# Lecture 16: Operations - Scaling

AC215

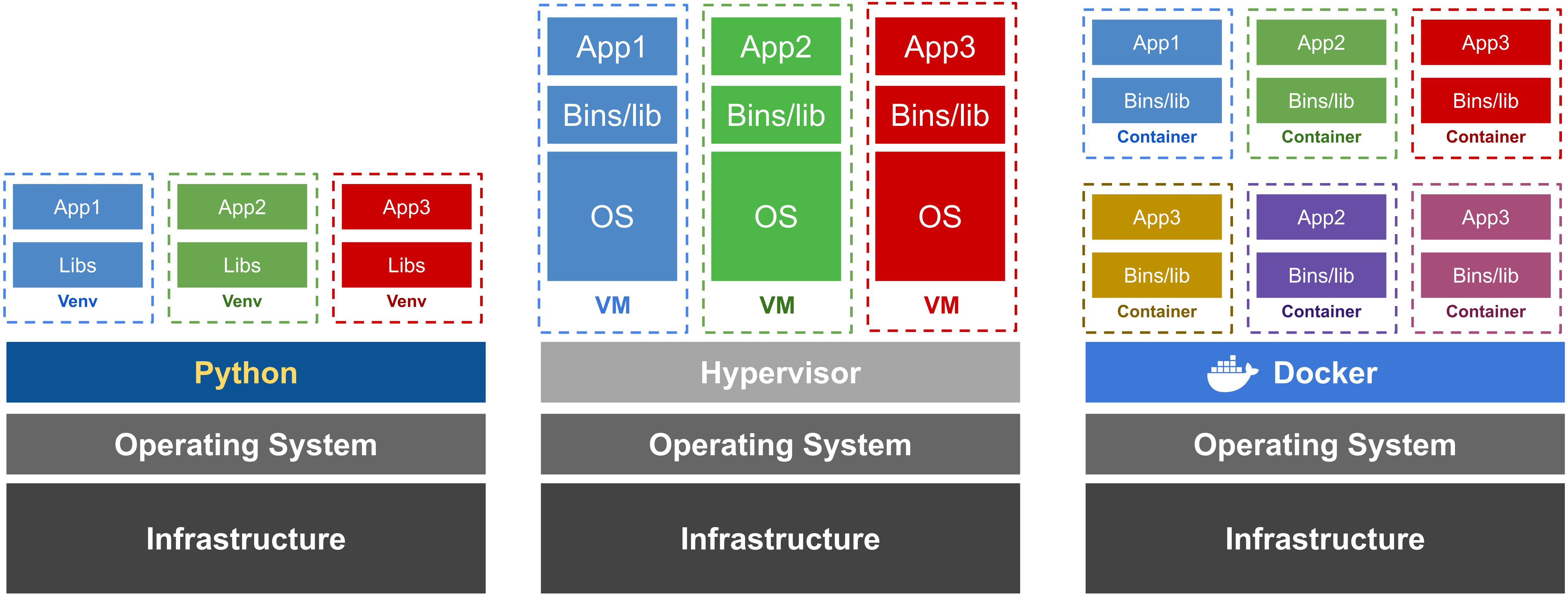
Pavlos Protopapas

SEAS/ Harvard

# Outline

1. Recap
2. Motivation
3. Introduction to Kubernetes
4. Tutorial: Deploying a Kubernetes Cluster
5. Advantages of using Kubernetes

# Recap



Virtual Environments

Virtualization

Containerization

# Recap

### Virtual Environment

**Pros:** remove complexity
**Cons:** does not isolate from OS

### Virtual Machines

**Pros:** isolate OS guest from host
**Cons:** intensive use hardware

### Containers

**Pros:** lightweight
**Cons:** issues with security, scalability, and control

# Recap

### Virtual Environment

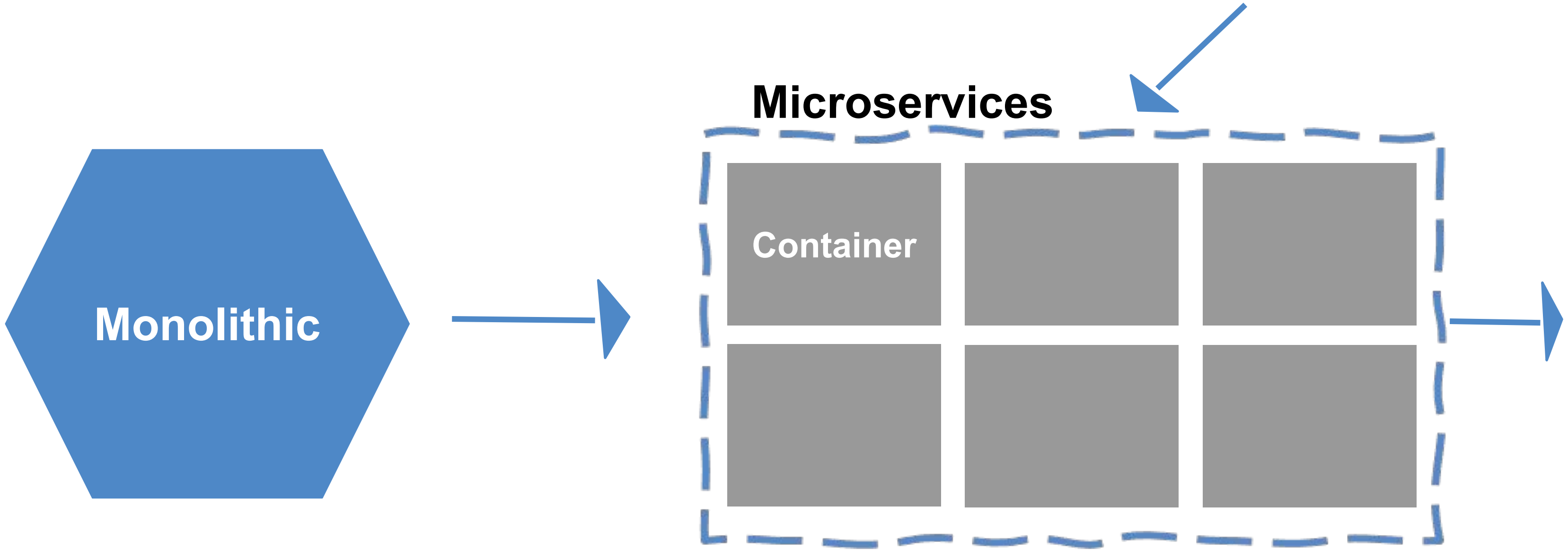
**Pros:** remove complexity
**Cons:** does not isolate from OS

