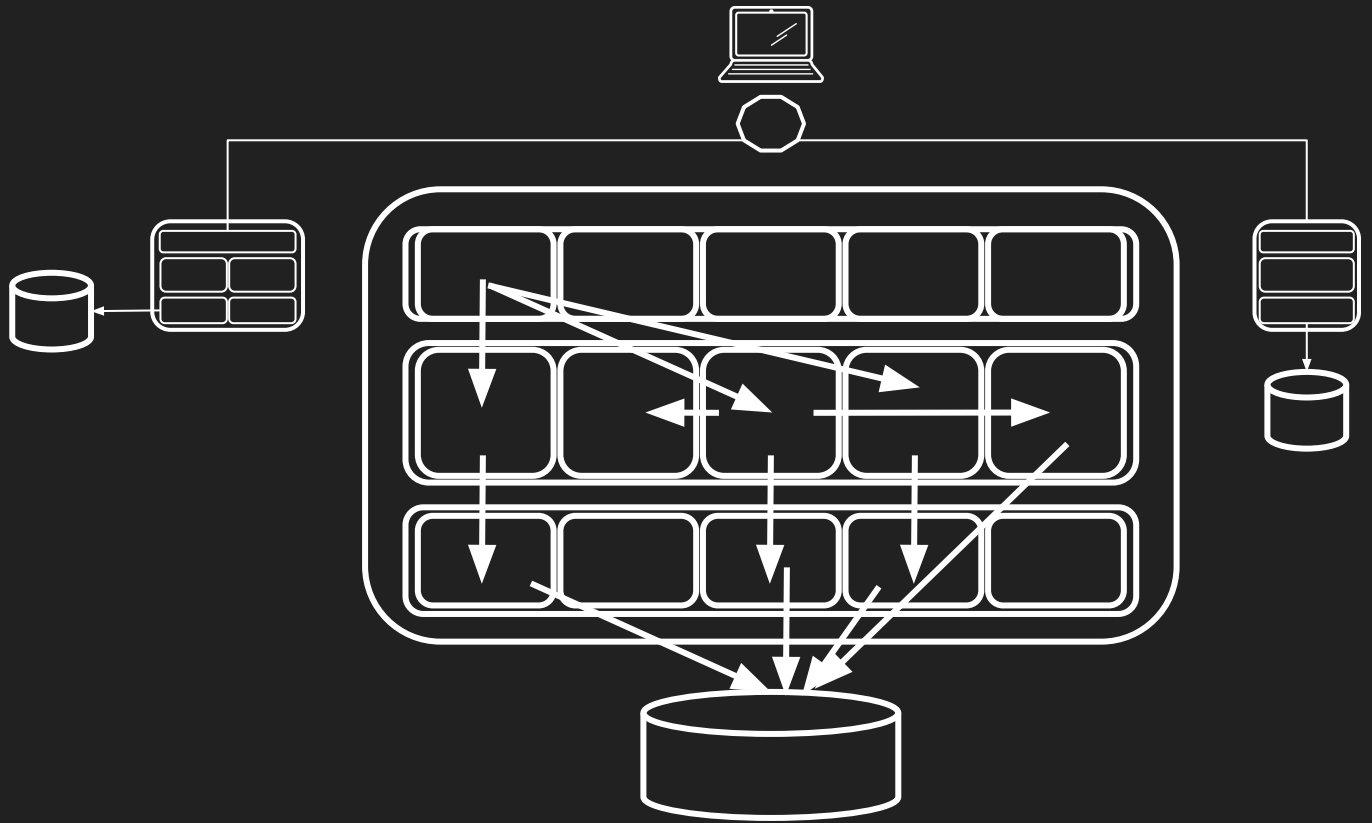
# Strangle



