Kubernetes Fundamentals

Jakarta Kubernetes Meetup

Hello!

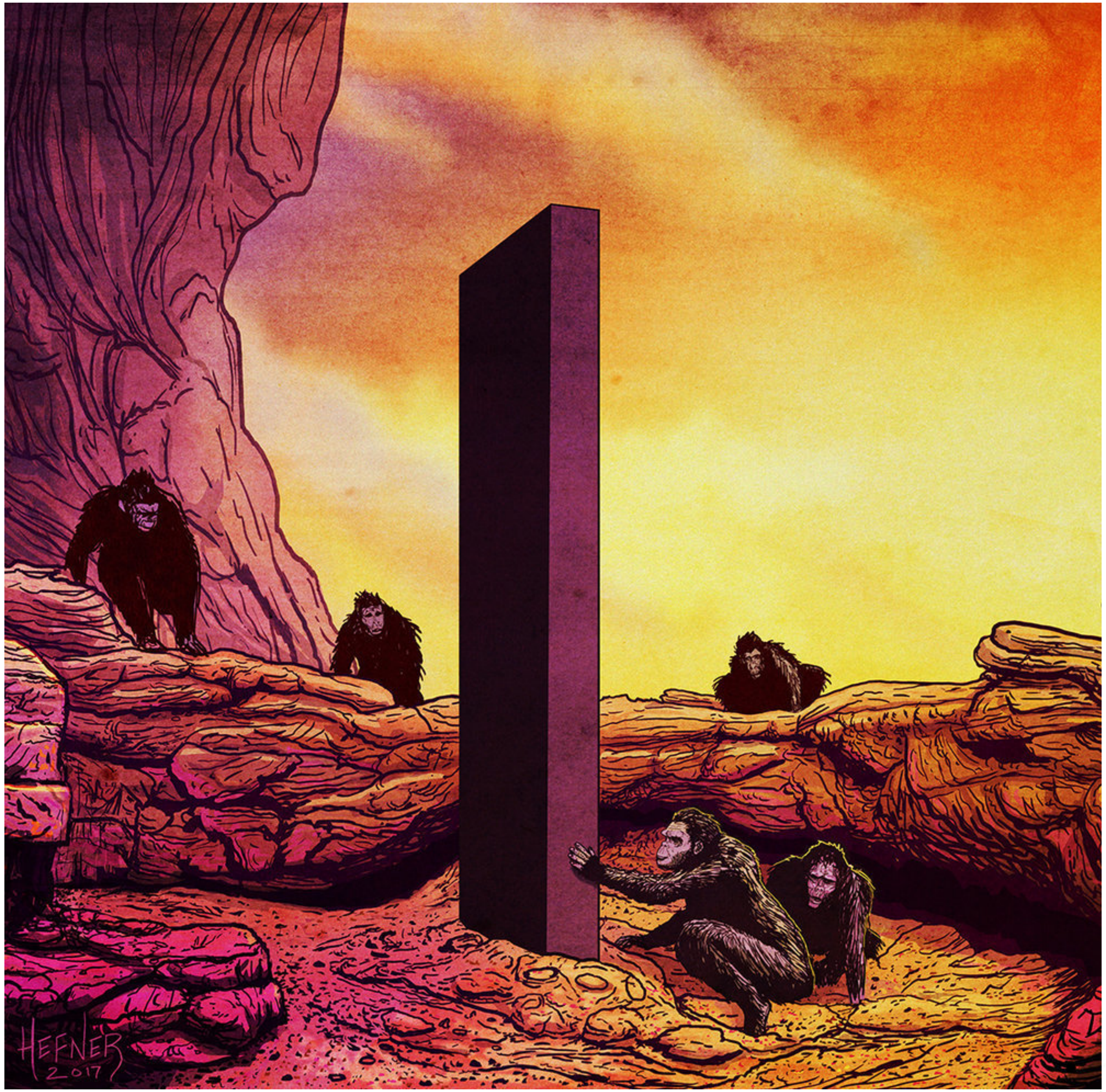
Iqbal Farabi
System Engineer
Go-Jek Indonesia

At the end of this talk…

We will have discussed about...

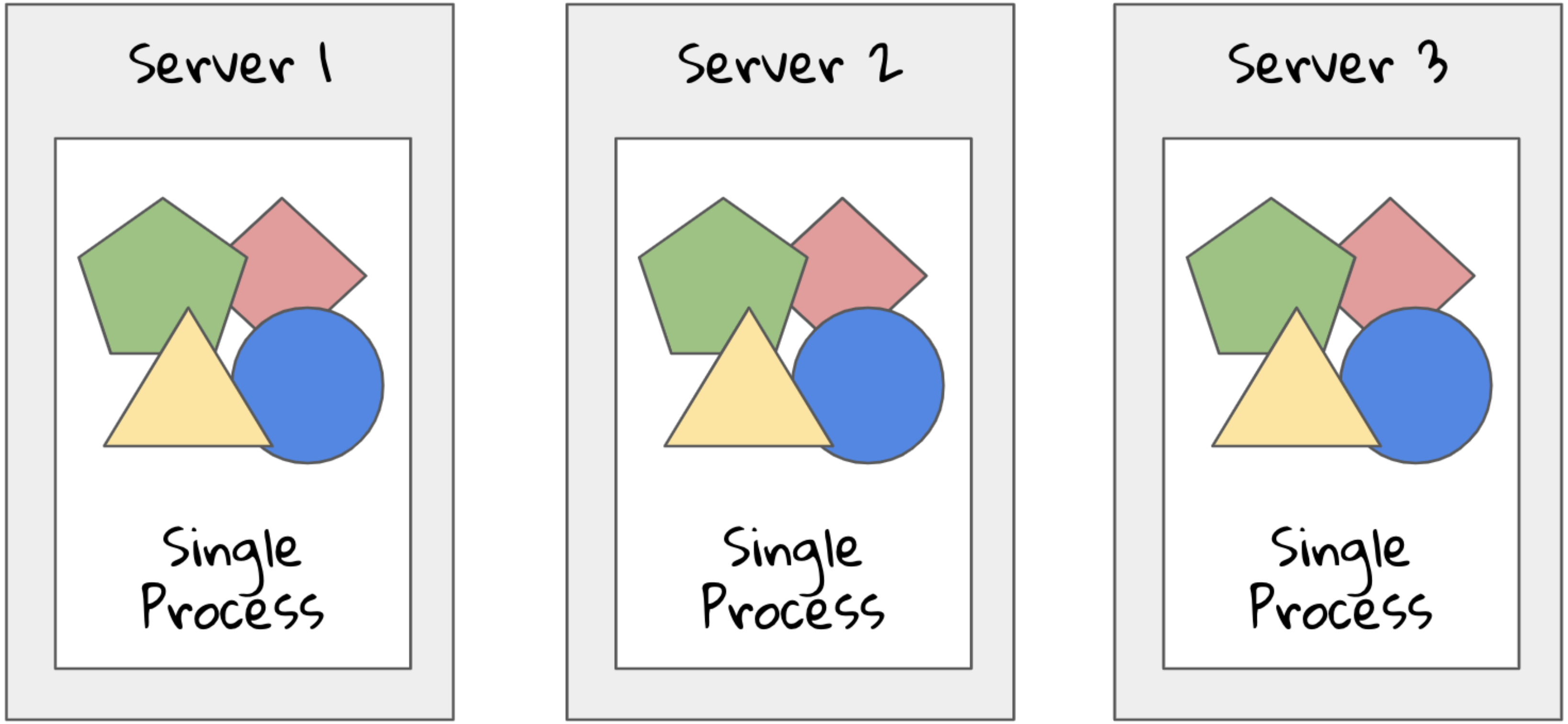
- Why a tool such Kubernetes is needed?
- Test drive with Kubernetes
- A sneak peek of how Kubernetes in Gojek
- Curriculum of Kubernetes Fundamentals talk series