### Virtual Machines

**Pros:** isolate OS guest from host
**Cons:** intensive use hardware

### Containers

**Pros:** lightweight
**Cons:** issues with security, scalability, and control



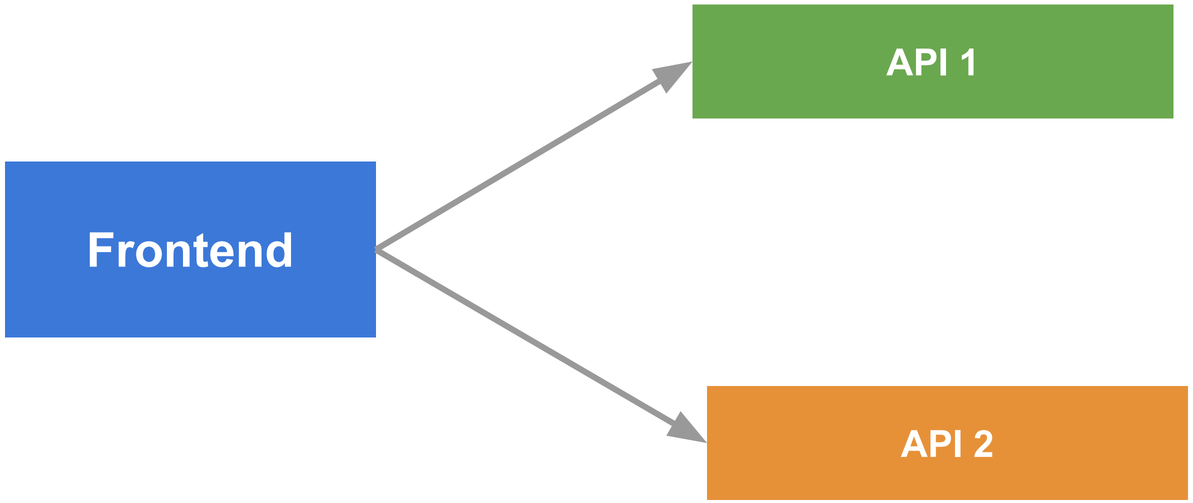
**How to manage microservices?**

# Outline

1. Recap
2. **Motivation**
3. Introduction to Kubernetes
4. Tutorial: Deploying a Kubernetes Cluster
5. Advantages of using Kubernetes

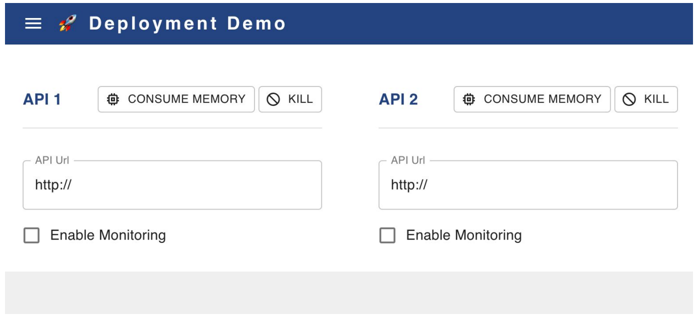
# Motivation

Pavlos wants an app with 1 frontend & 2 backends



# Motivation - 3 Containers in 1 VM

***Support*** builds and deploys the app with the following architecture

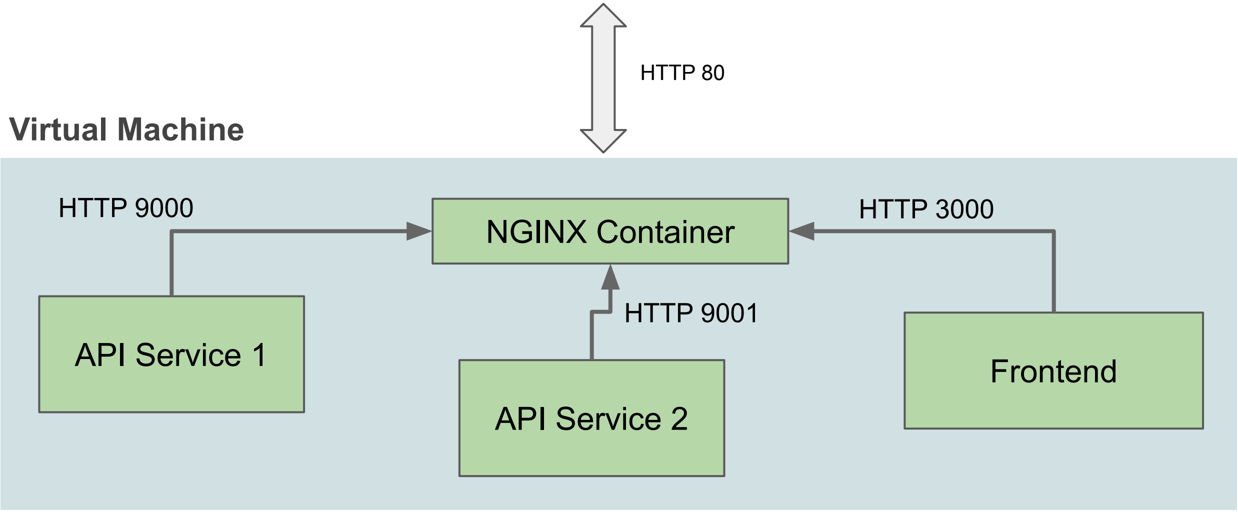


API Url: http://

☐ Enable Monitoring

API Url: http://

☐ Enable Monitoring



# Motivation - 3 Containers in 1 VM

**Demo… [3 Containers in 1 VM]**

# Motivation - 3 Containers in 1 VM

**Problems:**

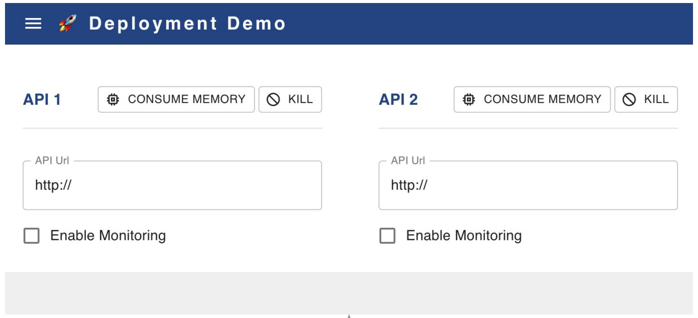
- When container crashes Pavlos has to call *support*
- Support SSHs into server and fix:
  - Memory reset with container restart
  - Startup a killed container

# Motivation - 3 Containers in 3 VM

Pavlos asks *support*: *“can we deploy the app in multiple servers so when one goes down i have a backup to use?”*

# Motivation - 3 Containers in 3 VM

Support deploys the app on to 3 servers with backup apis

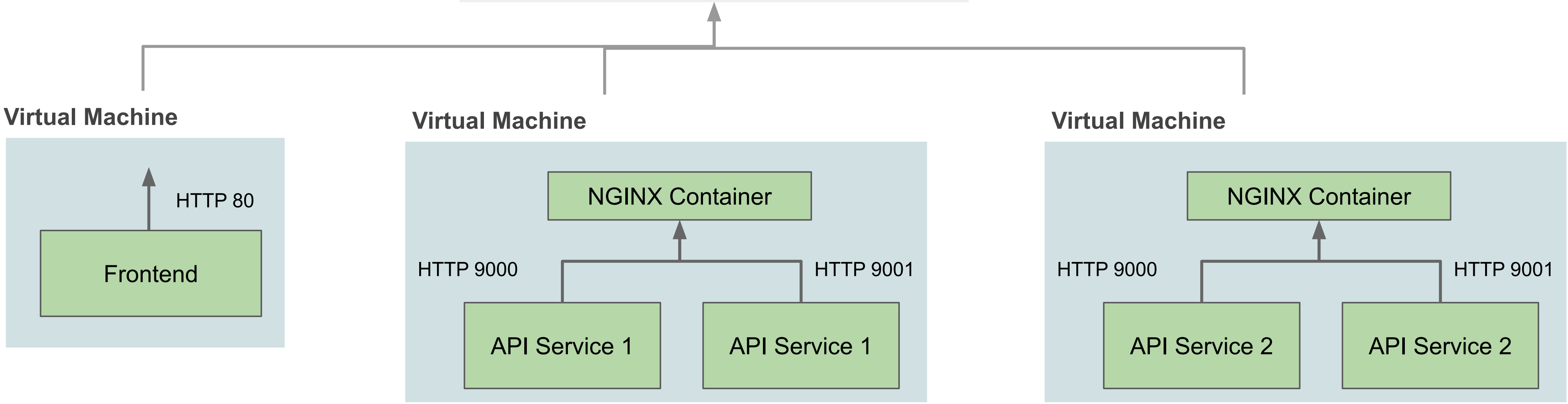


API Url
http://

□ Enable Monitoring

API Url
http://

□ Enable Monitoring



# Motivation - 3 Containers in 3 VM

**Demo… [3 Containers in 3 VMs]**

# Motivation - 3 Containers in 3 VM

## Problems:

- When container crashes, Pavlos can switch to backup API manually
- *Support* SSHs into server and fix when available:
  - Memory reset with container restart
  - Startup a killed container

