Lecture 22-23: Scaling & Automation

Advanced Practical Data Science, MLOps

AC215

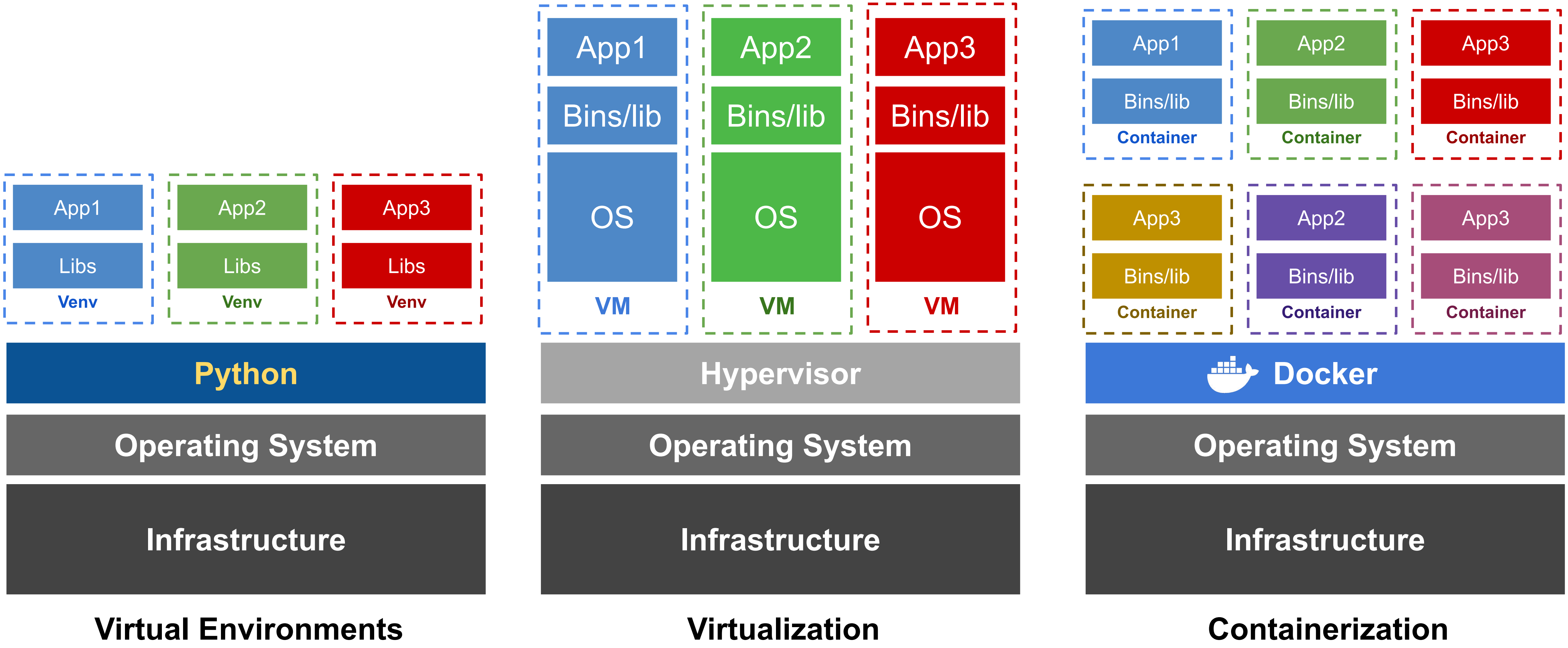
Pavlos Protopapas

Institute for Applied Computational Science, Harvard

Outline

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

Recap


App1
Libs
Venv
App2
Libs
Venv
App3
Libs
Venv
Python
Operating System
Infrastructure
Virtual Environments
App1
Bins/lib
OS
VM
App2
Bins/lib
OS
VM
App3
Bins/lib
OS
VM
Hypervisor
Operating System
Infrastructure
Virtualization
App1
Bins/lib
Container
App2
Bins/lib
Container
App3
Bins/lib
Container
App3
Bins/lib
Container
App2
Bins/lib
Container
App3
Bins/lib
Container
Docker
Operating System
Infrastructure
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

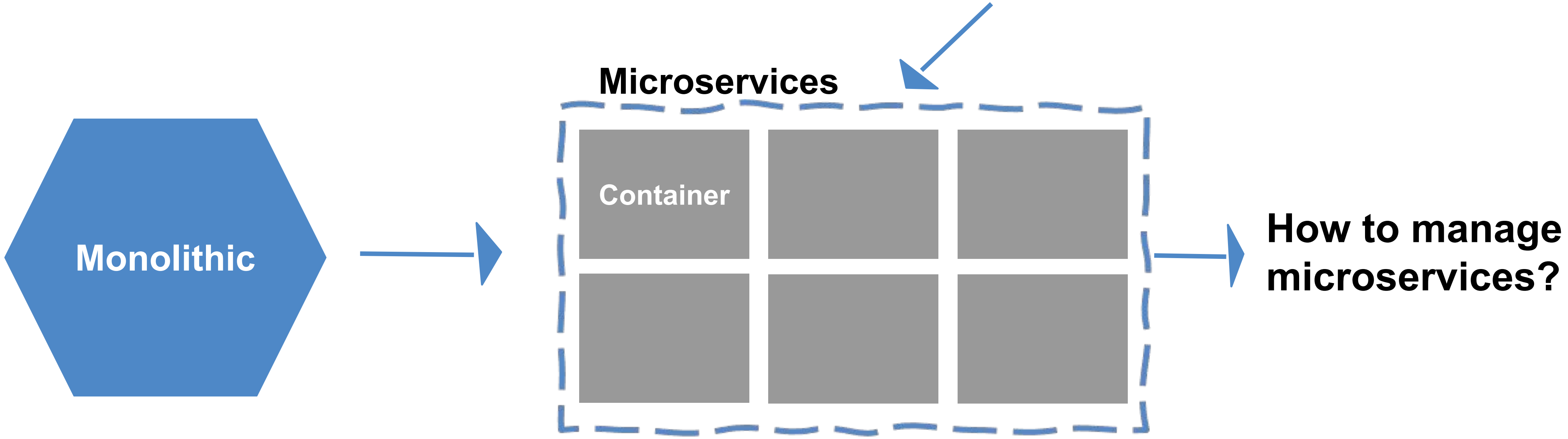
Pros: remove complexity
Cons: does not isolate from OS