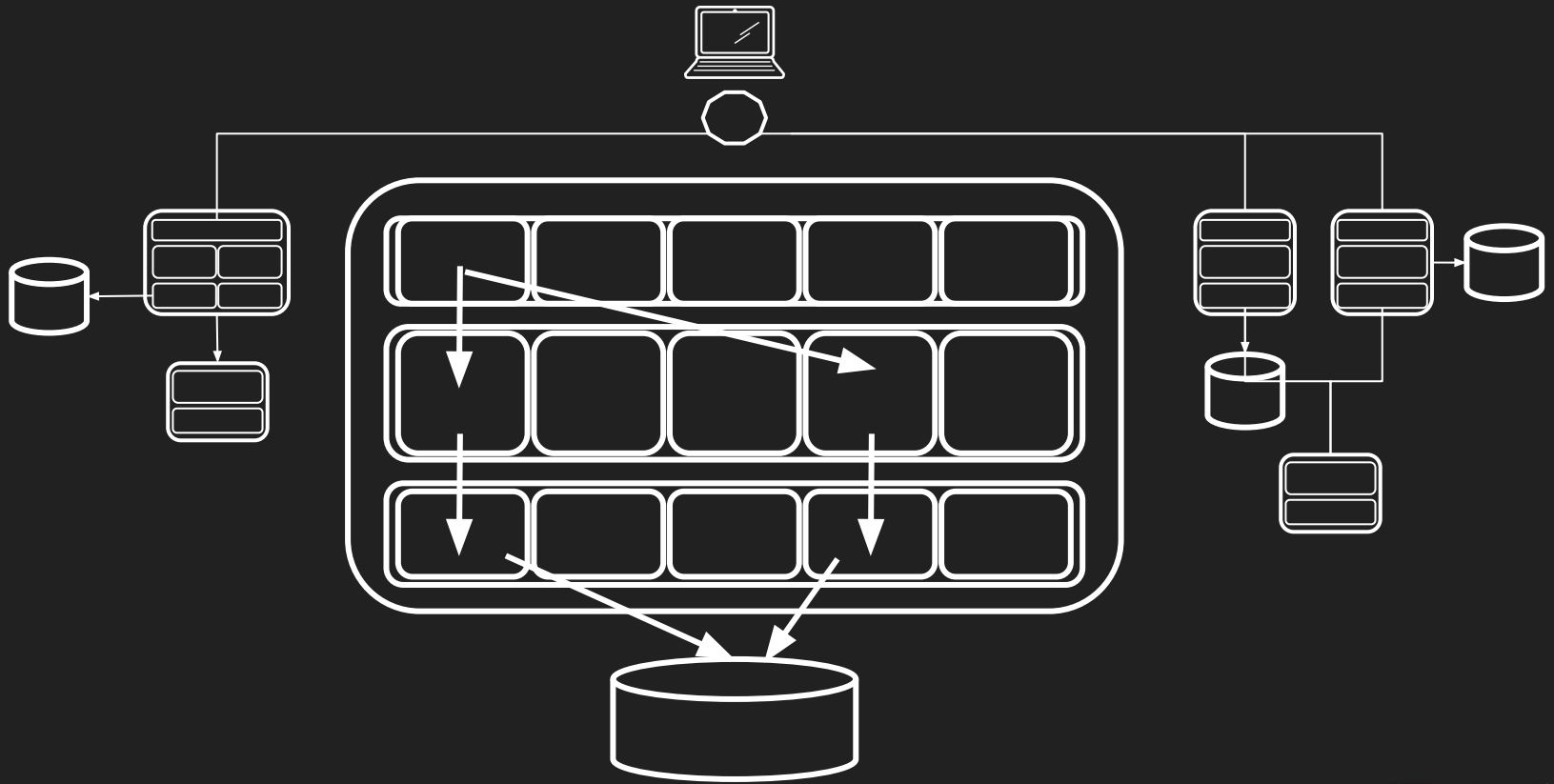
# ~~Strangle~~ Hug