# Motivation - Kubernetes

Pavlos asks: *can we automate*:

- Failovers
- Load balancing
- Scaling
- etc.

**Kubernetes to the rescue...**

# Kubernetes (K8s) to the Rescue

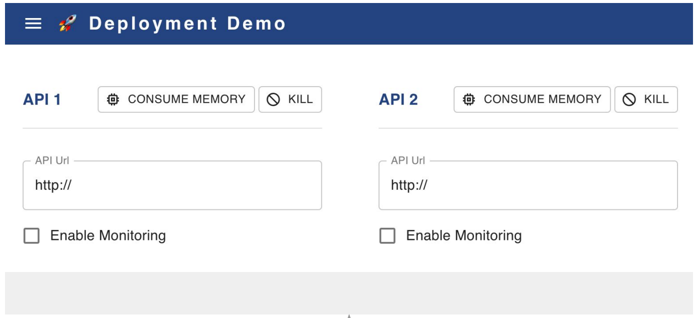
- K8s is an orchestration tool for **managing distributed containers** across a cluster of nodes (VMs).

- K8s itself follows a **client-server architecture with a master and worker nodes**. Core concepts in Kubernetes include pods, services and deployments.

- K8s **users define rules** for how container management should occur, and then K8s handles the rest!

# Kubernetes to the Rescue

*Support* deploys the app on to 3 k8s clusters with 2 nodes each



Deployment Demo

API 1 — CONSUME MEMORY — KILL

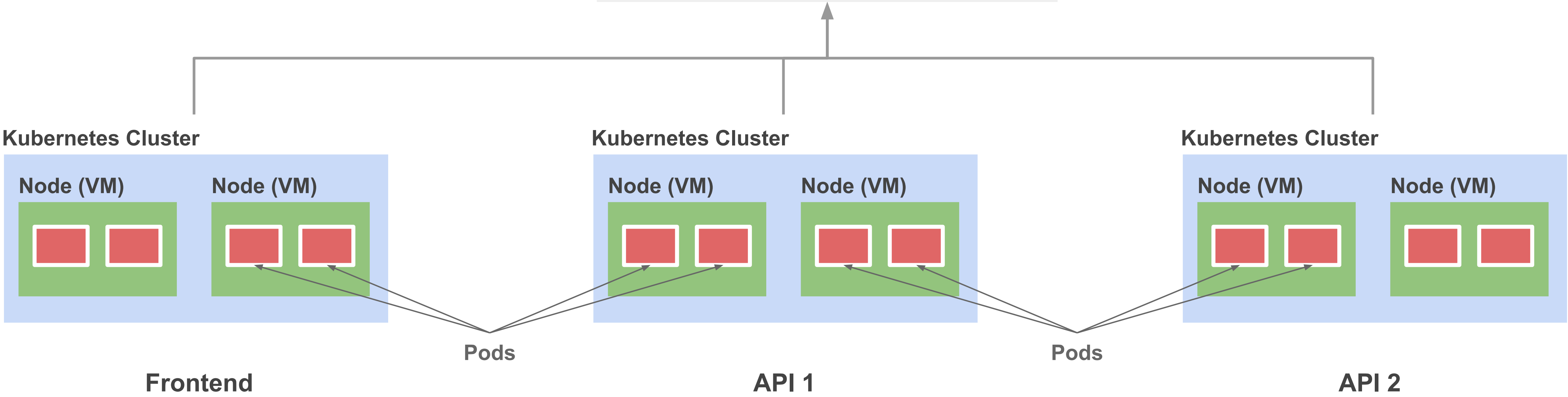
API Url: http://

□ Enable Monitoring

API 2 — CONSUME MEMORY — KILL

API Url: http://

□ Enable Monitoring



# Kubernetes to the Rescue

**Demo… [Kubernetes Cluster]**

# Kubernetes

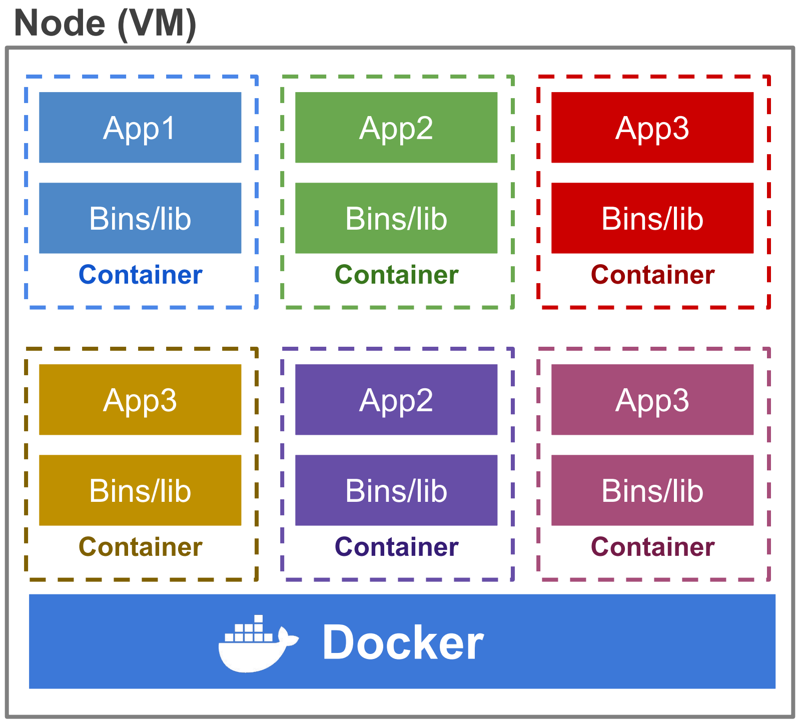
Pavlos requests on automation:

- ✓ Failovers
- ✓ Load balancing
- ✓ Scaling

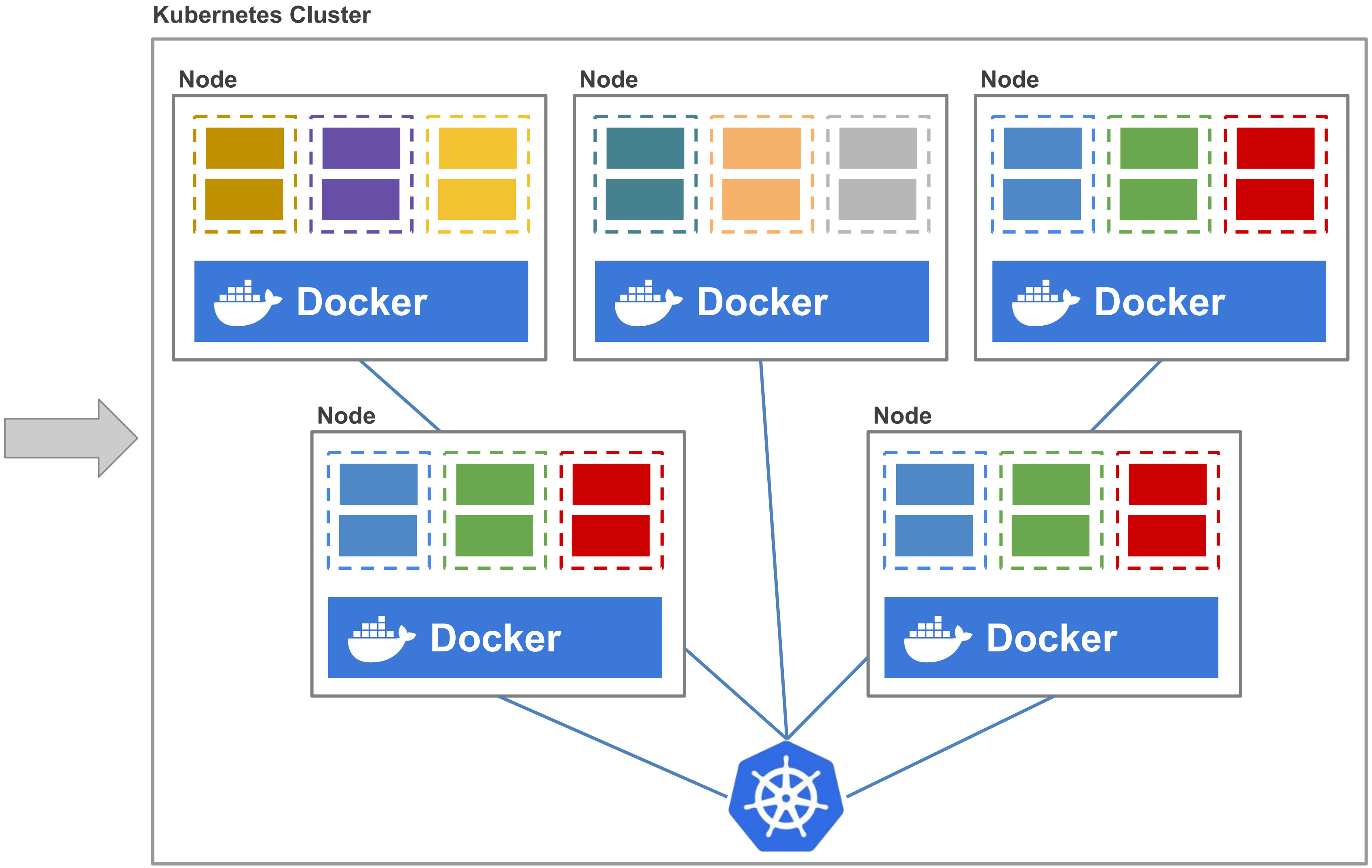
# Outline

1. Recap
2. Motivation
3. **Introduction to Kubernetes**
4. Tutorial: Deploying a Kubernetes Cluster
5. Advantages of using Kubernetes

# Container vs Kubernetes Deployment



**Container Deployment**



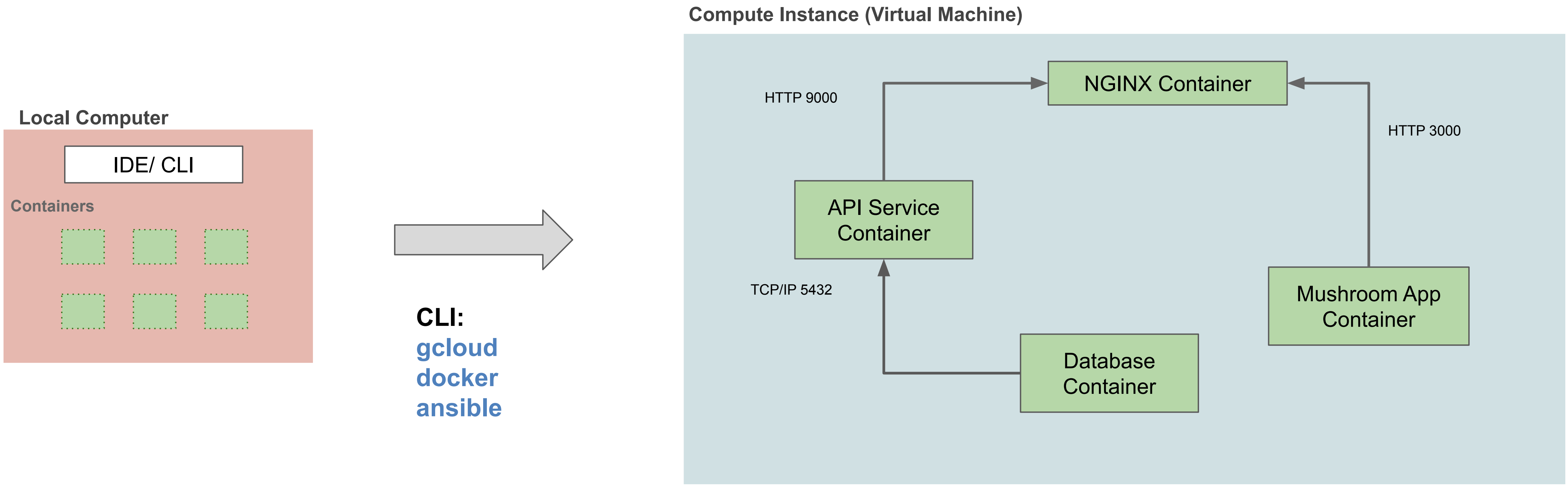
**Kubernetes Deployment**

# Why Kubernetes?

- **Automating** and **Management** of Microservices
- **Bridging** Application Deployment & Deployment (Dev + Ops)
- **Standardizing** Cloud Deployments
- Daily **Management** of Applications

# How do we build with Kubernetes?

## Remember the Mushroom App Architecture:



Local Computer

IDE/ CLI

Containers

**CLI:**
**gcloud**
**docker**
**ansible**

Compute Instance (Virtual Machine)