Virtual Machines

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

Containers

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



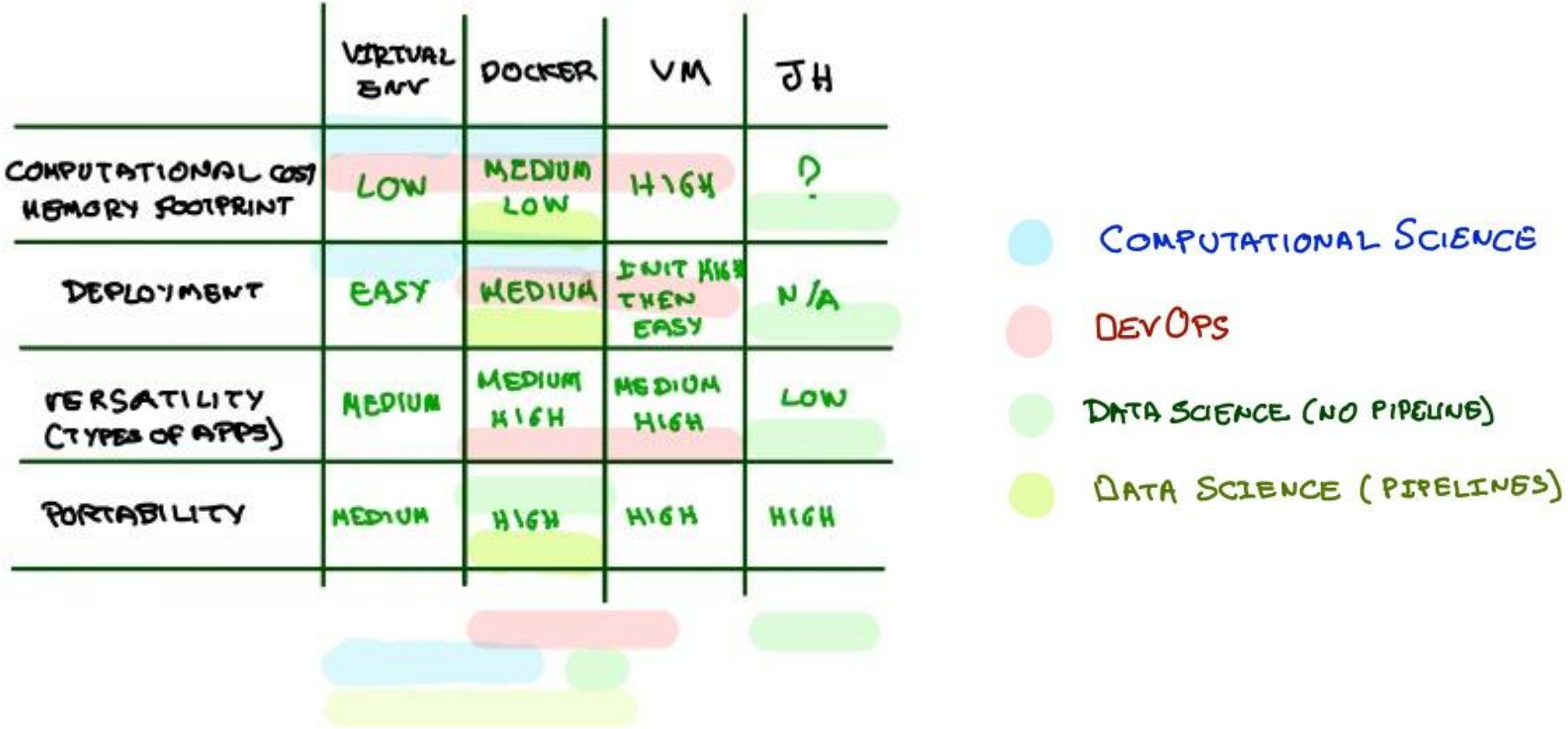
Monolithic

Microservices

Container

How to manage microservices?

Recap



	VIRTUAL ENV	DOCKER	VM	JH
COMPUTATIONAL COST MEMORY FOOTPRINT	LOW	MEDIUM LOW	HIGH	?
DEPLOYMENT	EASY	MEDIUM	INIT HIGH THEN EASY	N/A
VERSATILITY (TYPES OF APPS)	MEDIUM	MEDIUM HIGH	MEDIUM HIGH	LOW
PORTABILITY	MEDIUM	HIGH	HIGH	HIGH

● COMPUTATIONAL SCIENCE

● DEVOPS

● DATA SCIENCE (NO PIPELINE)

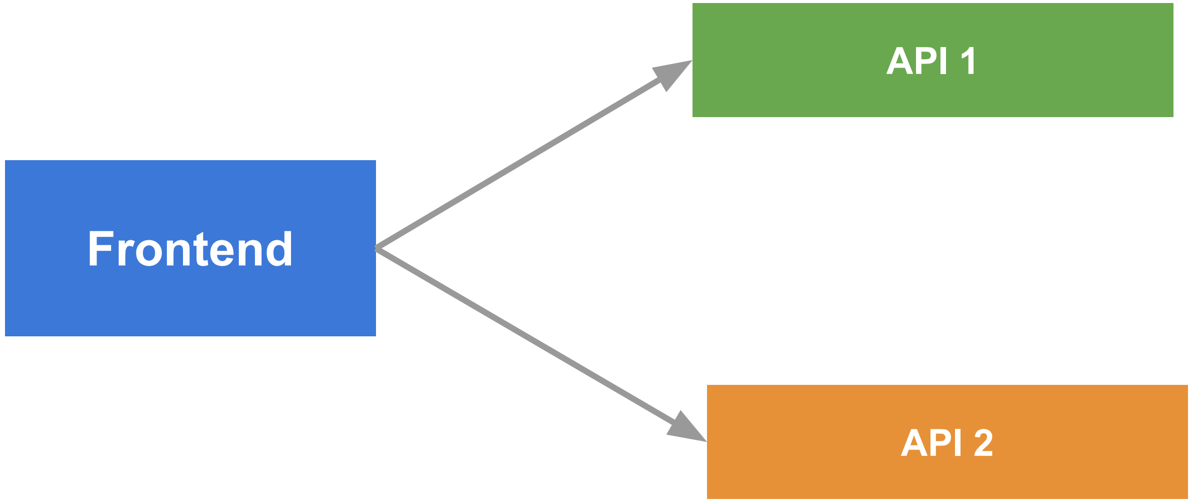
● DATA SCIENCE (PIPELINES)

Outline

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

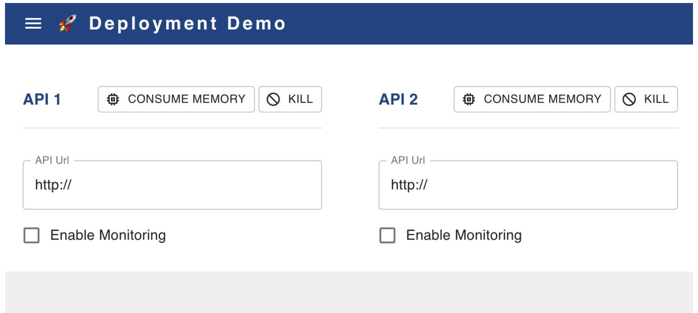
Motivation

Pavlos wants an app with 1 frontend & 2 backends



Motivation - 3 Containers in 1 VM

Shivas build and deploys the app with the following architecture

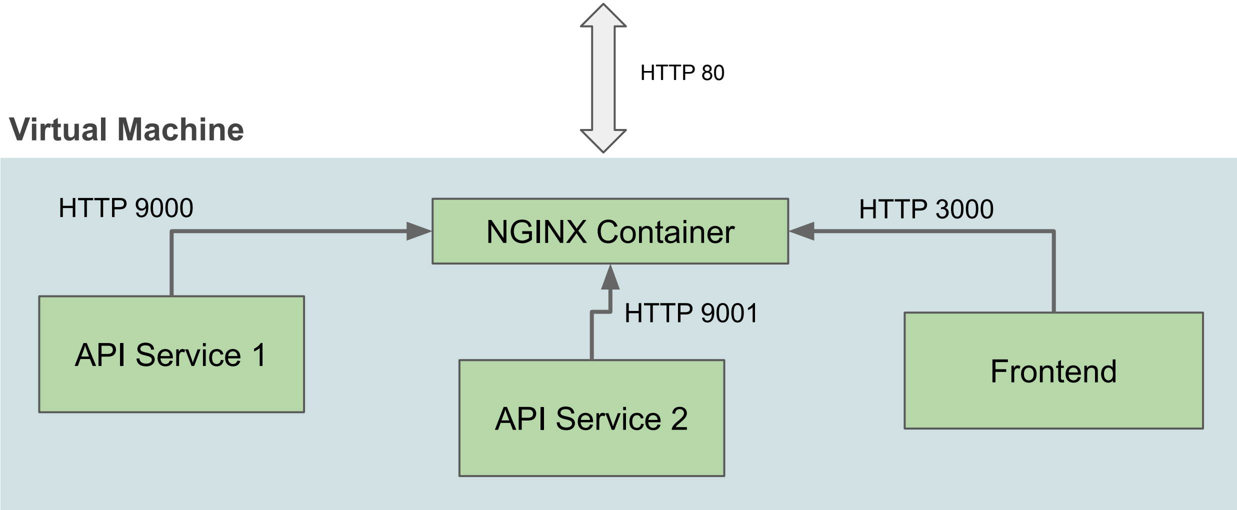


API Url
http://

☐ Enable Monitoring

API Url
http://

☐ Enable Monitoring



HTTP 80

Virtual Machine

HTTP 9000

NGINX Container

HTTP 3000

HTTP 9001

API Service 1

API Service 2

Frontend

Motivation - 3 Containers in 1 VM

Demo… [3 Containers in 1 VM]

Motivation - 3 Containers in 1 VM

Problems:

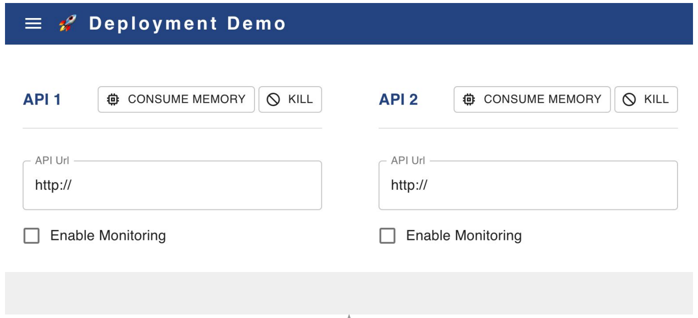
- When container crashes Pavlos has to call *support*
- Support SSHs into server and fix:
    - Memory reset with container restart
    - Startup a killed container