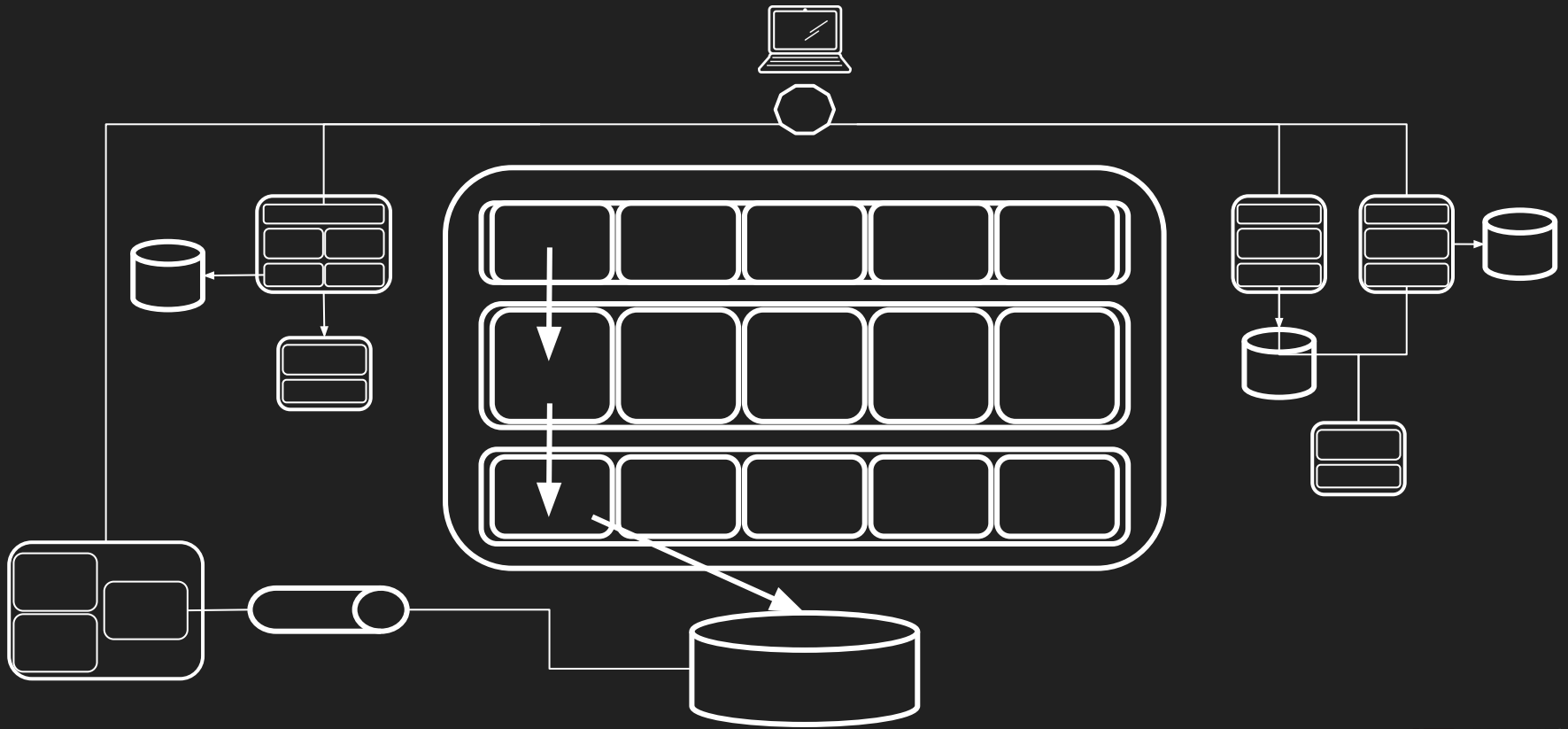
# Friend Hug



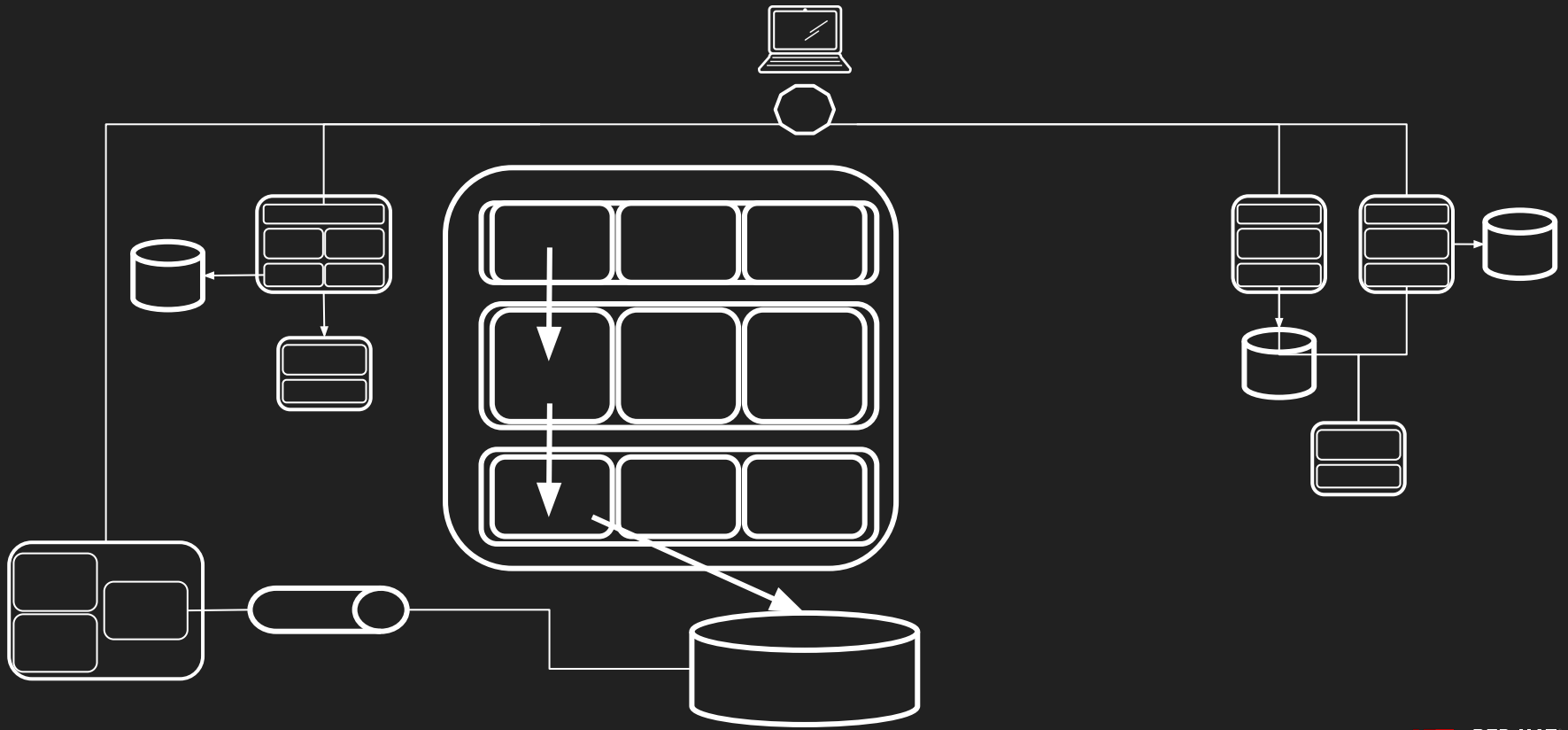