NGINX Container

HTTP 9000

HTTP 3000

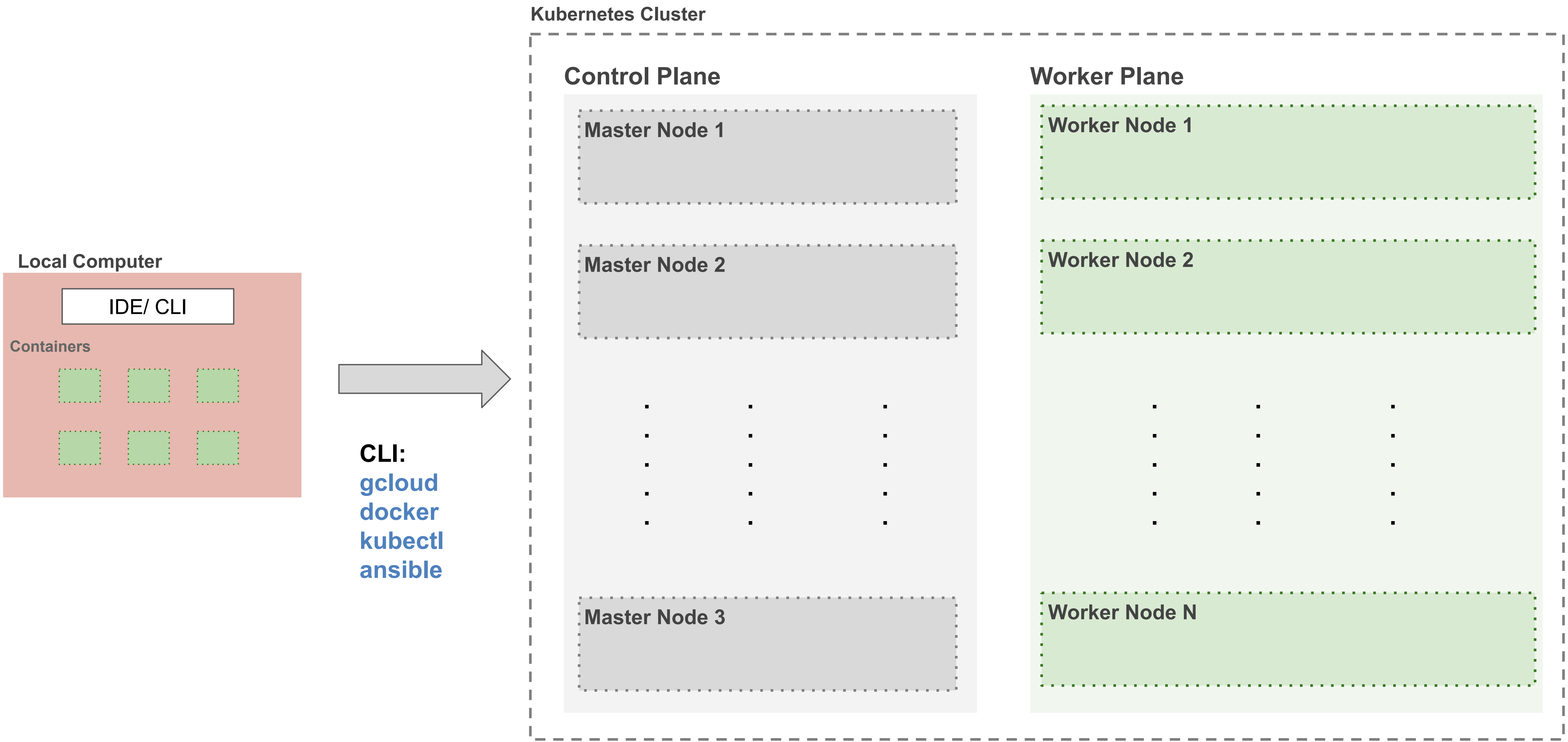
API Service Container

TCP/IP 5432

Mushroom App Container

Database Container

# K8s Components & Architecture



Local Computer

IDE/ CLI

Containers

**CLI:**
gcloud
docker
kubectl
ansible

Kubernetes Cluster

Control Plane

Master Node 1

Master Node 2

Master Node 3

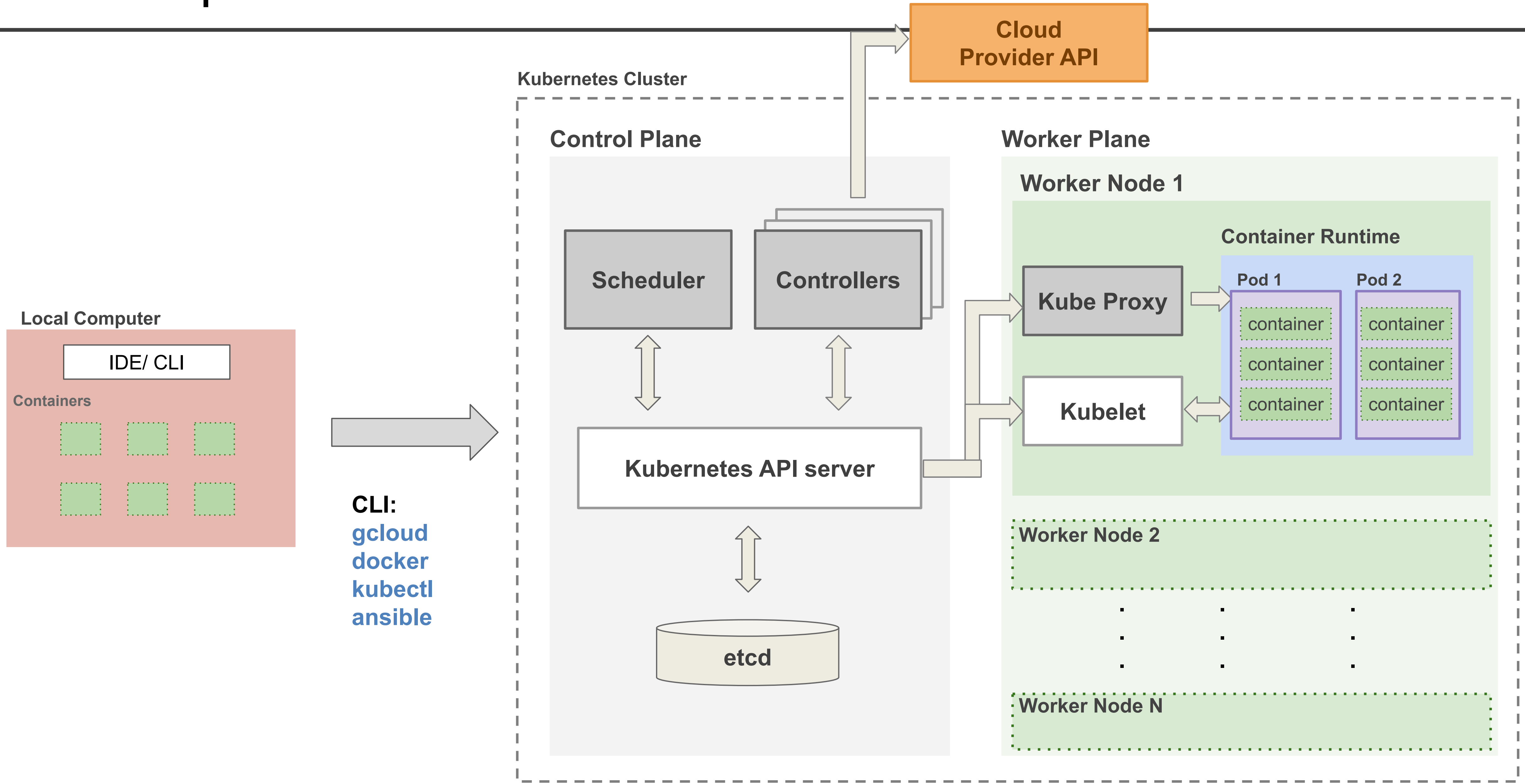
Worker Plane

Worker Node 1

Worker Node 2

Worker Node N

# K8s Components & Architecture



Local Computer

IDE/ CLI

Containers

**CLI:**
gcloud
docker
kubectl
ansible

Cloud Provider API

Kubernetes Cluster

Control Plane

Scheduler

Controllers

Kubernetes API server

etcd

Worker Plane

Worker Node 1

Container Runtime

Kube Proxy

Kubelet

Pod 1

container

container