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



≡ 🚀 Deployment Demo

API 1 · CONSUME MEMORY · KILL

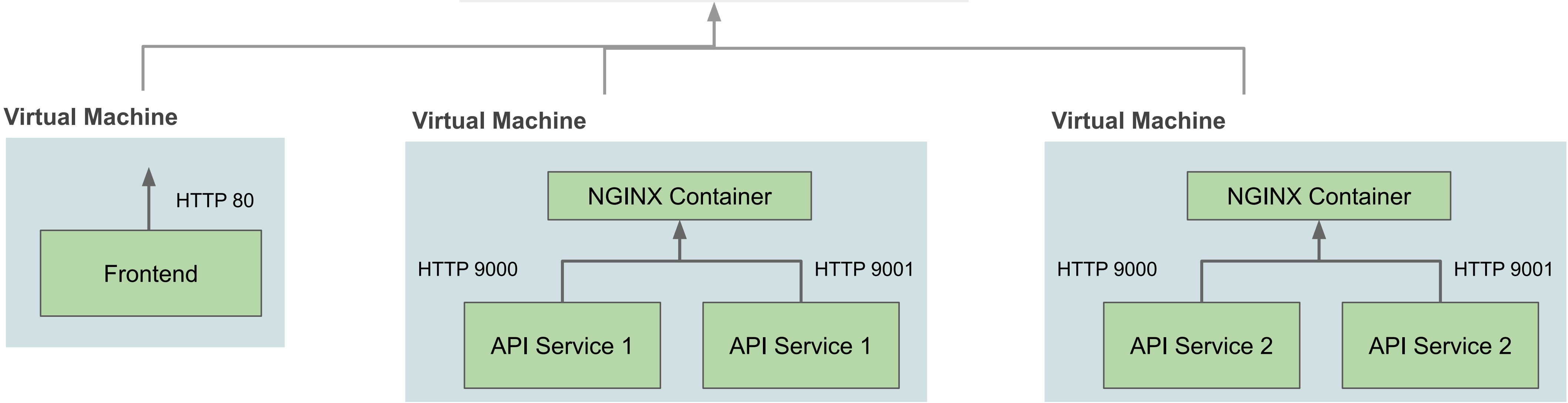
API Url: http://

□ Enable Monitoring

API 2 · CONSUME MEMORY · KILL

API Url: http://

□ Enable Monitoring



Virtual Machine

HTTP 80

Frontend

Virtual Machine

NGINX Container

HTTP 9000

HTTP 9001

API Service 1

API Service 1

Virtual Machine

NGINX Container

HTTP 9000

HTTP 9001

API Service 2

API Service 2

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

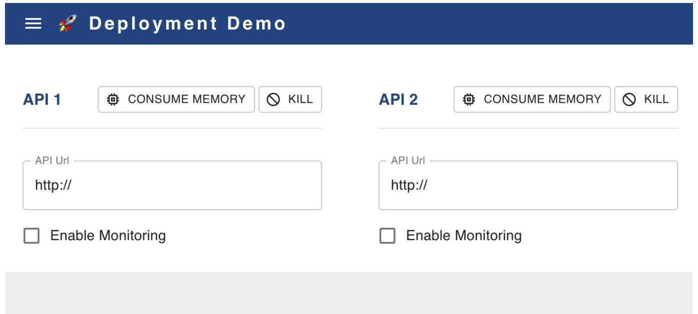
- K8s is an orchestration tool for **managing distributed containers** across a cluster of nodes (VMs).

- 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

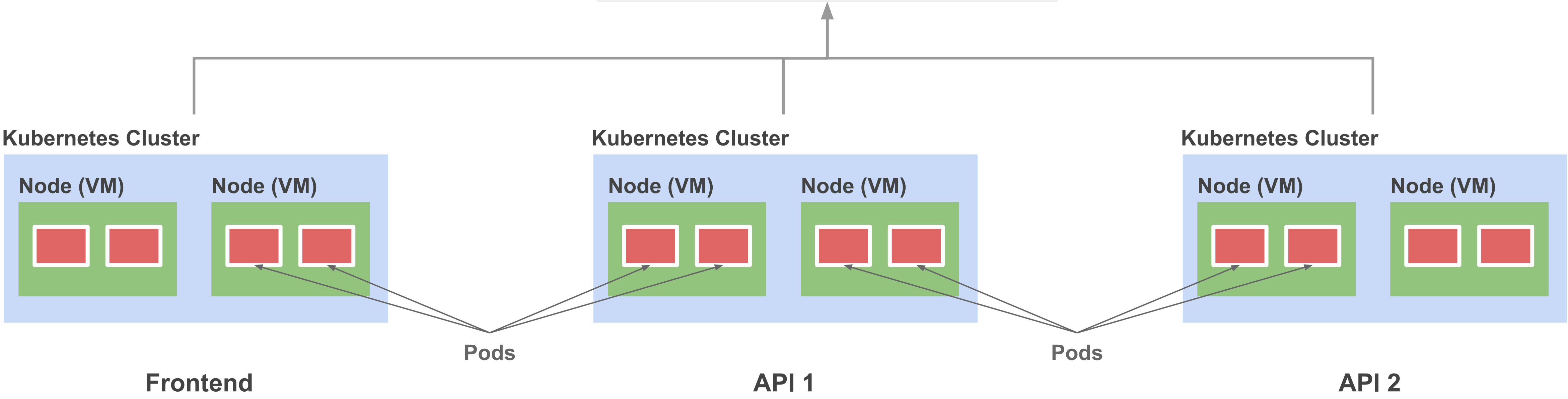


API Url: http://

□ Enable Monitoring

API Url: http://

□ Enable Monitoring



Kubernetes to the Rescue

Demo… [Kubernetes Cluster]

Kubernetes

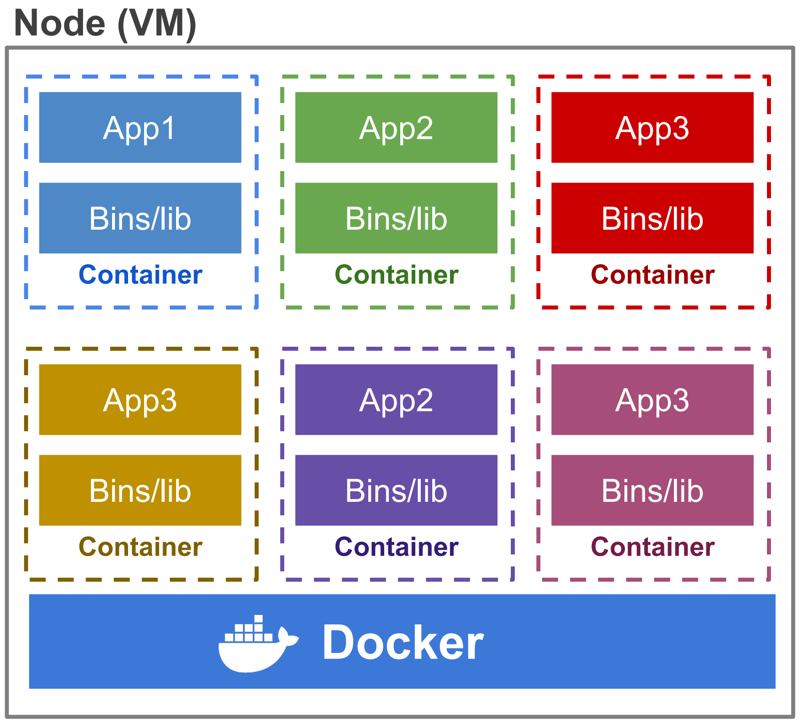
Pavlos requests on automation:

- ✓ Failovers
- ✓ Load balancing
- ✓ Scaling

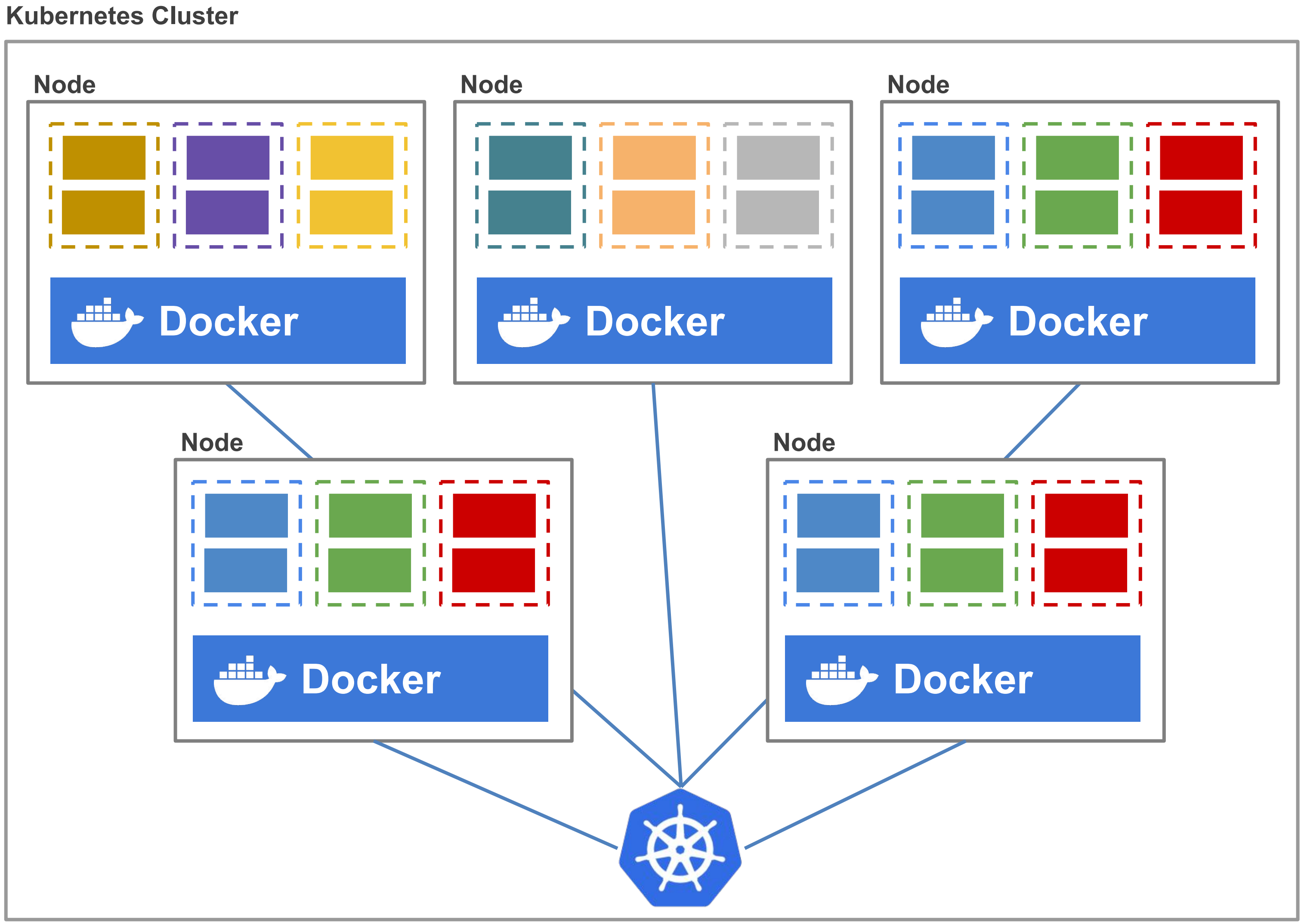
Outline

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

Container vs Kubernetes Deployment



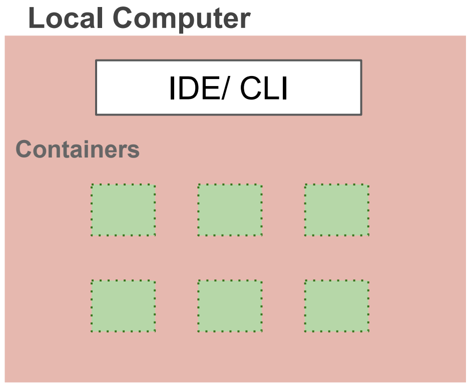
Container Deployment



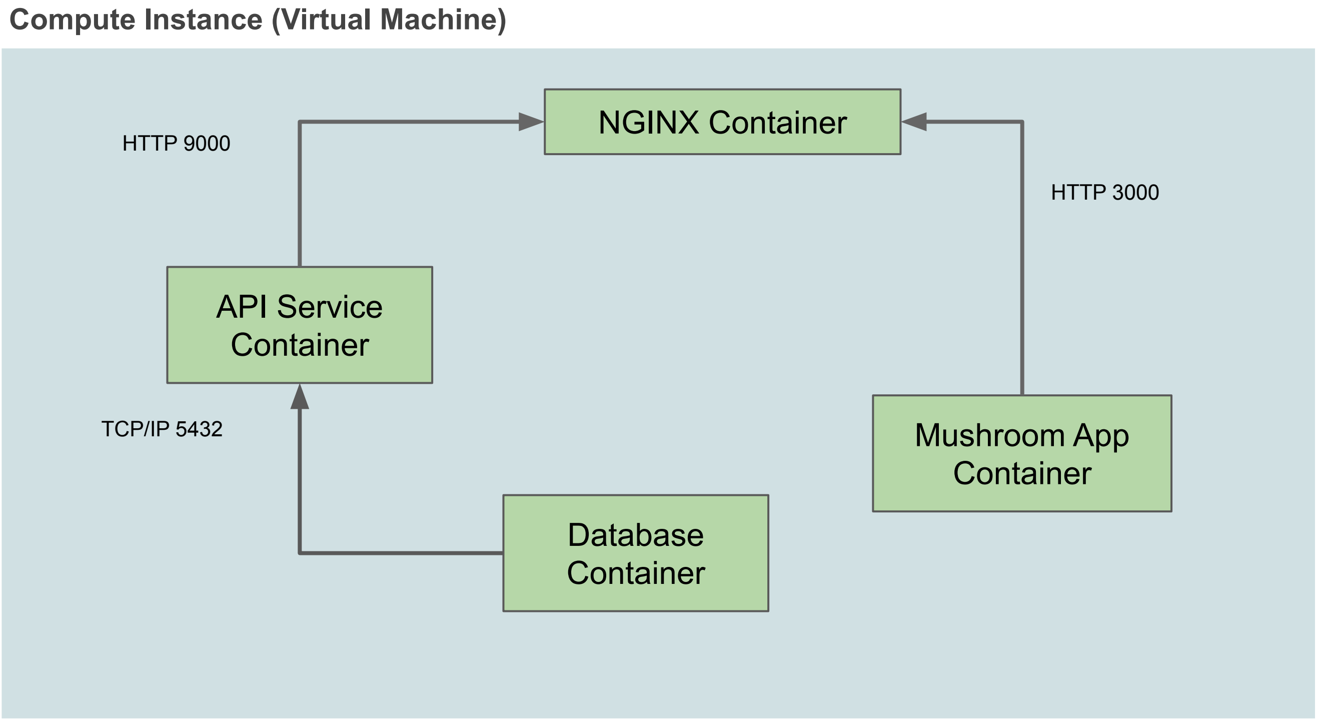
Kubernetes Deployment

How do we build with Kubernetes?

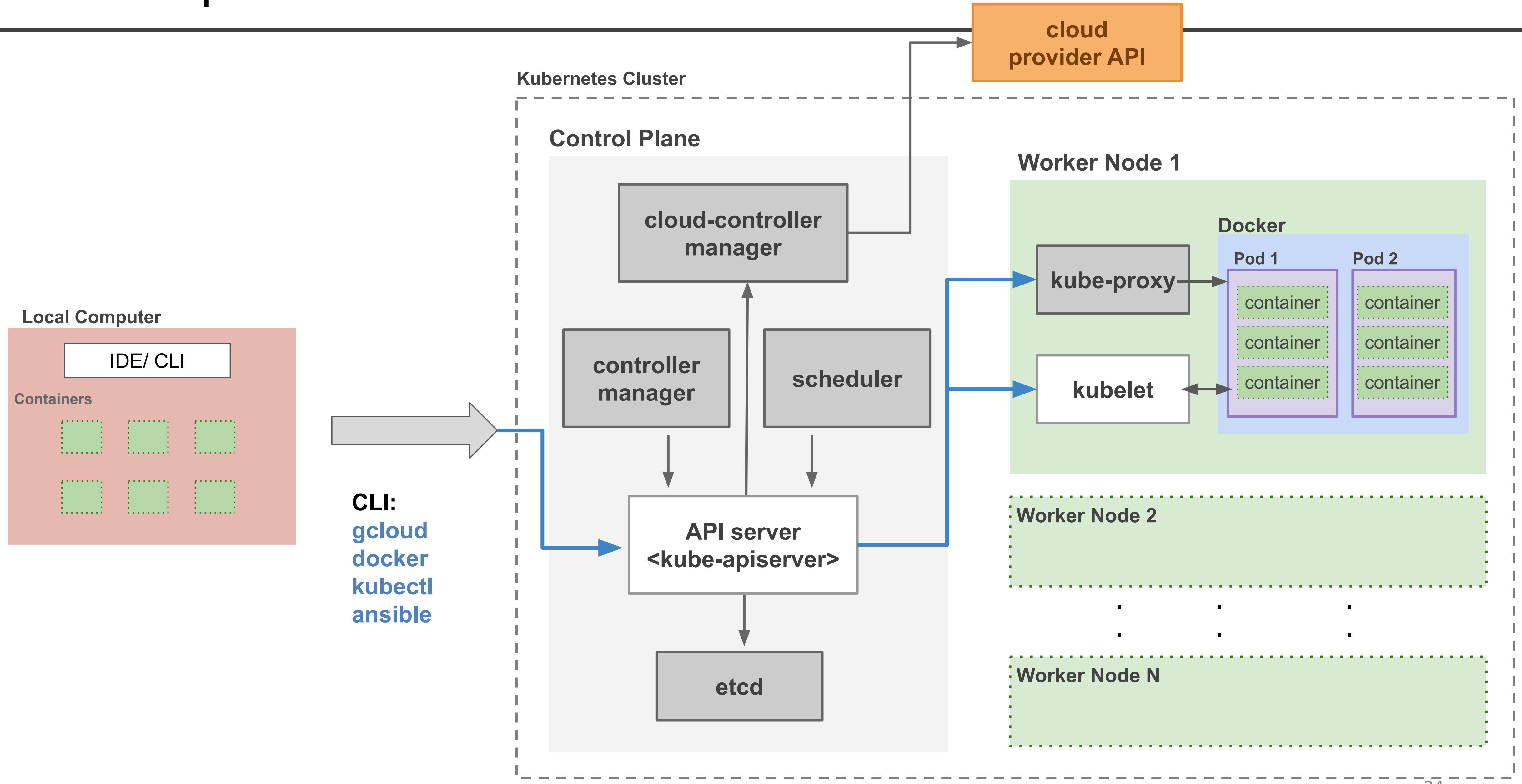
Remember the Mushroom App Architecture:



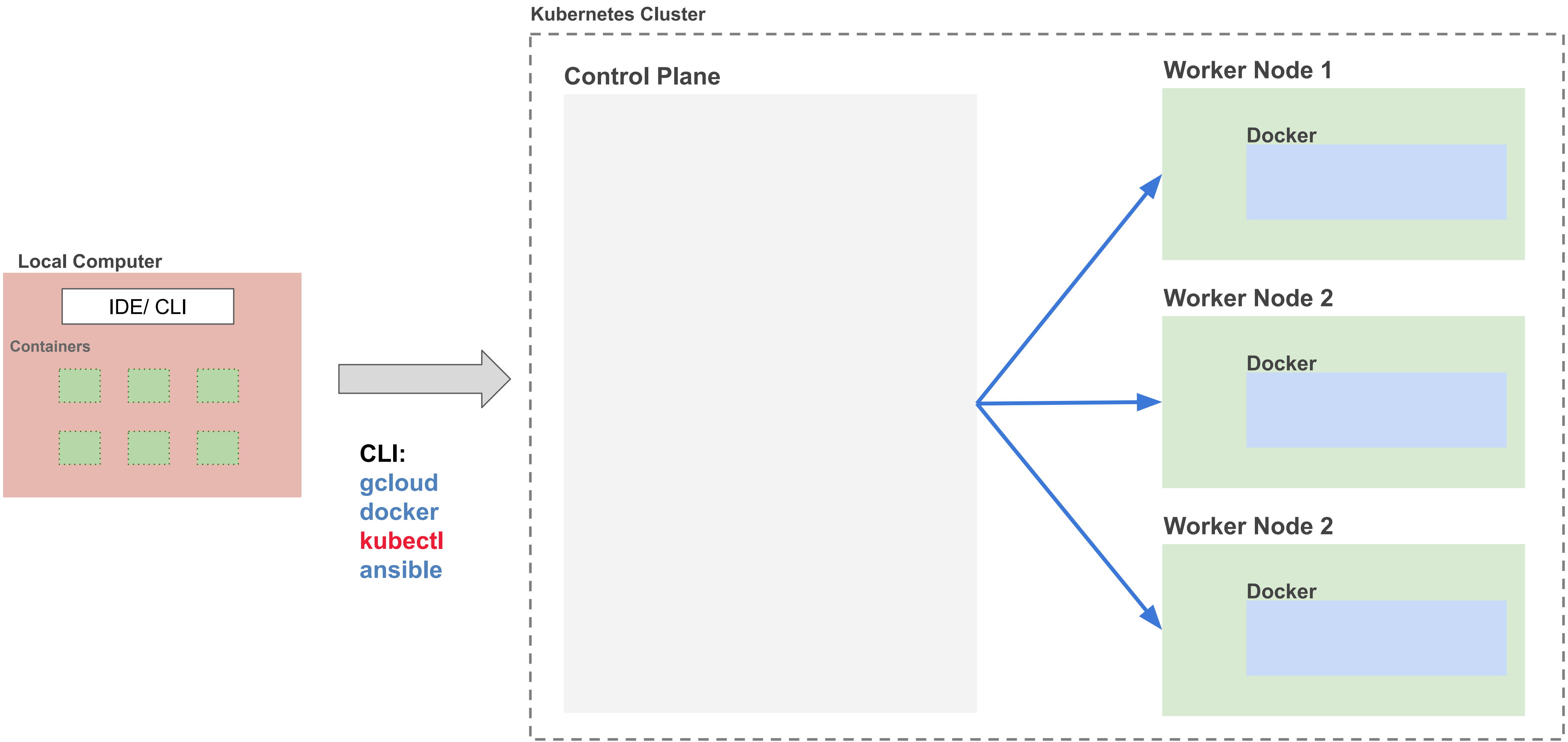
CLI:
gcloud
docker
ansible



K8s Components & Architecture

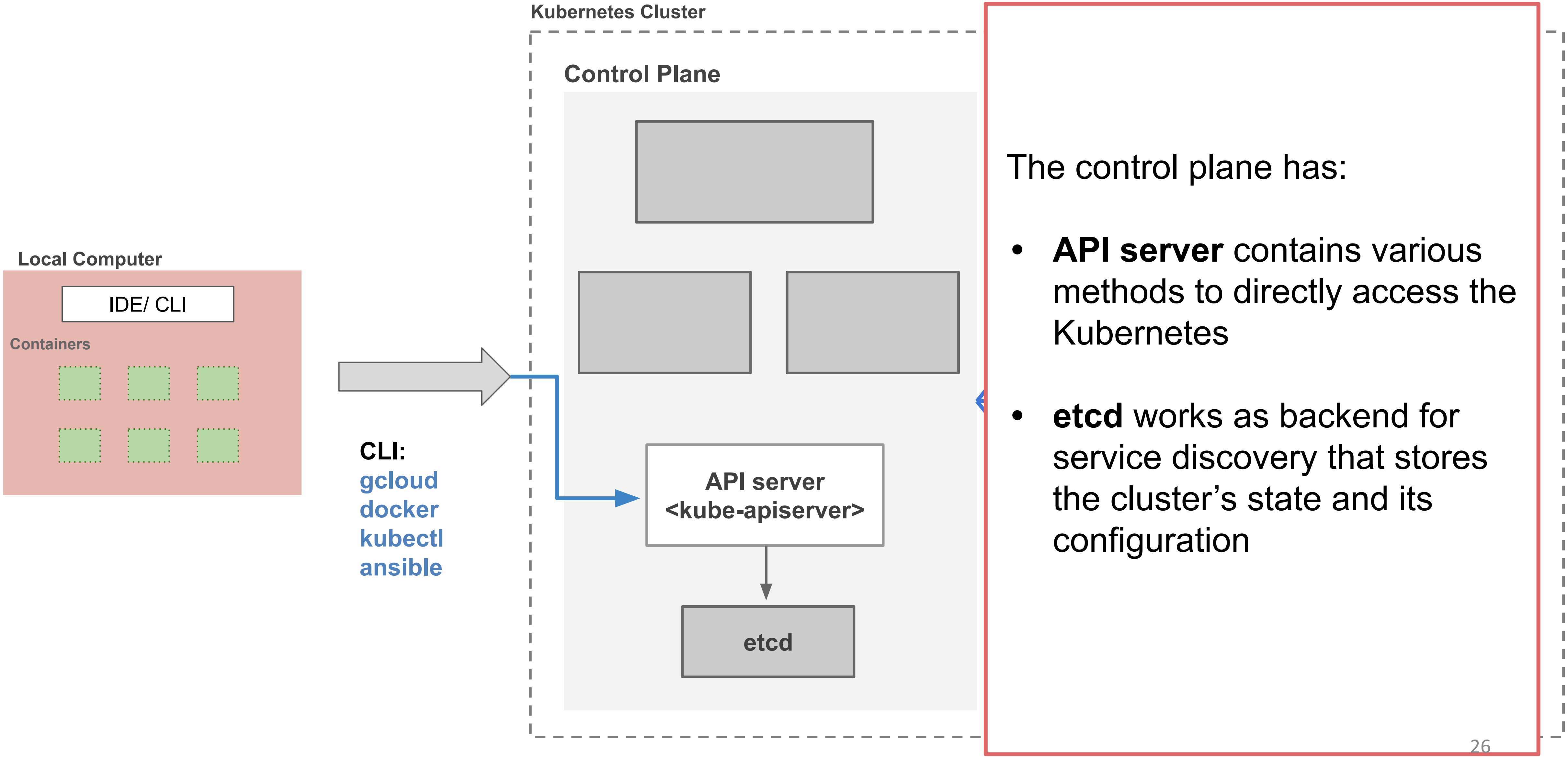


How do we build with Kubernetes?