Start with Why

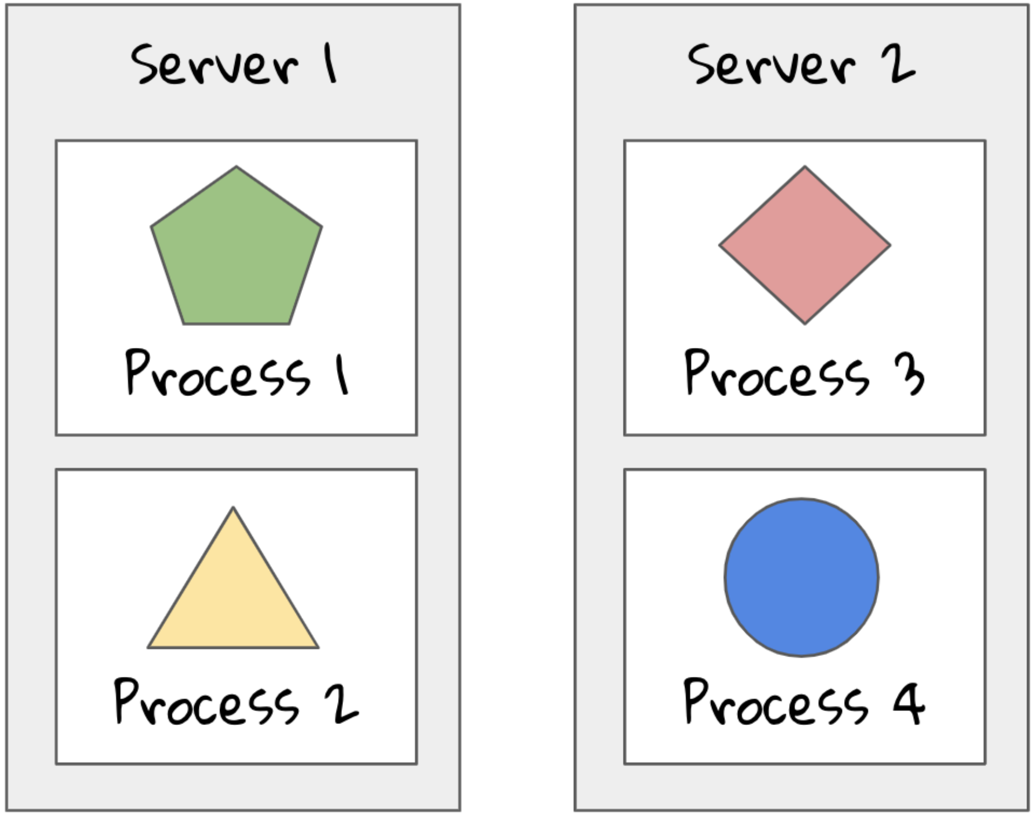

HEFNER
2017

Monolithic architecture is simple to develop, deploy, and scale.

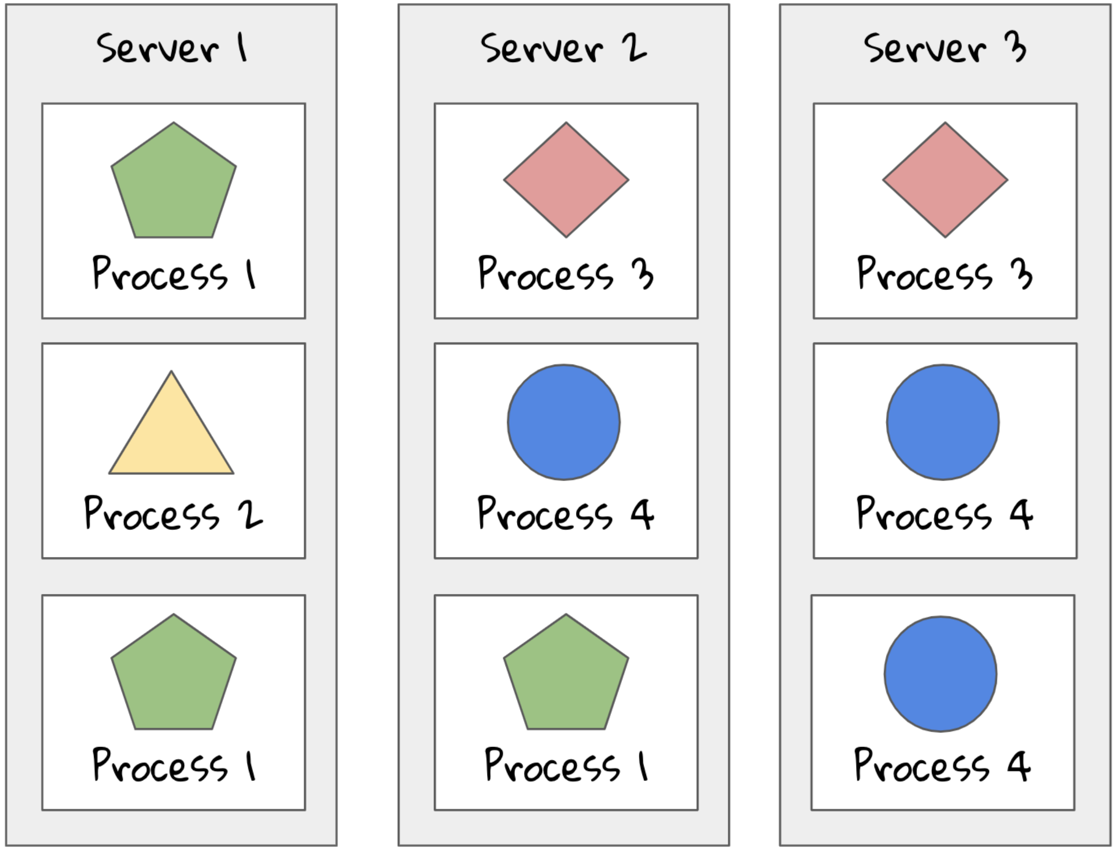
Horizontal Scaling with Monolithic Apps



along came Microservices...

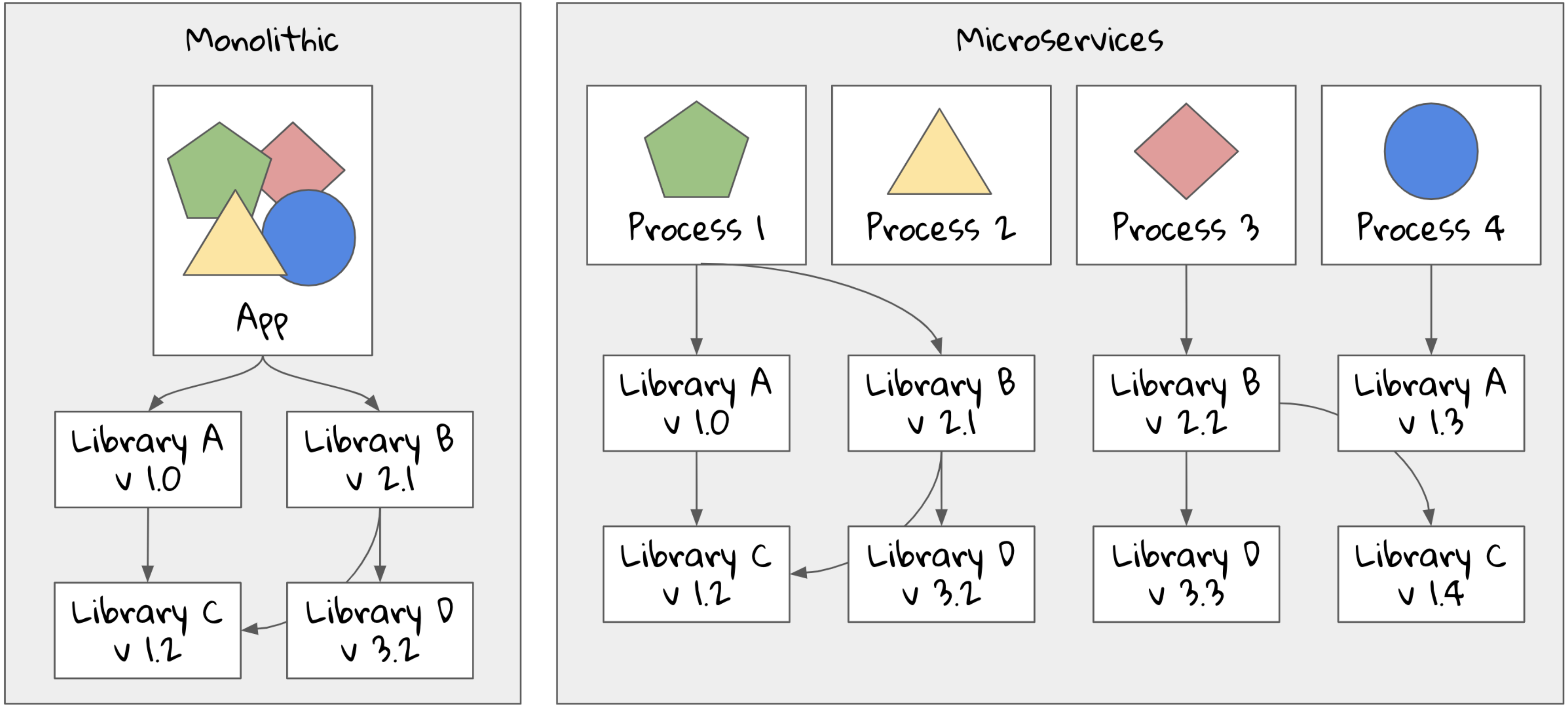


Horizontal Scaling with Microservices



But every new solution introduces new problems...