container

Pod 2

container

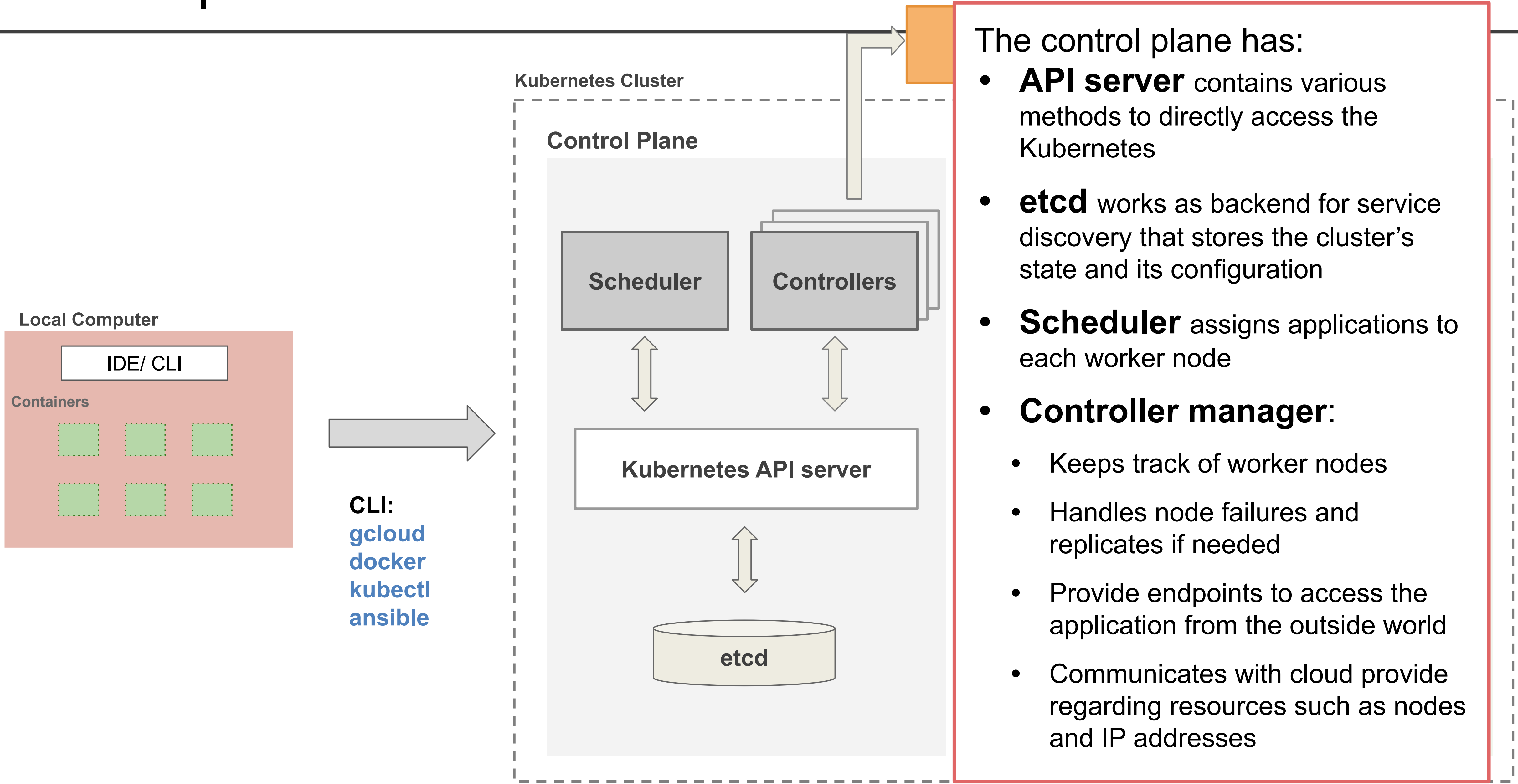
container

container

Worker Node 2

Worker Node N

# K8s Components & Architecture



CLI:
gcloud
docker
kubectl
ansible

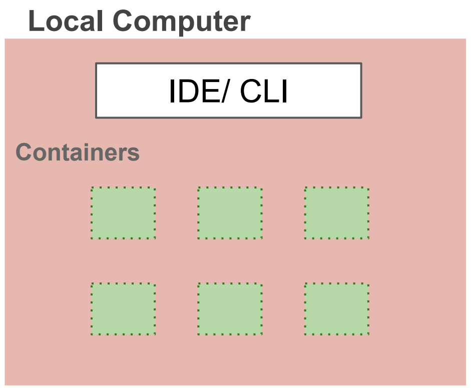
The control plane has:

- **API server** contains various methods to directly access the Kubernetes
- **etcd** works as backend for service discovery that stores the cluster's state and its configuration
- **Scheduler** assigns applications to each worker node
- **Controller manager**:
  - Keeps track of worker nodes
  - Handles node failures and replicates if needed
  - Provide endpoints to access the application from the outside world
  - Communicates with cloud provide regarding resources such as nodes and IP addresses