# Family Hug



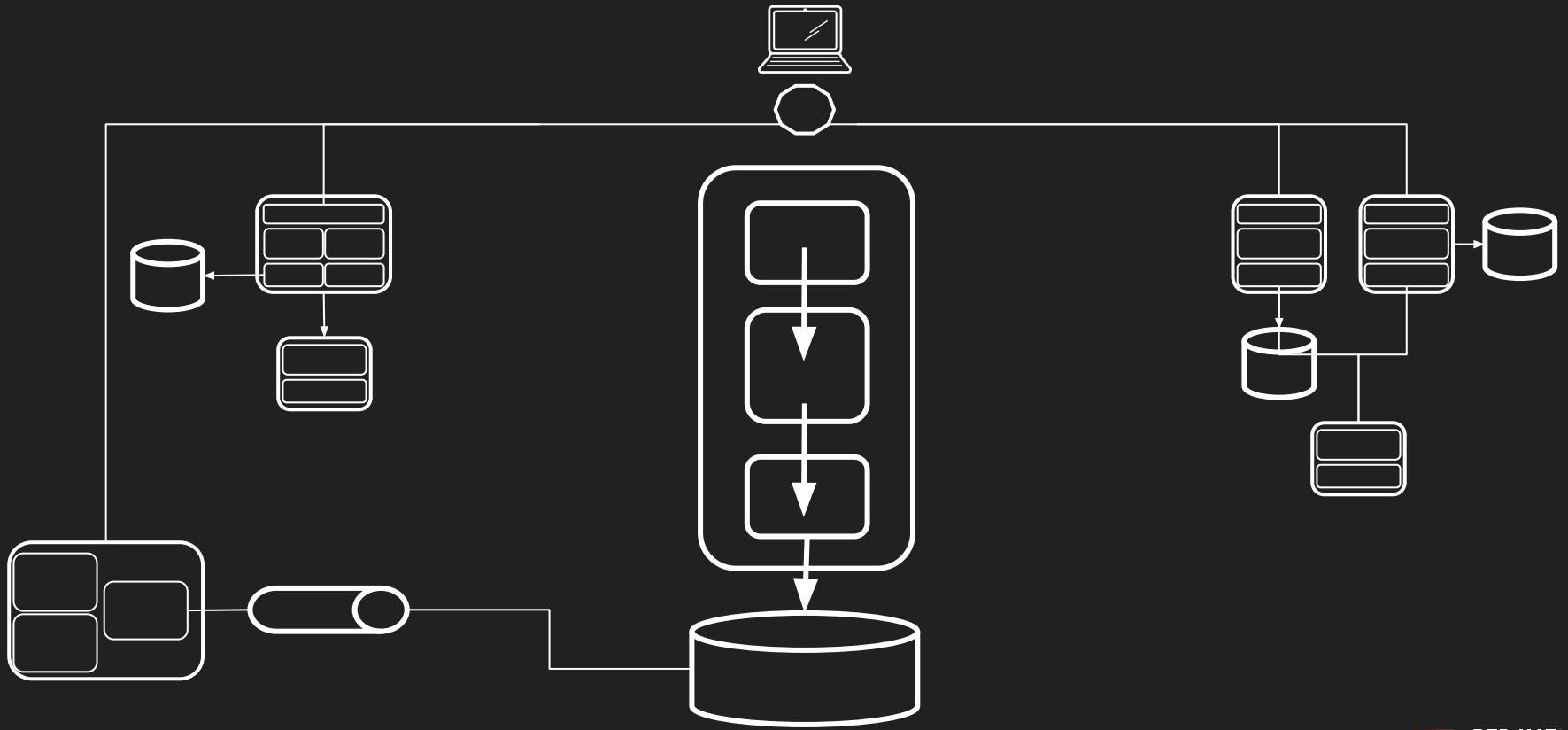