Dependencies Madness



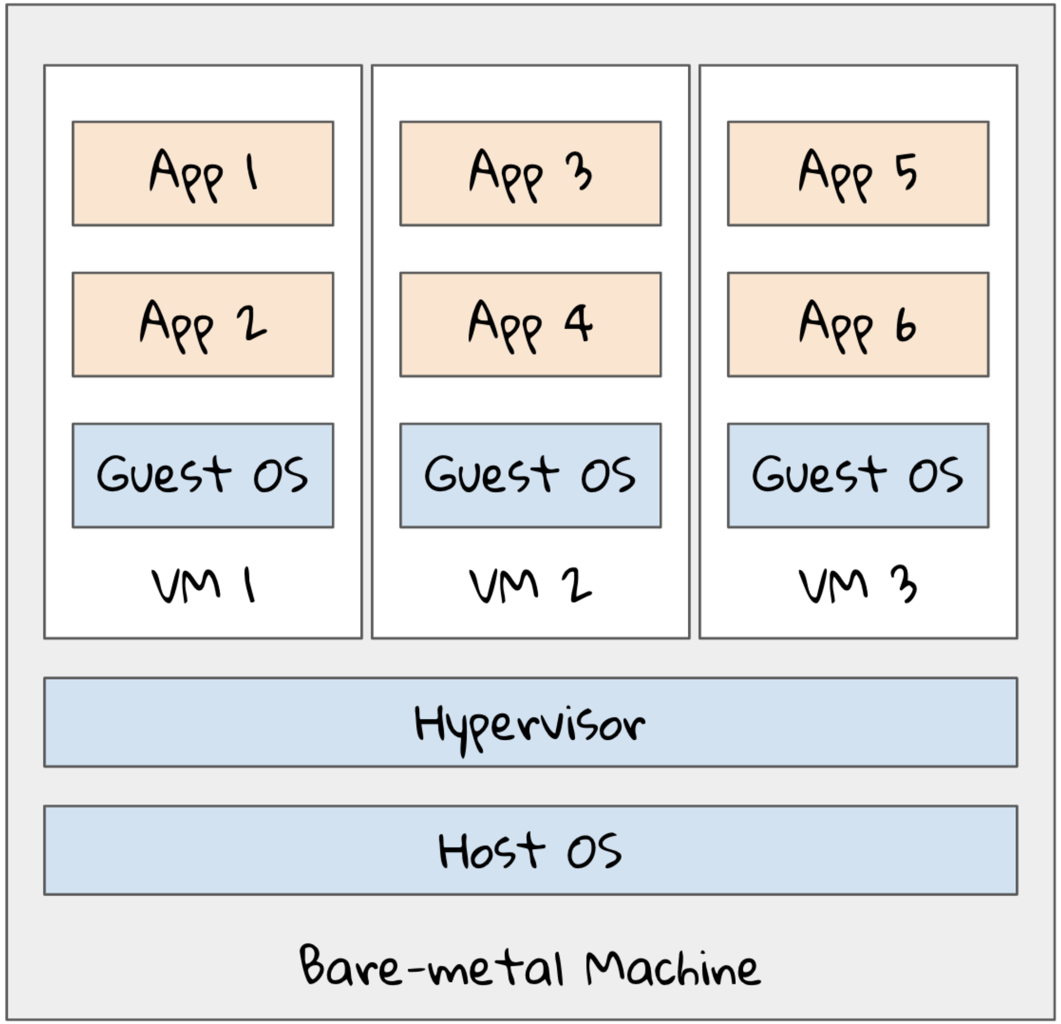
We Need Isolation

Virtual Machines

We can isolate components using Virtual Machines (VMs). This way, each components can have their own dependencies satisfied without getting in the way of each other.

The problem with VM is that it takes a lot of hardware resources, therefore not ideal for microservice-based app with large number of services.

Virtual Machines



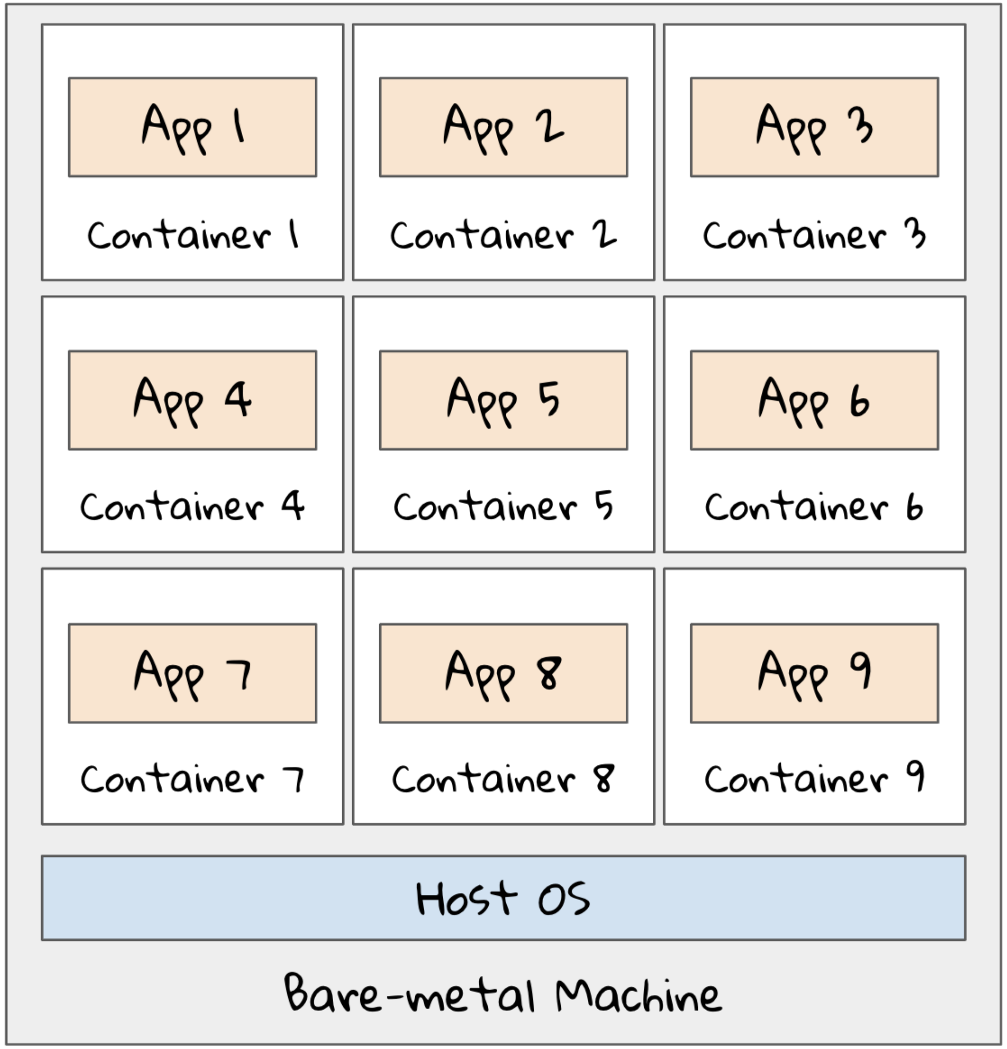
Containers

Containers run as isolated process on host OS instead of running its own guest OS. This is achieved by using Linux namespaces, cgroup, and chroot*.