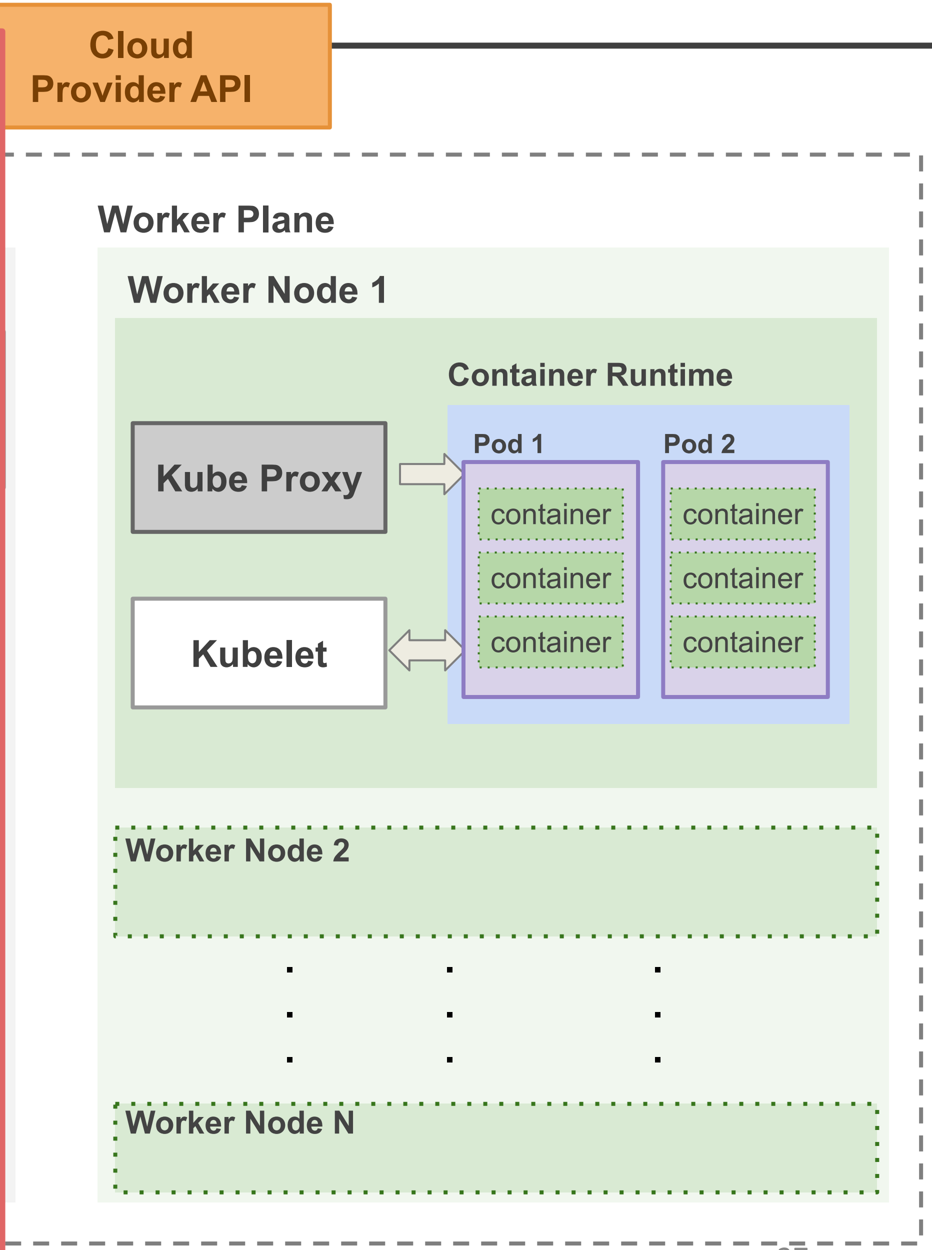
# K8s Components & Architecture

The worker node consists of:

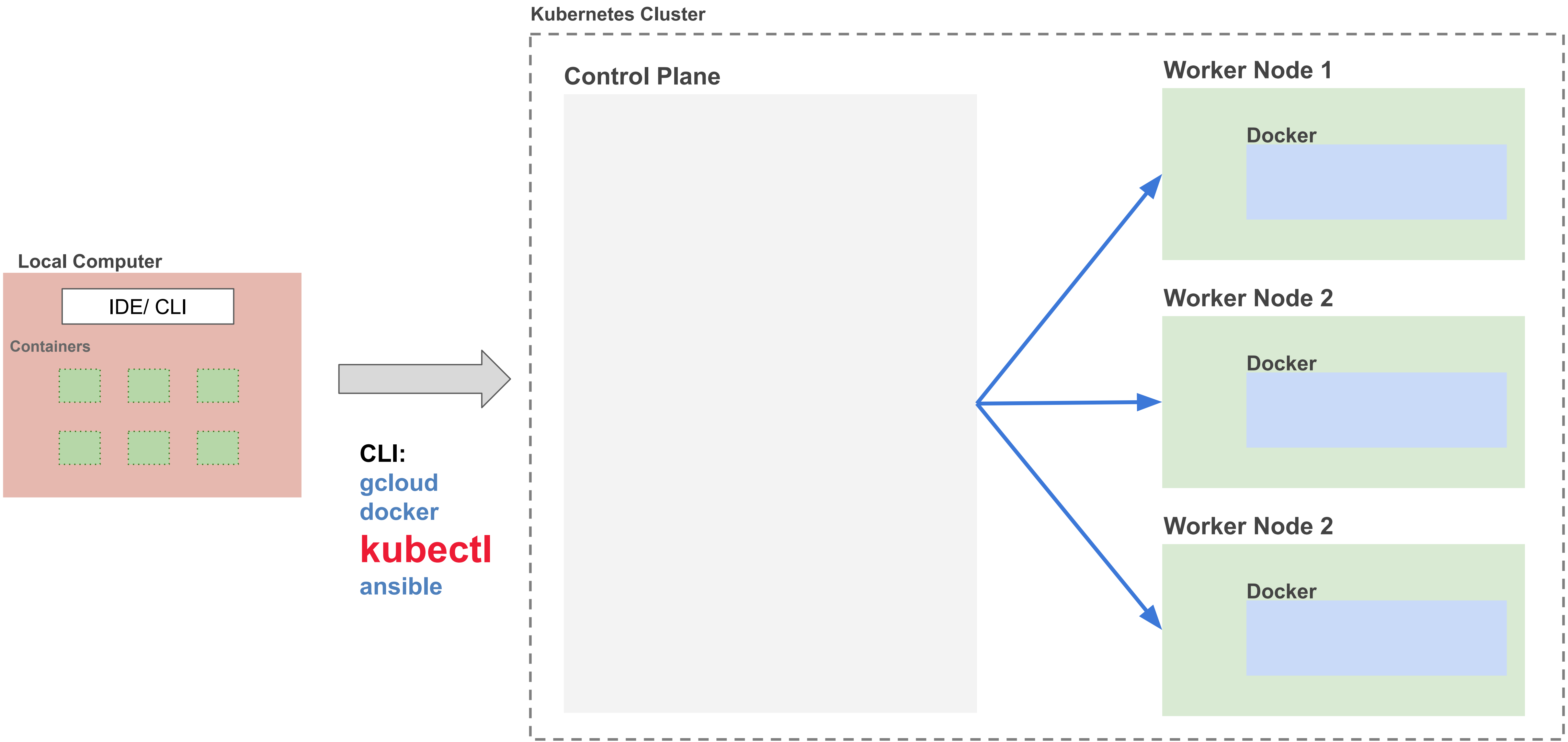
- **Kubelet** talks to the API server and manages containers on its node
- **Kube Proxy** load-balances network traffic between application components and the outside world
- **Container Runtime**: In our case this will be Docker. The runtime host Pods which run container instances



**CLI:**
gclou
dock
kube
ansil



# How do we build with Kubernetes?



# Kubernetes Summary

- **Abstracting** Infrastructure
- **Standardize** Application Deployment
- Deploy Applications **Declaratively**
- Daily **Management** of Applications

# Outline

1. Recap
2. Motivation
3. Introduction to Kubernetes
4. **Tutorial: Deploying a Kubernetes Cluster**
5. Advantages of using Kubernetes

# Tutorial: Deploying a Kubernetes Cluster

**Deploying a Kubernetes Cluster**

**https://github.com/dlops-io/mushroom-app-v3#create-kubernetes-cluster-tutorial**

# Create Kubernetes Cluster

To create a Kubernetes cluster

- You must first install ***gcloud*** which is the GCPs command-line tool
- You create and delete clusters using ***gcloud***

Example:

**Create a 2 node Kubernetes Cluster**

```
gcloud container clusters create test-cluster --num-nodes 2 --zone us-east1-c
```

```
Creating cluster test-cluster in us-east1-c...::
```

# Create Kubernetes Cluster

### Create a 2 node Kubernetes Cluster

```
gcloud container clusters create test-cluster --num-nodes 2 --zone us-east1-c
```

```
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/...
kubeconfig entry generated for test-cluster.
NAME          LOCATION    MASTER_VERSION  MASTER_IP      MACHINE_TYPE  NODE_VERSION    NUM_NODES  STATUS
test-cluster  us-east1-c  1.20.9-gke.701  34.73.126.138  e2-medium     1.20.9-gke.701  2          RUNNING
```

# Deploying to Kubernetes Cluster

To create a Kubernetes cluster and deploy app to it.