# Bear Hug



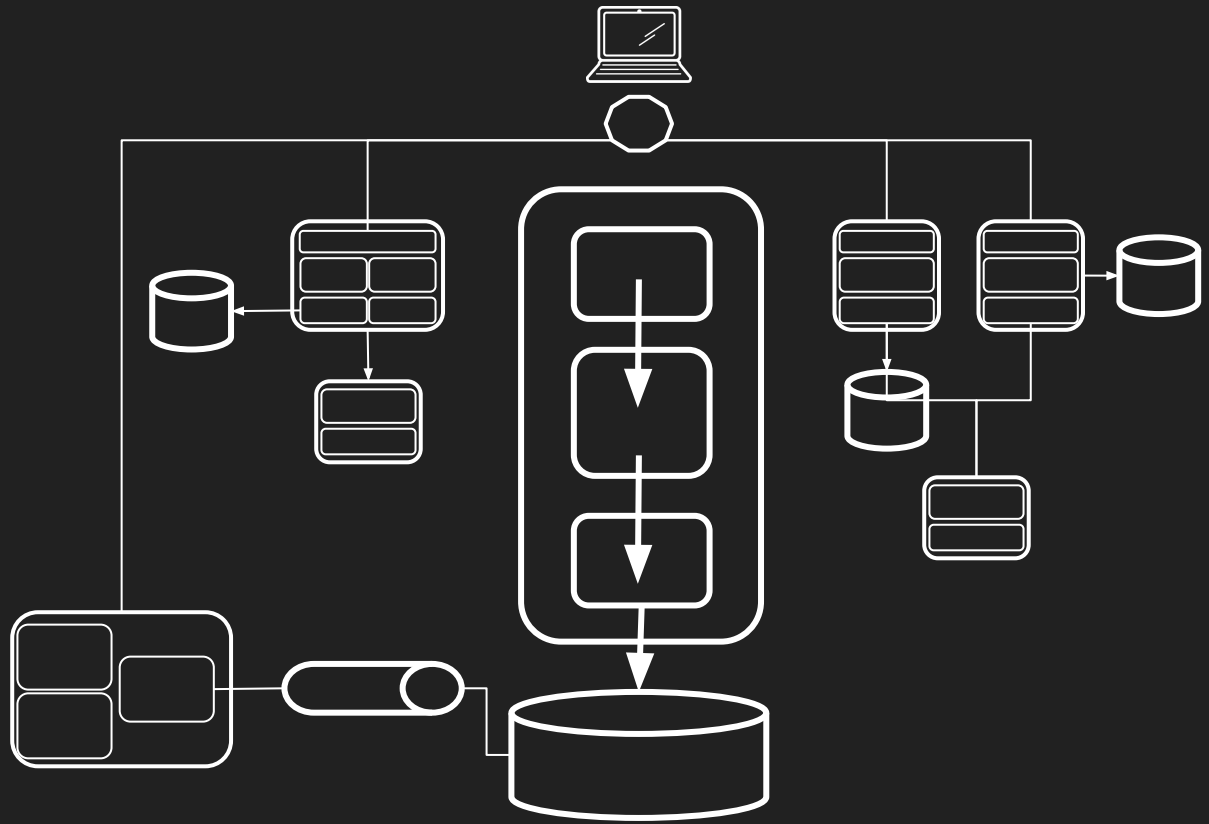
# "Refactor"