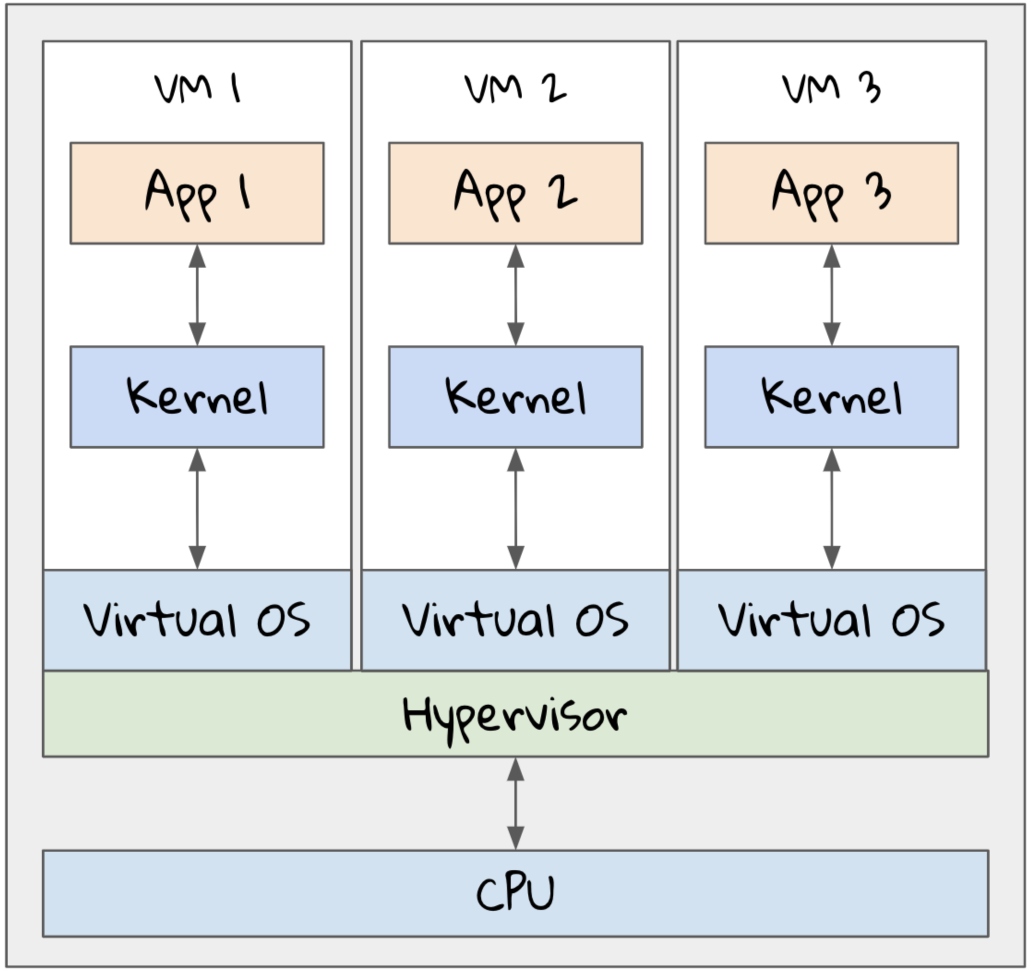
That way, containers provide isolation without consuming as much resources as VMs. With the same specs, a bare-metal host can run more containers than VMs.

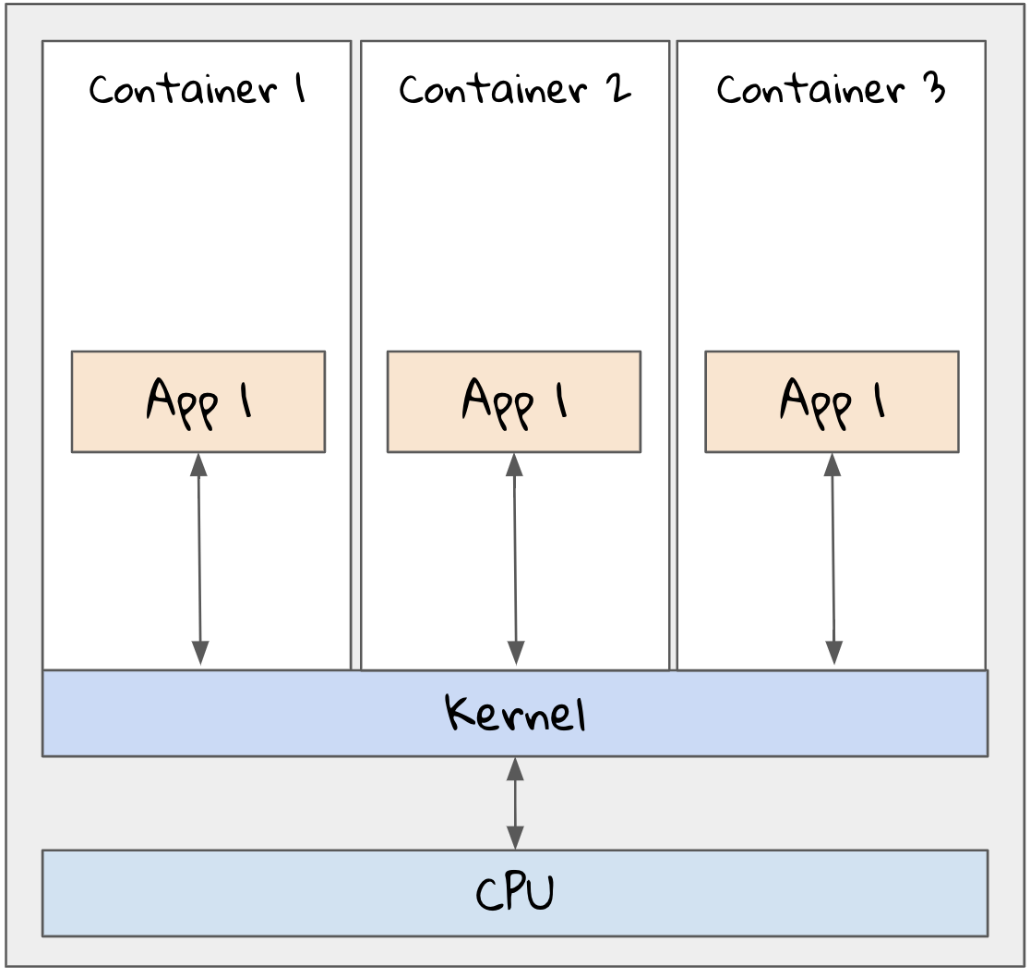
* checkout: Building Containers from Scratch

Containers


App 1
Container 1
App 2
Container 2
App 3
Container 3
App 4
Container 4
App 5
Container 5
App 6
Container 6
App 7
Container 7
App 8
Container 8
App 9
Container 9
Host OS
Bare-metal Machine

VMs and Containers





Kubernetes

kubernetes

Greek for pilot, helmsman, governor.

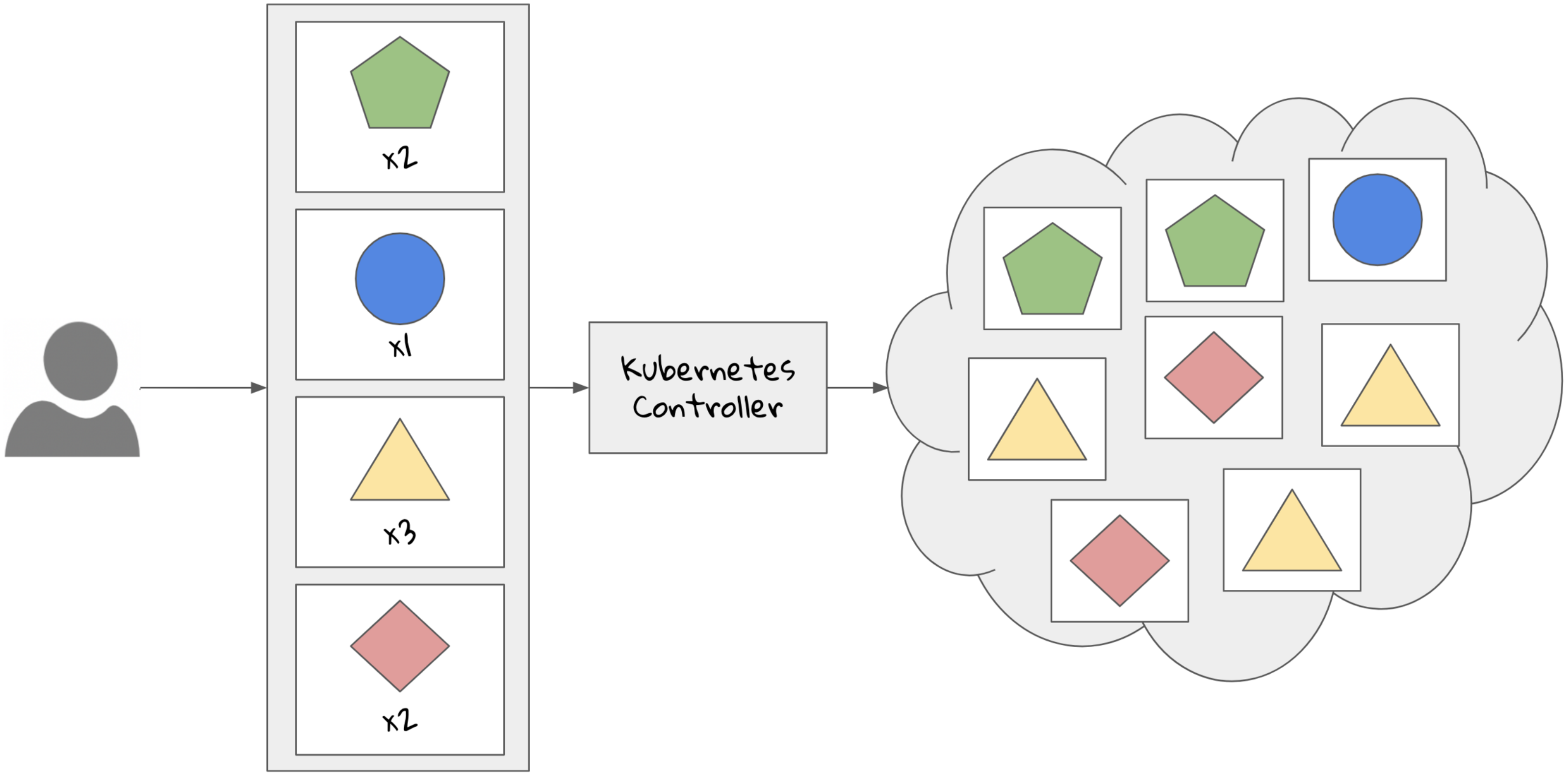
In a brief...