- You must first install ***kubectl*** which is the Kubernetes command-line tool
- You can manage all resources in Kubernetes using ***kubectl***

Examples:

**Get version of client**

```
kubectl version --client
```

```
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.1",
GitCommit:"632ed300f2c34f6d6d15ca4cef3d3c7073412212",
GitTreeState:"clean", BuildDate:"2021-08-19T15:45:37Z",
GoVersion:"go1.16.7", Compiler:"gc", Platform:"linux/amd64"}
```

**Get version of server**

```
kubectl version
```

```
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.1",
GitCommit:"632ed300f2c34f6d6d15ca4cef3d3c7073412212",
GitTreeState:"clean", BuildDate:"2021-08-19T15:45:37Z",
GoVersion:"go1.16.7", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you
specify the right host or port?
```

# Deploying to Kubernetes Cluster

## Examples:

### Get Kubernetes Cluster Information

```
kubectl get all
```

```
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.3.240.1   <none>        443/TCP   48m
```

### Get Kubernetes Component Status

```
kubectl get componentstatuses
```

```
NAME                 STATUS    MESSAGE             ERROR
scheduler            Healthy   ok
etcd-1               Healthy   {"health":"true"}
controller-manager   Healthy   ok
etcd-0               Healthy   {"health":"true"}
```

# Deploying to Kubernetes Cluster

## Examples:

### Get Kubernetes Cluster Nodes

```
kubectl get nodes
```

```
NAME                                        STATUS   ROLES    AGE   VERSION
gke-test-cluster-default-pool-2e9eafc9-kj0s   Ready    <none>   51m   v1.20.9-gke.701
gke-test-cluster-default-pool-2e9eafc9-t4pw   Ready    <none>   51m   v1.20.9-gke.701
```

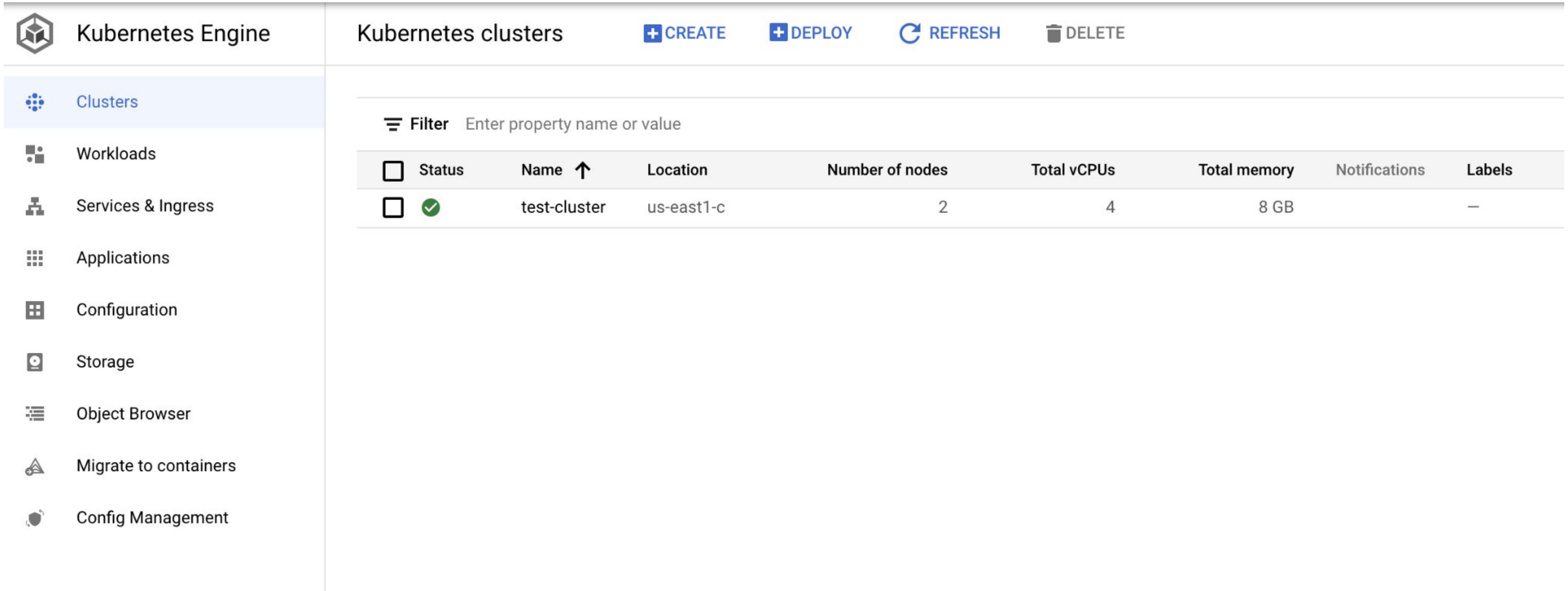
### Get Kubernetes Pods

```
kubectl get pods
```

```
No resources found in default namespace.
```

# Deploying to Kubernetes Cluster

You can view Kubernetes cluster details directly from GCP



| ☐ | Status | Name ↑ | Location | Number of nodes | Total vCPUs | Total memory | Notifications | Labels |
|---|---|---|---|---|---|---|---|---|
| ☐ | ✓ | test-cluster | us-east1-c | 2 | 4 | 8 GB | | — |

# Deploying to Kubernetes Cluster

## Examples:

### Deploy App to Kubernetes

```
kubectl apply -f deploy-k8s-tic-tac-toe.yml
```

```
deployment.apps/web created
service/web created
```

### Get Services

```
kubectl get services
```

```
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)        AGE
kubernetes   ClusterIP      10.3.240.1     <none>           443/TCP        29m
web          LoadBalancer   10.3.242.77    34.139.195.206   80:32088/TCP   3m51s
```

# Deploying to Kubernetes Cluster

### Deployment YAML

```yaml
---
apiVersion: apps/v1
kind: Deployment
spec:
  replicas: 2
  containers:
   - image: dlops/tic-tac-toe
     imagePullPolicy: IfNotPresent
     name: web
     ports:
      - containerPort: 8080
        protocol: TCP
```

**Deployment:**

- Decares what is in a pod and how many replicas
- Is in charge of keeping the pod running

### Service YAML

```yaml
---
apiVersion: v1
kind: Service
spec:
 ports:
 - port: 80
   protocol: TCP
   targetPort: 8080
 type: LoadBalancer
```

**Service:**

- Decares how traffic is routed to a pod or a multiple replicas.
- Service allows pods to die

# Deleting a Kubernetes Cluster

## Example:

**Delete Kubernetes Cluster called test-cluster**

```
gcloud container clusters delete test-cluster --zone us-east1-c
```

```
The following clusters will be deleted.
 - [test-cluster] in [us-east1-c]

Do you want to continue (Y/n)?  Y

Deleting cluster test-cluster...done.
Deleted [https://container.googleapis.com/v1/projects/.../zones/us-east1-c/clusters/test-cluster].
```

# Deploy Mushroom App to Kubernetes

**Deploying Mushroom App to Kubernetes Cluster**

**https://github.com/dlops-io/mushroom-app-v3#deployment-with-scaling-using-kubernetes**

# Outline

1. Recap
2. Motivation
3. Introduction to Kubernetes
4. Tutorial: Deploying a Kubernetes Cluster
5. **Advantages of using Kubernetes**

# Advantages of using Kubernetes

- **Self-Service** Deployment of Applications
- **Reduce Cost** by better Infrastructure Utilization
- **Automatically Adjusting** to varying loads
- Running Applications **Smoothly**
- Simplifying Application **Development**

# THANK YOU