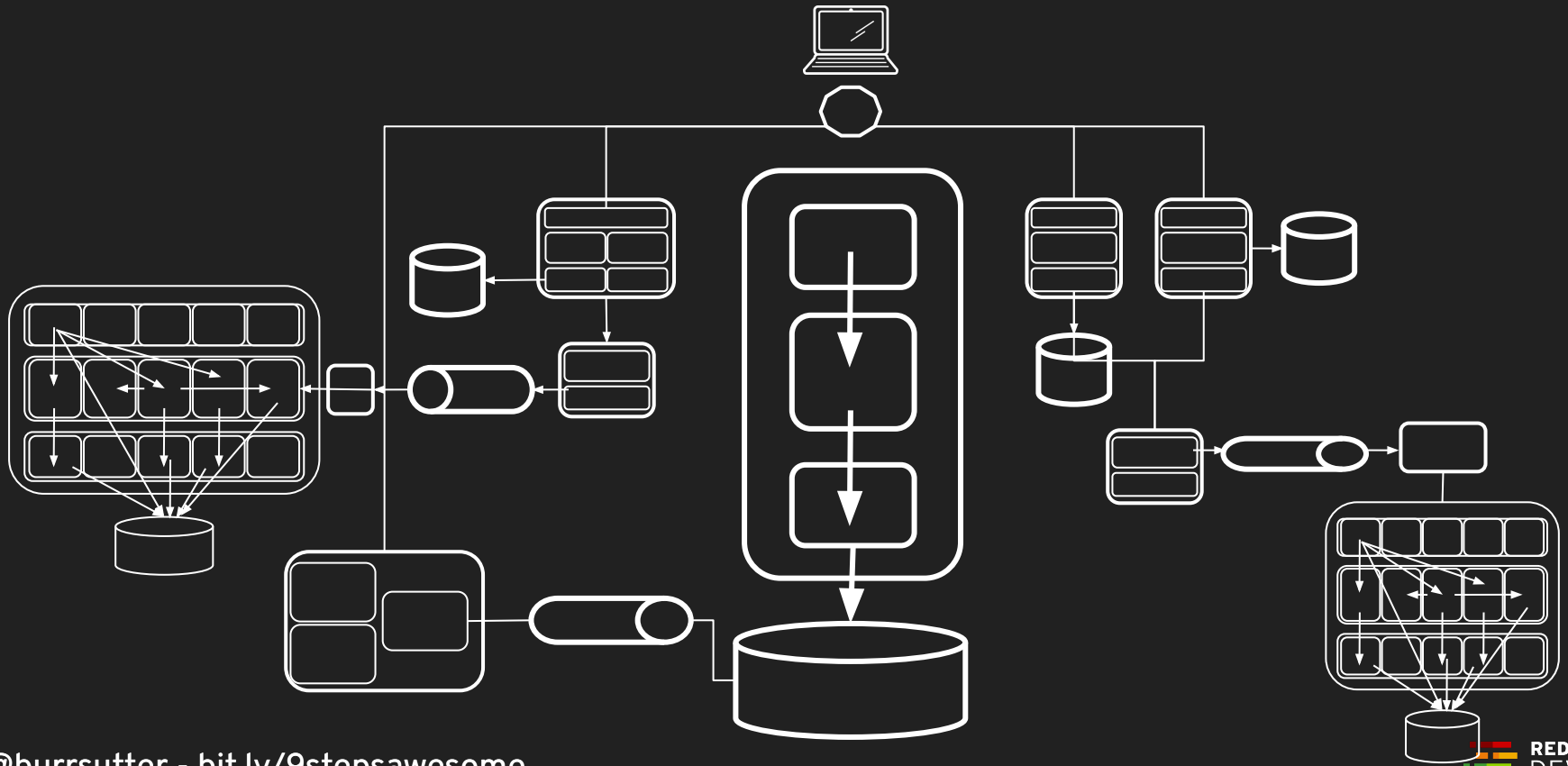
# Squeeze



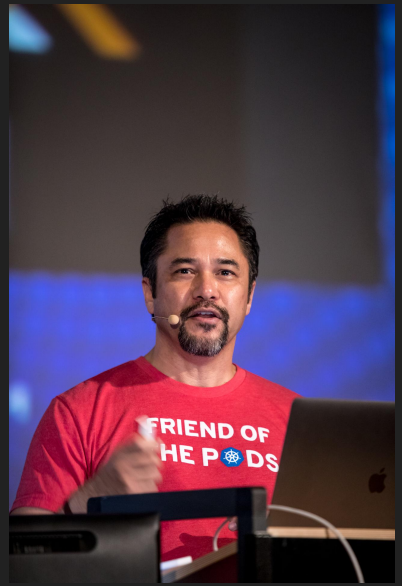
# Streamline



# Embrace Others



# Recording from Devoxx BE 2018: https://www.youtube.com/watch?v=ZpbXSdzp_vo


"excellent stuff.. worth 3 hours spending to watch this video.. Really loved it. Will give a complete picture on Kubernetes and how it works" - Pradeep


http://www.g-pop.net/fistoffury.htm

From: [redacted]@microsoft.com>
Date: Thu, Nov 21, 2019 at 6:18 PM
Subject: 119112124004928 | Microsoft Azure Support | Quota request for Compute-VM (cores-vCPUs) subscription limit increases
To: <burrsutter@gmail.com>
Cc: [redacted]@microsoft.com>

Microsoft Support

---

Support

Hello Burr,

Thank you for contacting Microsoft Support. My name is [redacted] I am the Support Professional who will be working with you on this Service Request. You may reach me using the contact information listed below, referencing the SR number 11911212404928.

Thank you for requesting additional quota in the US East (EUS) region. Due to high demand for virtual machines in this region, we are not able to immediately approve your quota request without more information.
• Can you redirect your deployments to alternative REGION: US West 2
• Can you start with a smaller # of cores

[redacted]

**Support Engineer | Microsoft Billing and Subscription Support**
*[redacted]@microsoft.com | Manager:[redacted]@microsoft.com
Working hours: 8am - 4:30pm Mon to Fri MST (UTC - 6) | Local Time
Here's a little housekeeping information regarding your request:
* Kindly **reply all** to ensure that your request is properly updated
* This case is Severity (B). You will receive an update every (24) hours until the issue is resolved.
* If you need to work with another Azure Technical Support Engineer outside of my working hours, please send email to [redacted]@microsoft.com with your case number and availability. One of the engineers will gladly continue working on this issue.