Kubernetes is a software system that allows you to **deploy** and **manage containerized applications** on top of it.

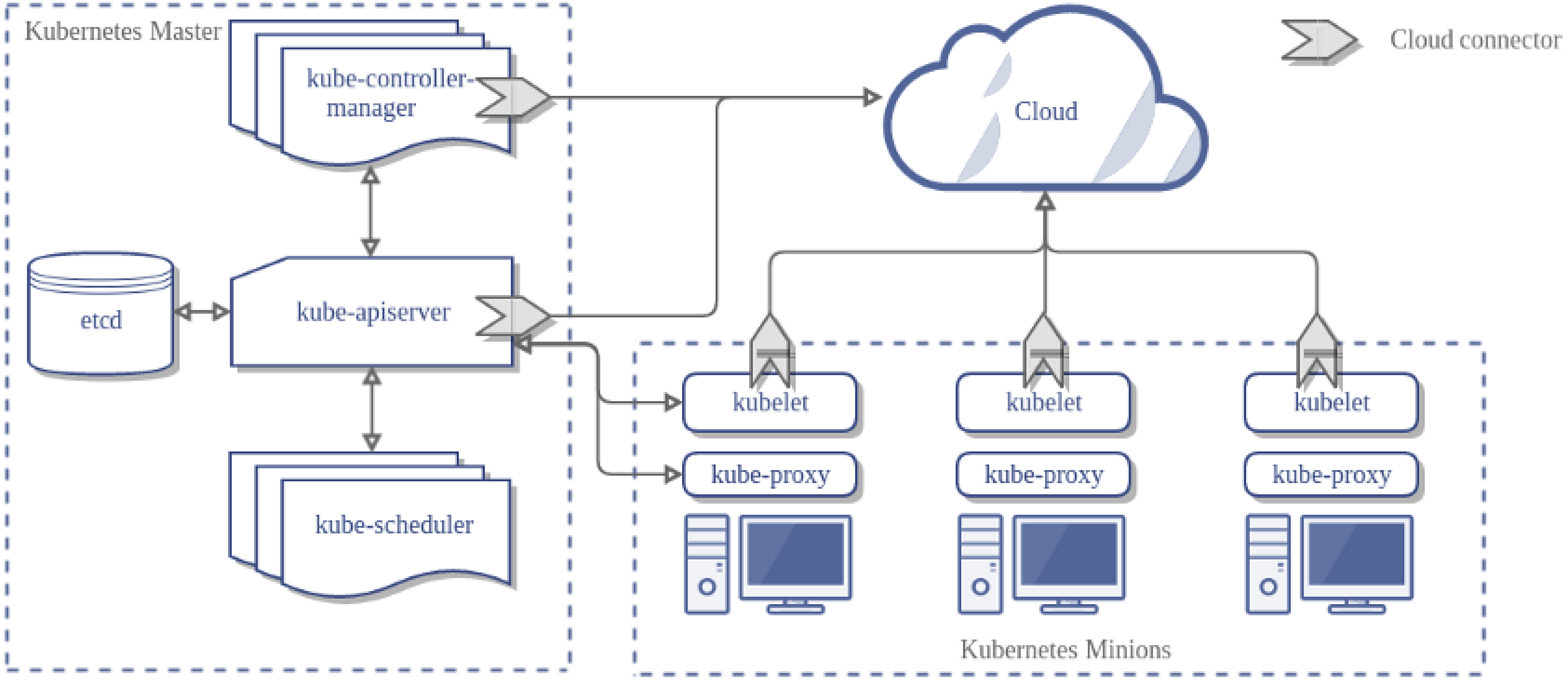
Kubernetes enables you to **run** your software applications **on multiple distributed nodes** as if all those nodes were a single, enormous resource.

Kubernetes can be thought of as an **operating system for the cluster**.

In a picture...



In an official diagram...



In action...

The Driving-a-Car Philosophy

It’s like learning to drive a car. In the beginning, you don’t really know what’s under the hood. You first want to learn how to drive it from point A to point B.

Only after you learn how to do that do you are interested in how a car makes that possible. After all, knowing what’s under the hood may someday help you get the car moving again after it breaks down and leaves you stranded at the side of the road.

Let's Ride

1 - Preparation

Install Kubernetes

```
brew install kubernetes-cli
```

For our test drive, let's just use minikube.

```
brew cask install minikube
```

Verify everything is ok

```
kubectl config get-contexts
```

2 - Deploy a Simple App

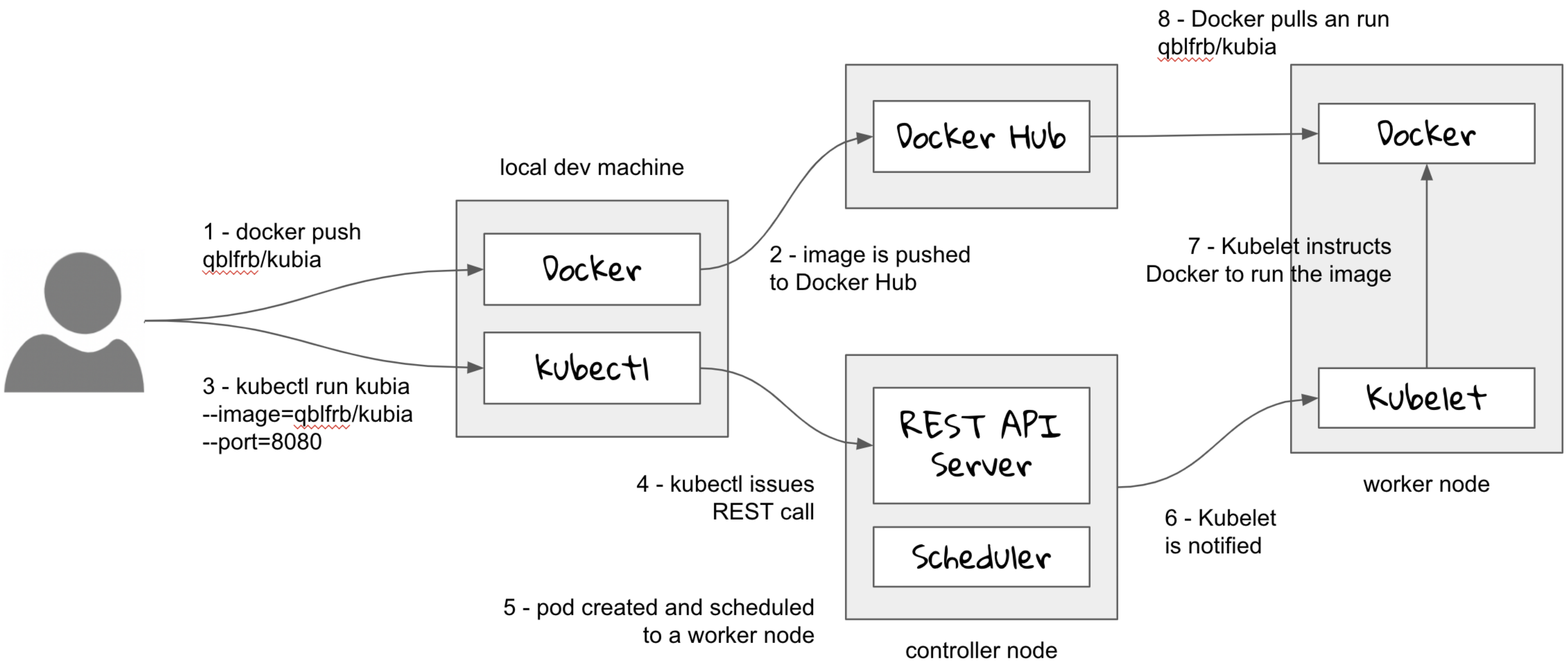
Use `kubectl run command`

```
kubectl run kubia --image=qblfrb/kubia --port=8080 --generator=run/v1
```

See the result

```
kubectl get pods
```

What Just Happened?



8 - Docker pulls an run qblfrb/kubia
Docker Hub
Docker
local dev machine
1 - docker push qblfrb/kubia
Docker
kubectl
2 - image is pushed to Docker Hub
7 - Kubelet instructs Docker to run the image
3 - kubectl run kubia --image=qblfrb/kubia --port=8080
REST API Server
Kubelet
4 - kubectl issues REST call
Scheduler
worker node
6 - Kubelet is notified
5 - pod created and scheduled to a worker node
controller node

3 - Expose The App

Get the replication controller:

```
kubectl get replicationcontroller
```

Expose it:

```
kubectl expose rc kubia --type=LoadBalancer --name kubia-http --port=8080
```

Test it:

```
minikube service kubia-http
```

Vocabularies

Pod

A co-located group of containers. Represents the basic building block in Kubernetes.

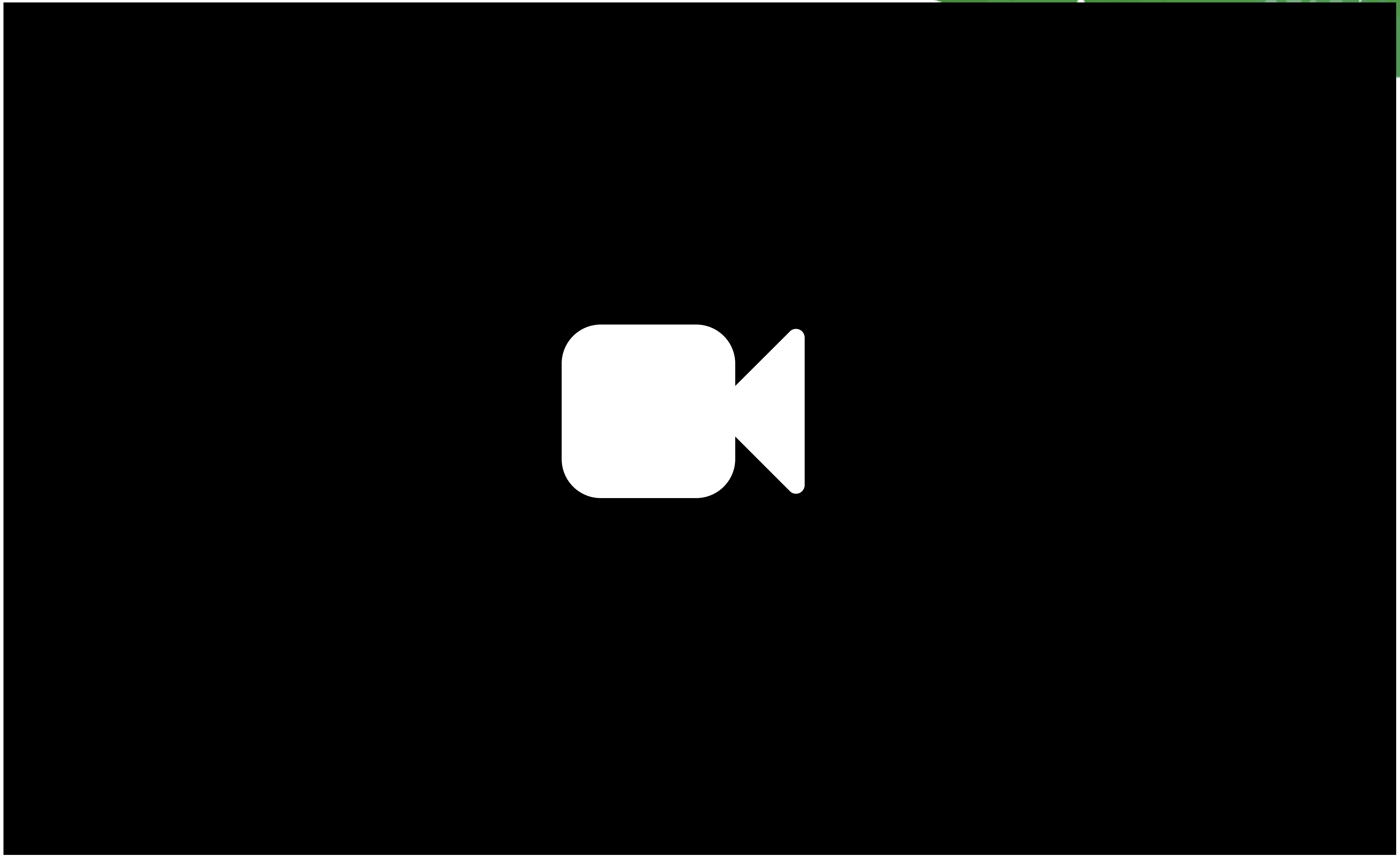
Replica Controller

A Kubernetes resource that ensures a desired number of pods are always kept running.

Service

A resource that serves a a single, constant point of entry to a group of pods providing the same service. Each service has an IP address and port that never change while the service exists.

Deploying a Sinatra App to Kubernetes



1 - A Simple Sinatra App

Hello world:

```
require 'sinatra'

enable :run, :show_exceptions

set :environment, :production
set :bind, '0.0.0.0'
set :port, 80

get '/' do
  'Hello, world!'
end
```

2 - Container Image

Write a Dockerfile:

```
FROM ruby:2.5.1

RUN gem install sinatra

WORKDIR /usr/src/app
COPY hello-world.rb .

EXPOSE 80

CMD /usr/local/bin/ruby ./hello-world.rb
```

Build and push to Dockerhub:

```
docker build -t qblfrb/hello-world-sinatra:0.1.0 .
docker push qblfrb/hello-world-sinatra:0.1.0
```

3 - Kubernetes Manifest (1)

Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sinatra
spec:
  selector:
    matchLabels:
      app: sinatra
  replicas: 1
  template:
    metadata:
      labels:
        app: sinatra
    spec:
      containers:
        - name: sinatra
          image: qblfrb/hello-world-sinatra:0.1.0
          ports:
            - containerPort: 30080
```

3 - Kubernetes Manifest (2)

Service:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: sinatra
  labels:
    app: sinatra
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30080
      protocol: TCP
      name: http
  selector:
    app: sinatra
```

4 - Run

Apply:

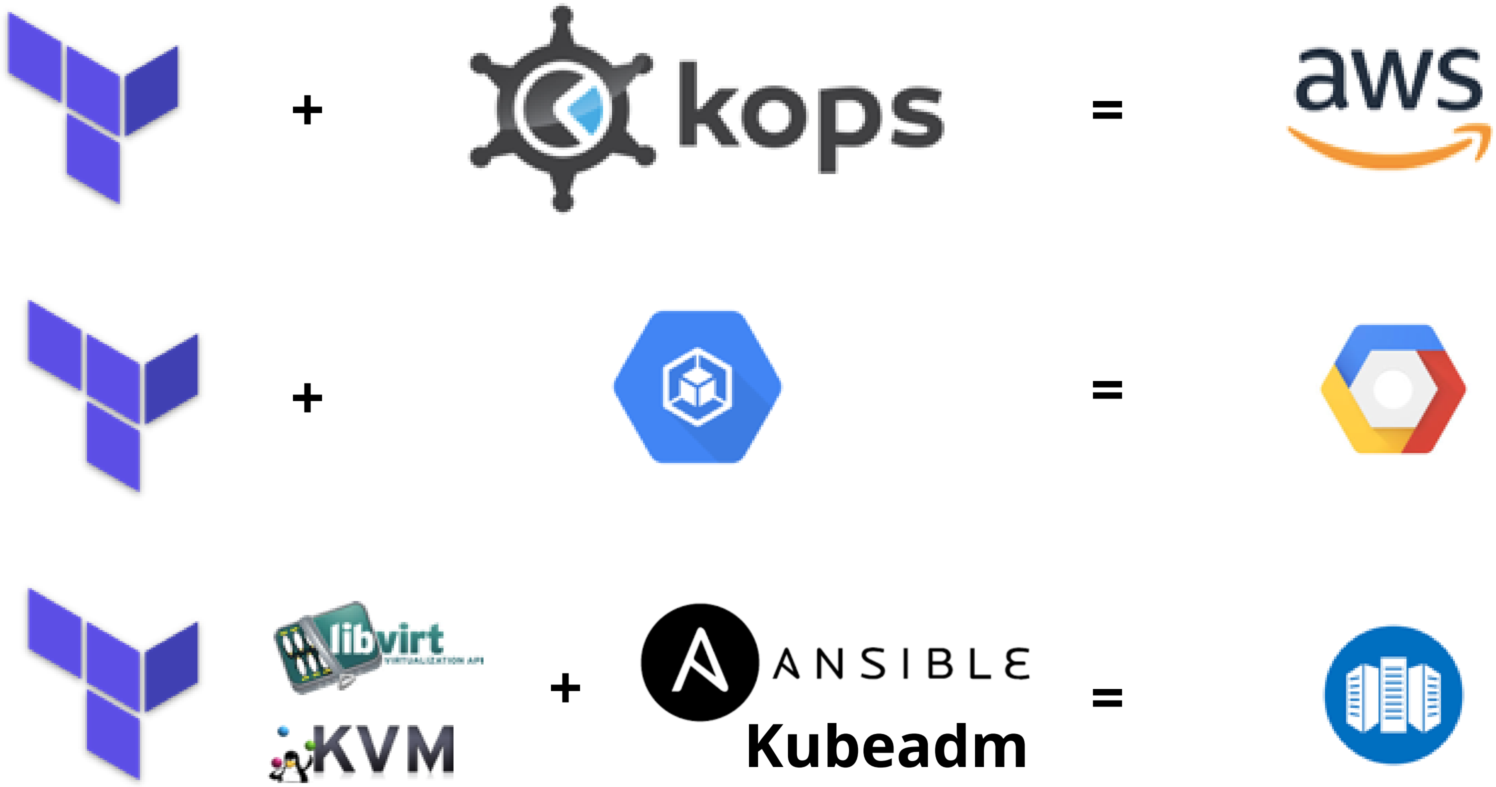
```
kubectl apply -f sinatra.yaml
```

Port-forward:

```
kubectl get pods -l app=sinatra
kubectl port-forward sinatra-788f79cff4-fpqxw 8080:80
```

Kubernetes in Go-Jek

Multiple Approaches



Kubeadm

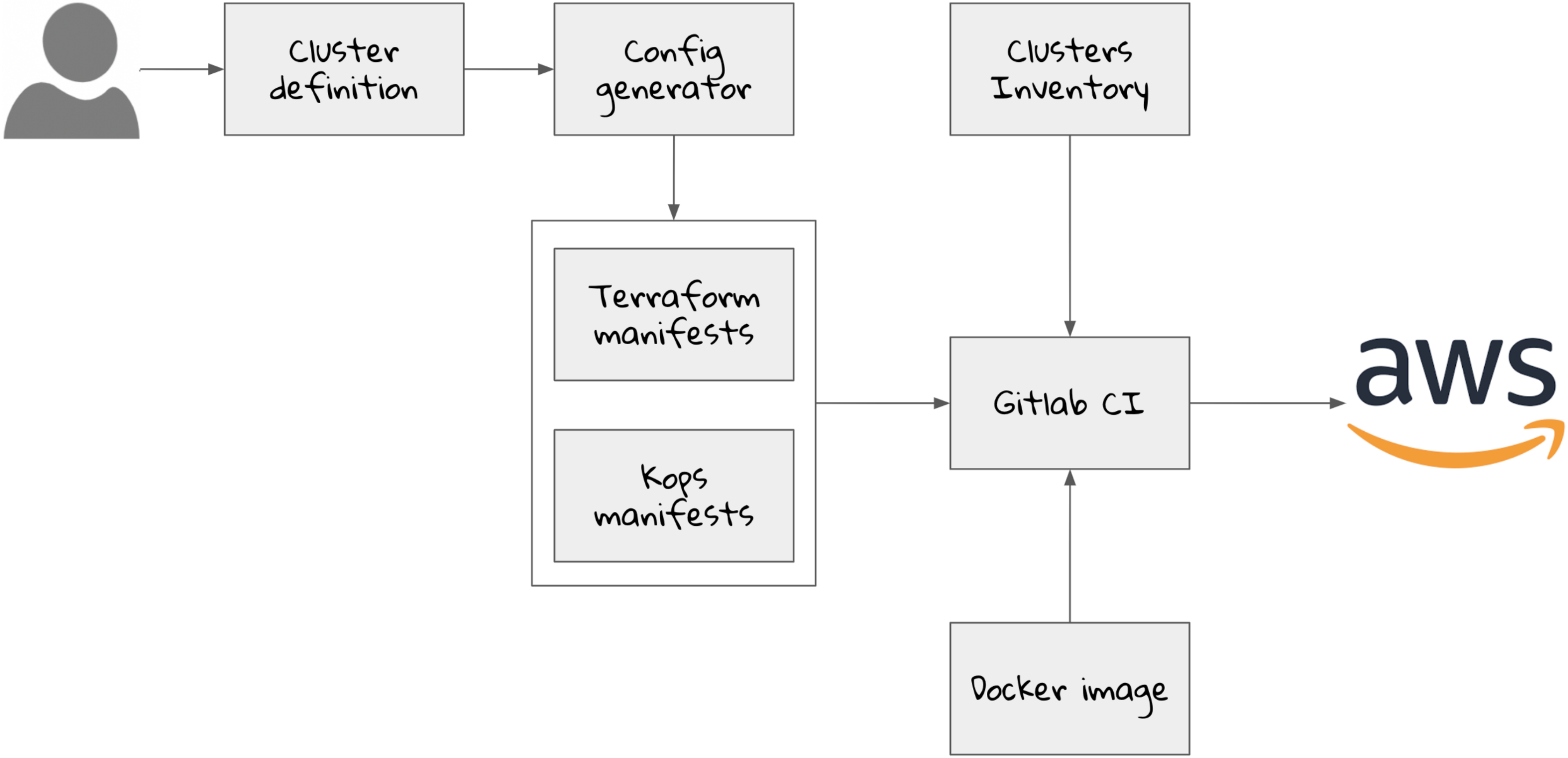
The Bootstrapper

The Bootstrapper is a tool that we created to empower developers in managing their own Kubernetes cluster on AWS. This is only one of many Kubernetes related tools we have in Go-Jek.

We implement the reconciler pattern as described in Kris Nova's book Cloud Native Infrastructure. We utilize Terraform and Kops to automate cluster creation, update, and deletion.

Simple things should be simple, complex things should be possible.

- Alan Kay



Cluster definition

Config generator

Clusters Inventory

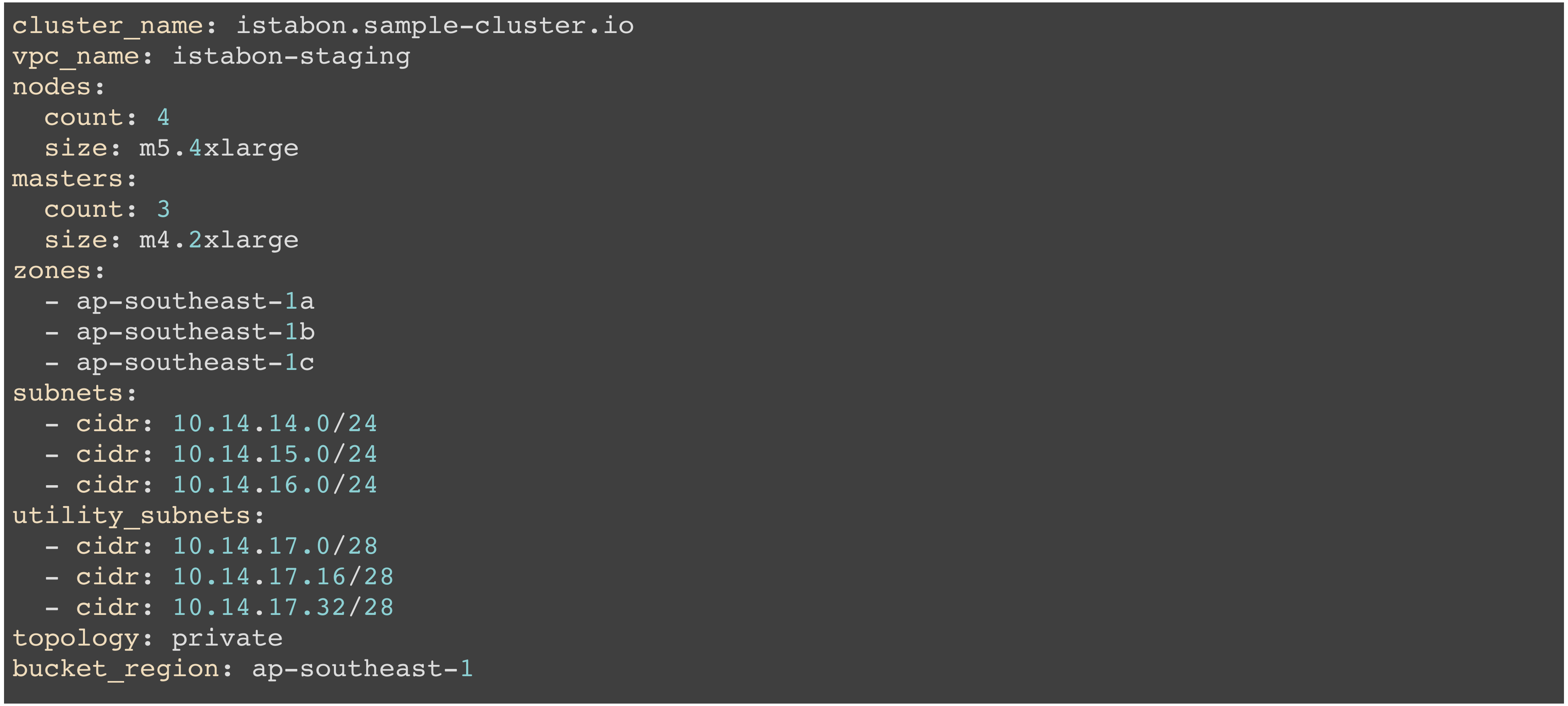
Terraform manifests

Kops manifests

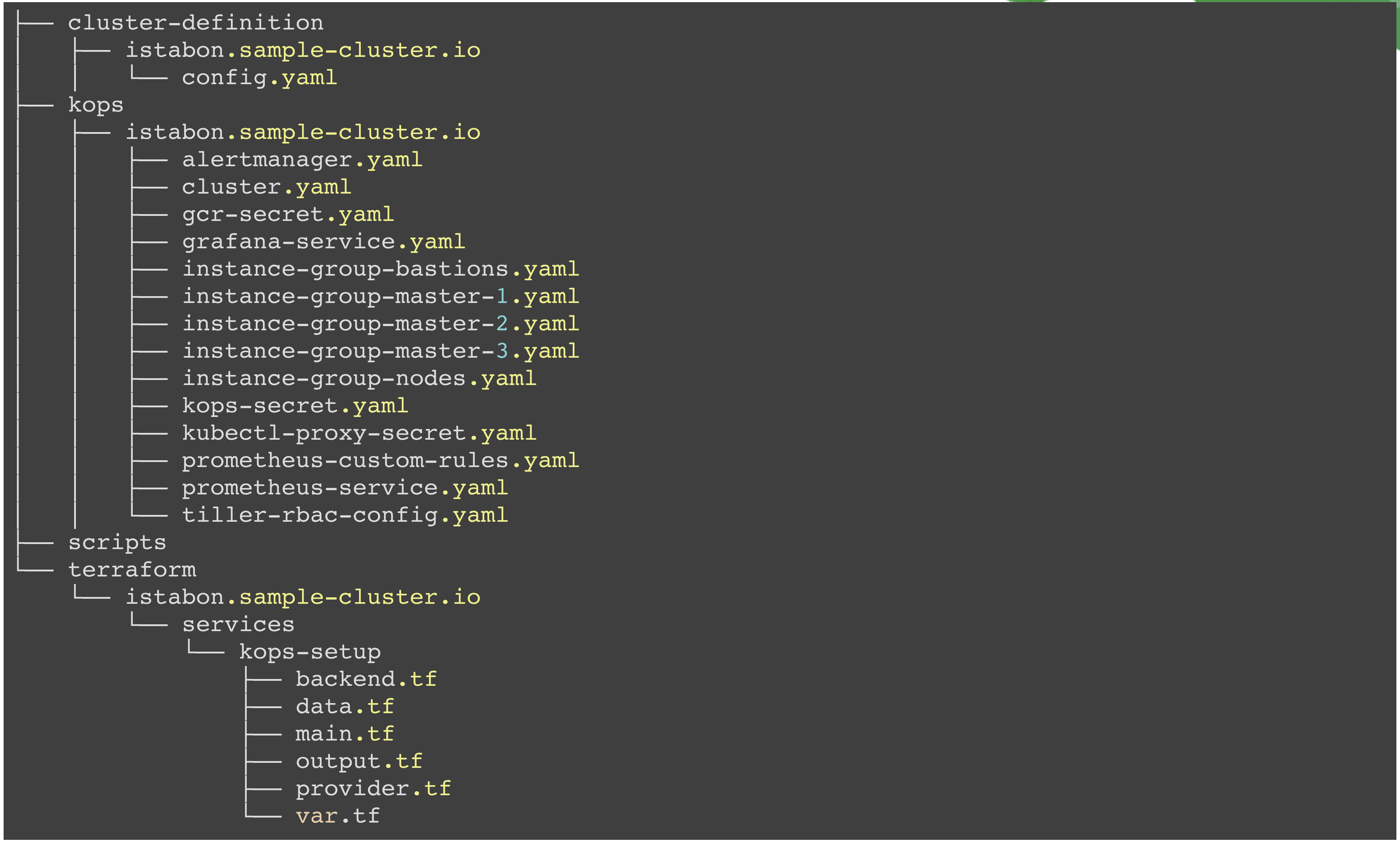
Gitlab CI

aws

Docker image



```
cluster_name: istabon.sample-cluster.io
vpc_name: istabon-staging
nodes:
  count: 4
  size: m5.4xlarge
masters:
  count: 3
  size: m4.2xlarge
zones:
  - ap-southeast-1a
  - ap-southeast-1b
  - ap-southeast-1c
subnets:
  - cidr: 10.14.14.0/24
  - cidr: 10.14.15.0/24
  - cidr: 10.14.16.0/24
utility_subnets:
  - cidr: 10.14.17.0/28
  - cidr: 10.14.17.16/28
  - cidr: 10.14.17.32/28
topology: private
bucket_region: ap-southeast-1
```



```
├── cluster-definition
│   ├── istabon.sample-cluster.io
│   │   └── config.yaml
├── kops
│   ├── istabon.sample-cluster.io
│   │   ├── alertmanager.yaml
│   │   ├── cluster.yaml
│   │   ├── gcr-secret.yaml
│   │   ├── grafana-service.yaml
│   │   ├── instance-group-bastions.yaml
│   │   ├── instance-group-master-1.yaml
│   │   ├── instance-group-master-2.yaml
│   │   ├── instance-group-master-3.yaml
│   │   ├── instance-group-nodes.yaml
│   │   ├── kops-secret.yaml
│   │   ├── kubectl-proxy-secret.yaml
│   │   ├── prometheus-custom-rules.yaml
│   │   ├── prometheus-service.yaml
│   │   └── tiller-rbac-config.yaml
├── scripts
└── terraform
    └── istabon.sample-cluster.io
        └── services
            └── kops-setup
                ├── backend.tf
                ├── data.tf
                ├── main.tf
                ├── output.tf
                ├── provider.tf
                └── var.tf
```

Cluster with Benefits

Benefits (1)

Faster Setup Time

Setting up the whole Go-Viet infrastructure only took four days.

Cookie Cutter Model

Repeatable/immutable nature of containerizing helps us to replicate our MVP launch strategy for different geographies.

Scalable

Scaling based on business growth is very easy.

Benefits (2)

Faster MTTR

In the case of traffic spike, for instance, we can spin up new containers much more quickly than setting up new VMs.

Higher Uptime

High availability setup lead to fewer outage.

Efficiency

System resources like CPU, memory, etc. are more effectively utilized in container world than in VMs.

Benefits (3)

Easy Configuration

Automatic service discovery allows engineers to not maintain any configuration for multi-data center deployments.

Cost Effective

Save > 60% cost compared to VM per year per country for international expansion projects.

Kubernetes Fundamentals Curriculum

The Series

We, Jakarta Kubernetes community organizers believe that we should dedicate one talk in every meetup to help people who are new to Kubernetes to learn together with the community.

The curriculum of Kubernetes Fundamentals series will be derived from several sources such as:

- Kubernetes Up and Running
- Kubernetes in Action
- Linux Foundation's Kubernetes Fundamentals Course
- etc

The Curriculum

Initial, but not definitive curriculum looks something like this:

- Kubernetes Basics
- Installation and Configuration
- Kubernetes Architecture
- APIs and Access
- API Objects
- Managing State with Deployments
- Services
- Volumes and Data
- Ingress
- Scheduling
- Logging and Troubleshooting

References and Reading Materials

- Kubernetes in Action – Mario Luksa
- Kubernetes: Up and Running – Joe Beda, Brendan Burns, Kelsey Hightower
- Cloud Native Infrastructure – Kris Nova, Justin Garrison
- Building Microservices – Sam Newman
- Designing Distributed System – Brendan Burns

Thank You!