CLI:
gcloud
docker
kubectl
ansible

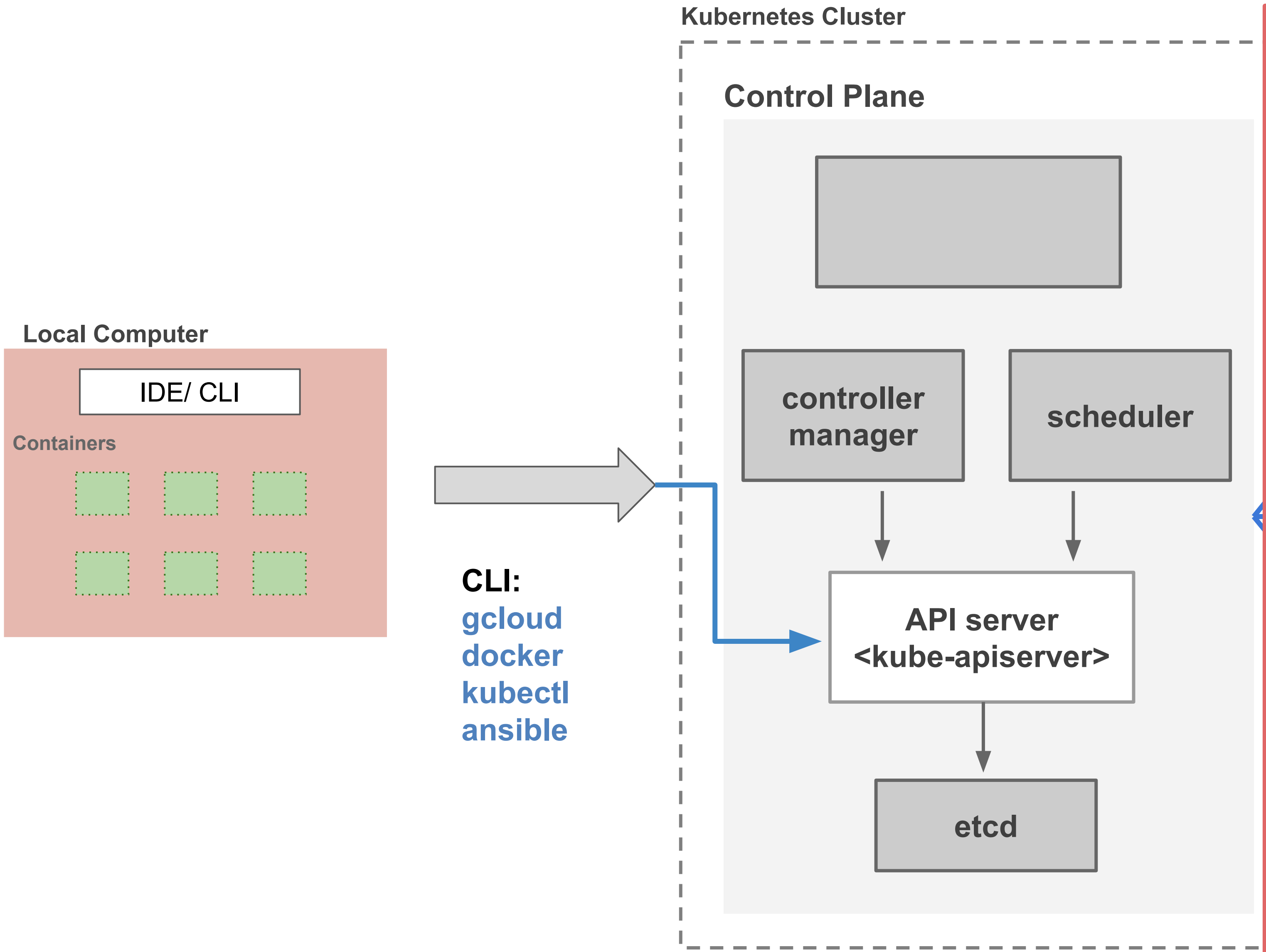
K8s Components & Architecture



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

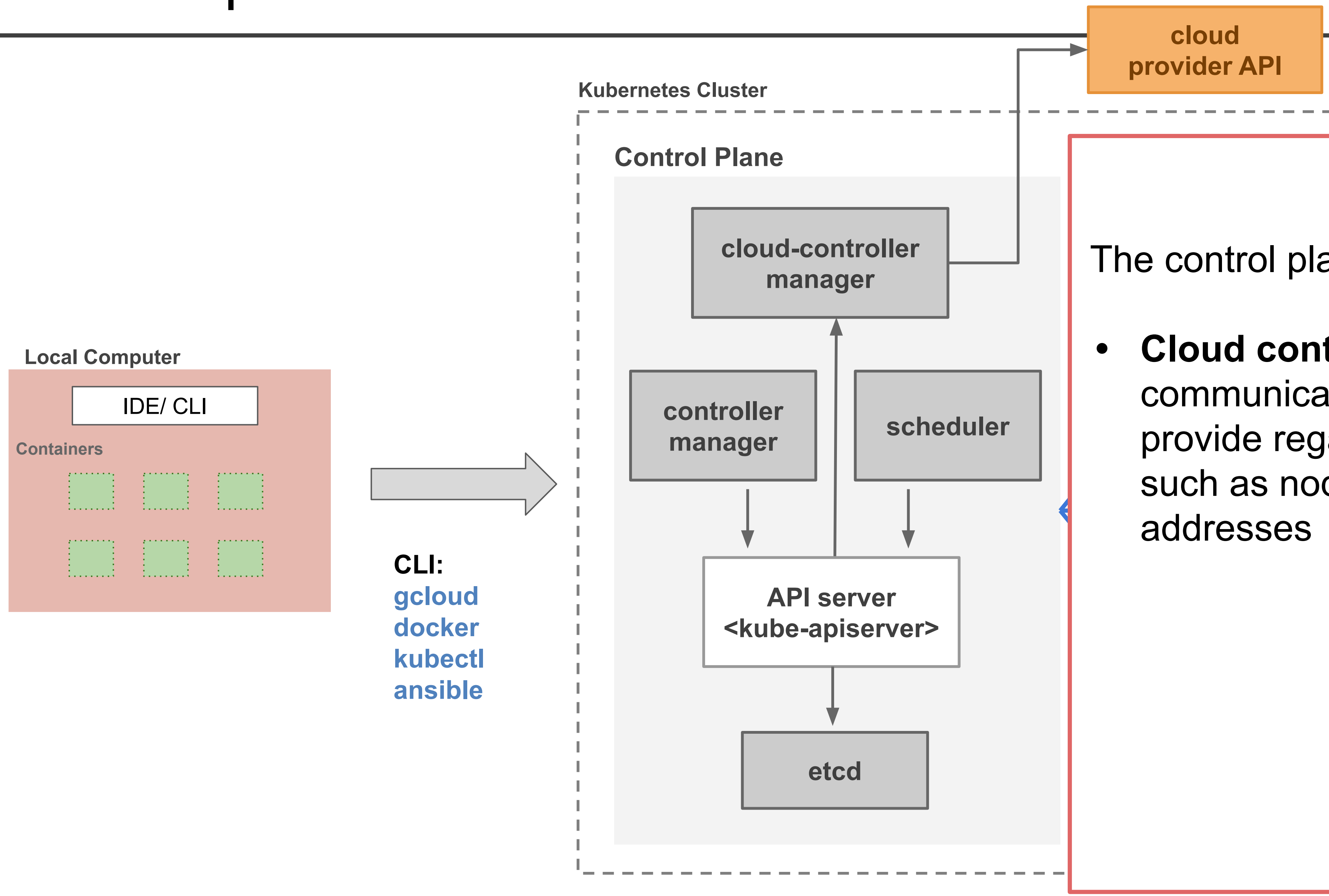
K8s Components & Architecture



The control plane also has:

- **Scheduler** assigns to each worker node an application
- **Controller manager:**
  - Keeps track of worker nodes
  - Handles node failures and replicates if needed
  - Provide endpoints to access the application from the outside world

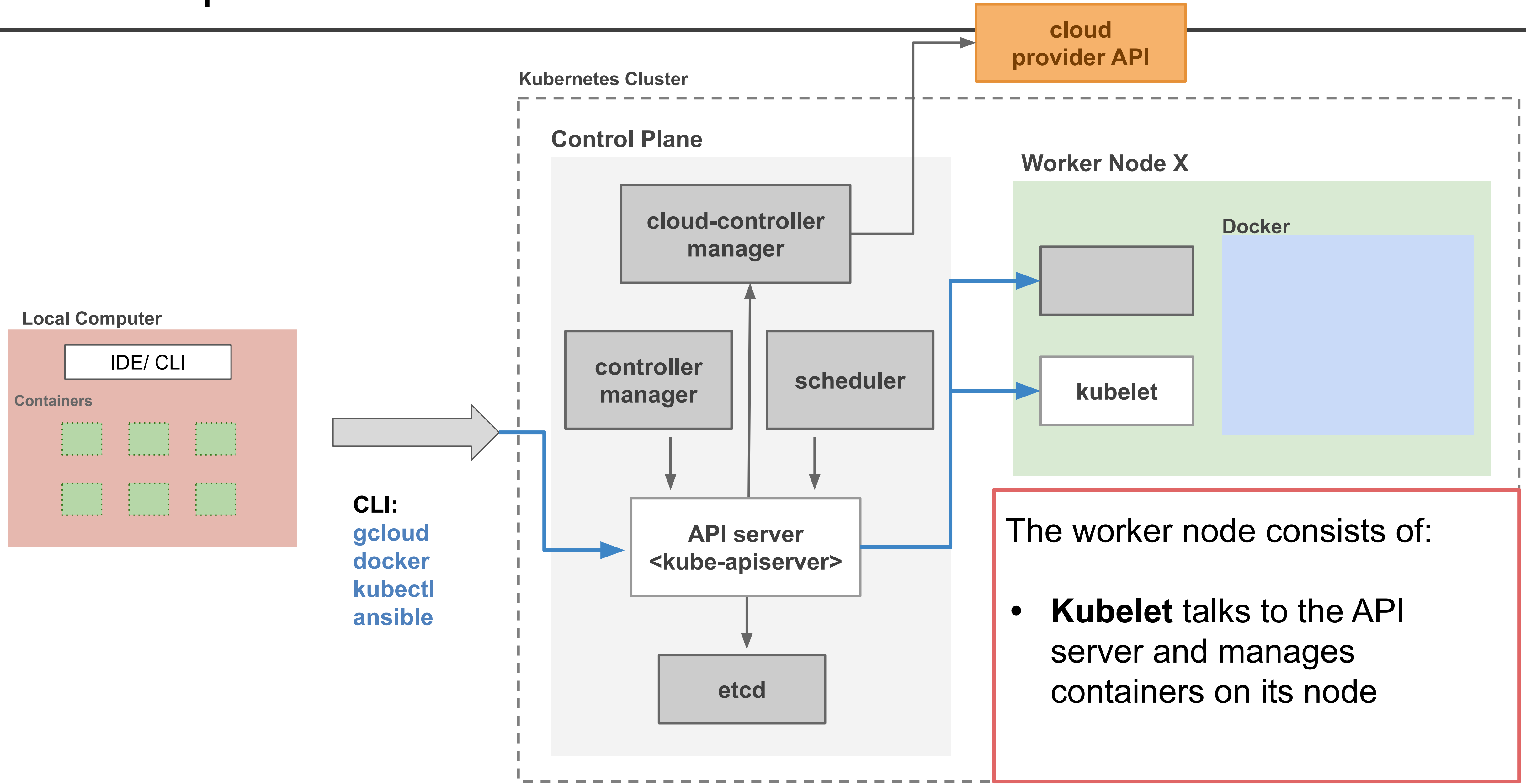
K8s Components & Architecture



The control plane also has:

- **Cloud controller** communicates with cloud provide regarding resources such as nodes and IP addresses

K8s Components & Architecture

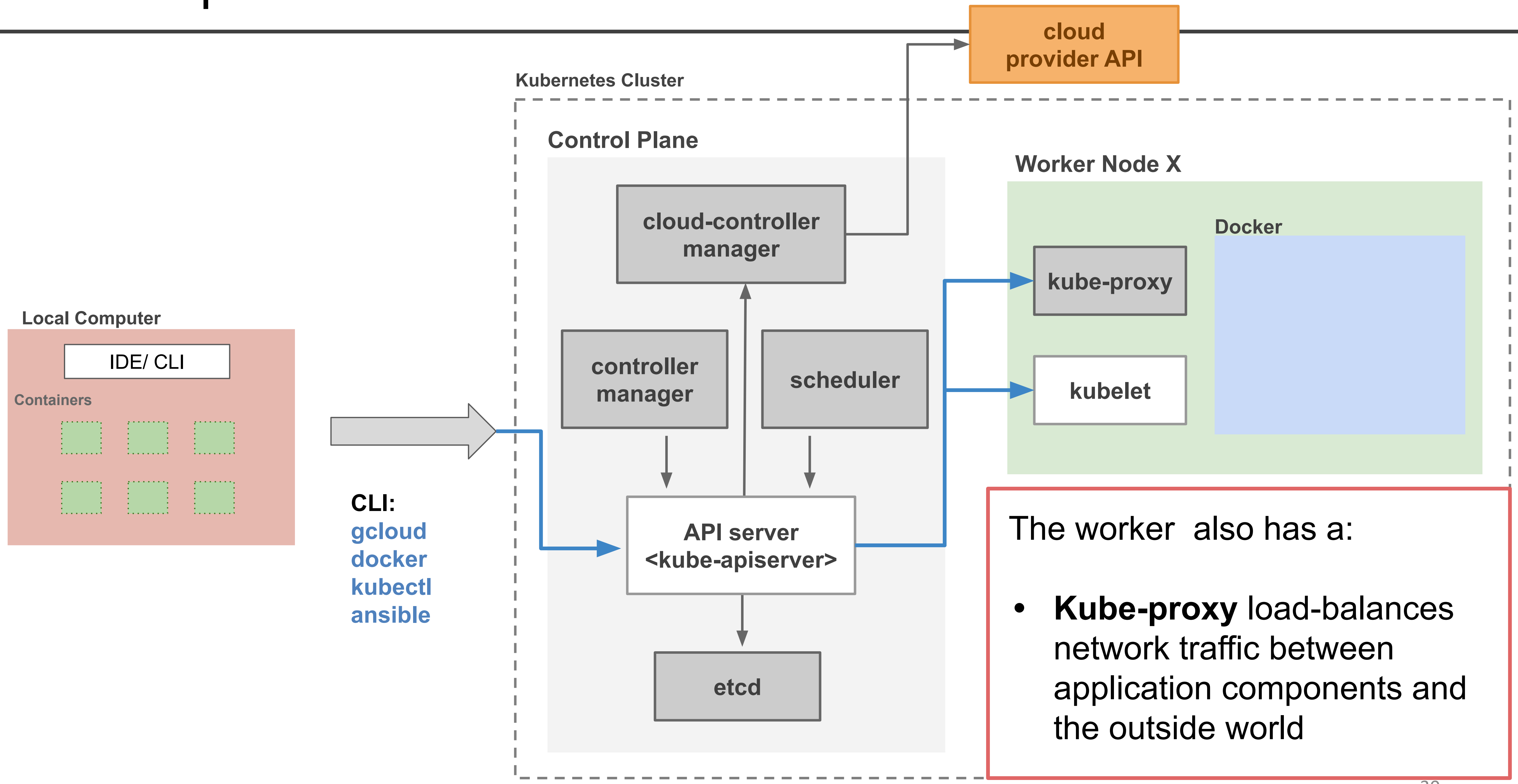


CLI:
gcloud
docker
kubectl
ansible

The worker node consists of:

- **Kubelet** talks to the API server and manages containers on its node

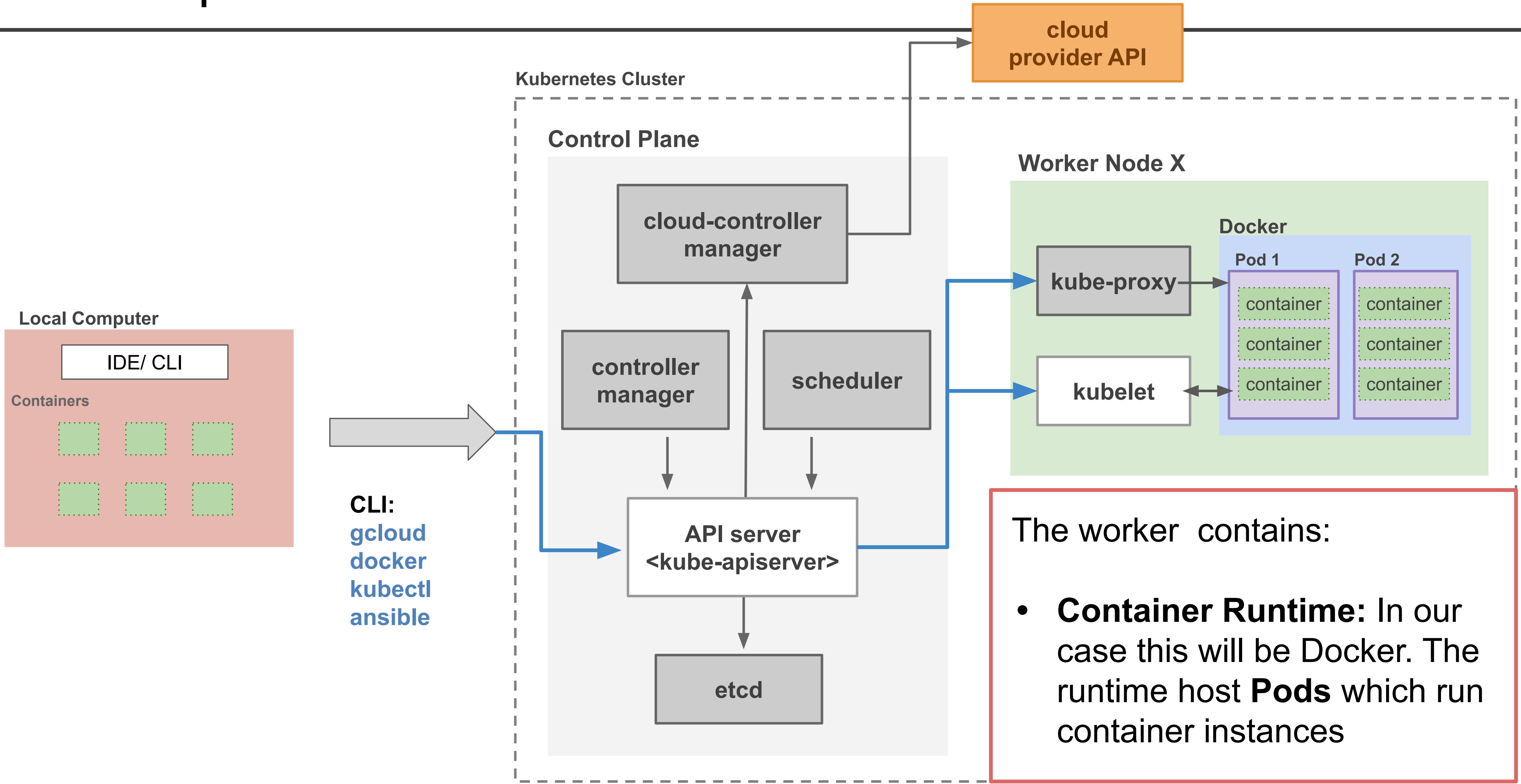
K8s Components & Architecture



The worker also has a:

- **Kube-proxy** load-balances network traffic between application components and the outside world

K8s Components & Architecture



CLI:
gcloud
docker
kubectl
ansible

The worker contains:

- **Container Runtime:** In our case this will be Docker. The runtime host **Pods** which run container instances

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]

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
```

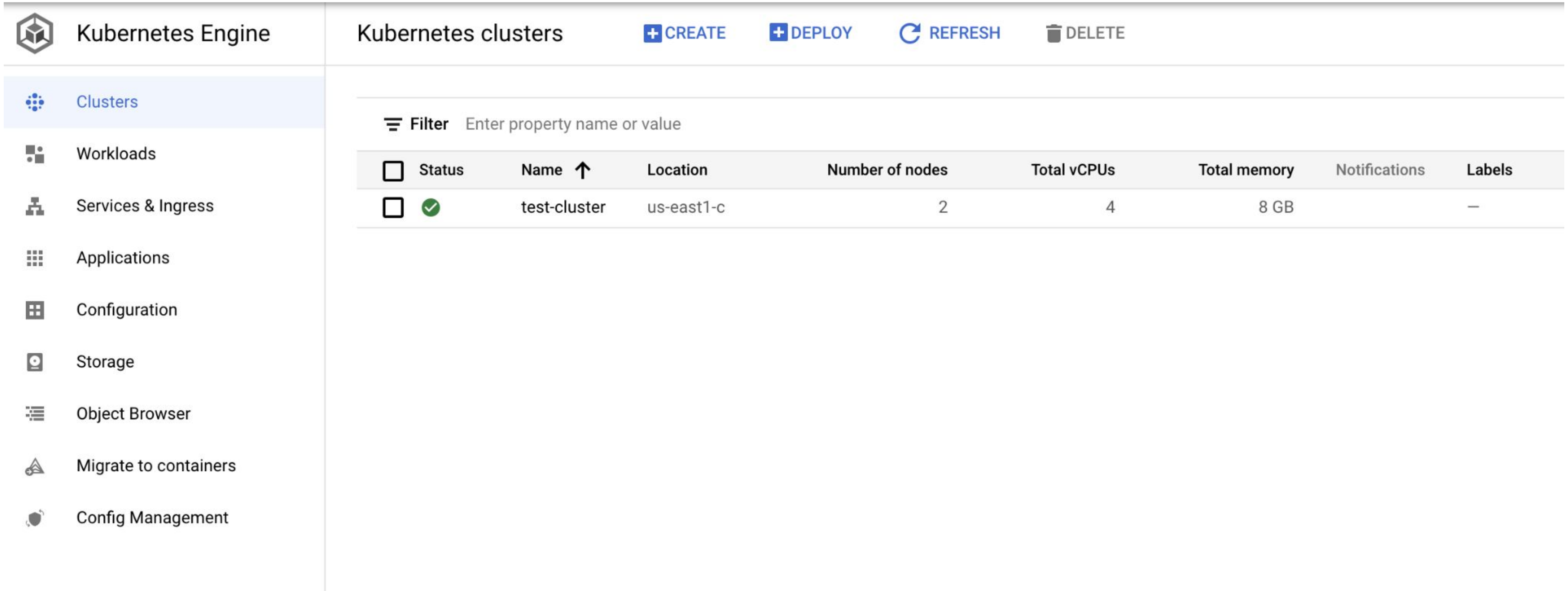
Get Kubernetes Pods

```
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

```
---
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

```
---
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

Outline

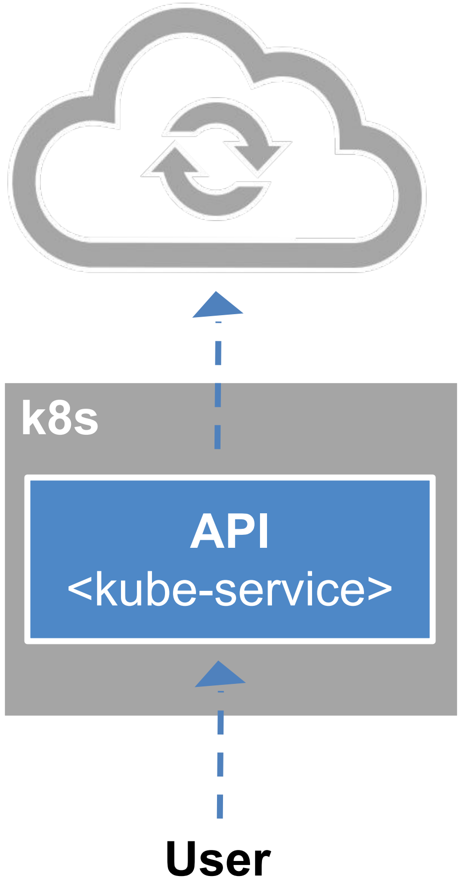
1. Recap
2. Motivation
3. Introduction to Kubernetes
4. Tutorial: Deploying a Kubernetes Cluster
5. **Advantages of using Kubernetes**

Advantages of using Kubernetes

There are many reasons why people come to use containers and container orchestration tools like Kubernetes:

1. **Velocity**
2. **Abstracting the infrastructure**
3. **Efficiency**
4. **Scaling** (software, hardware, and teams)

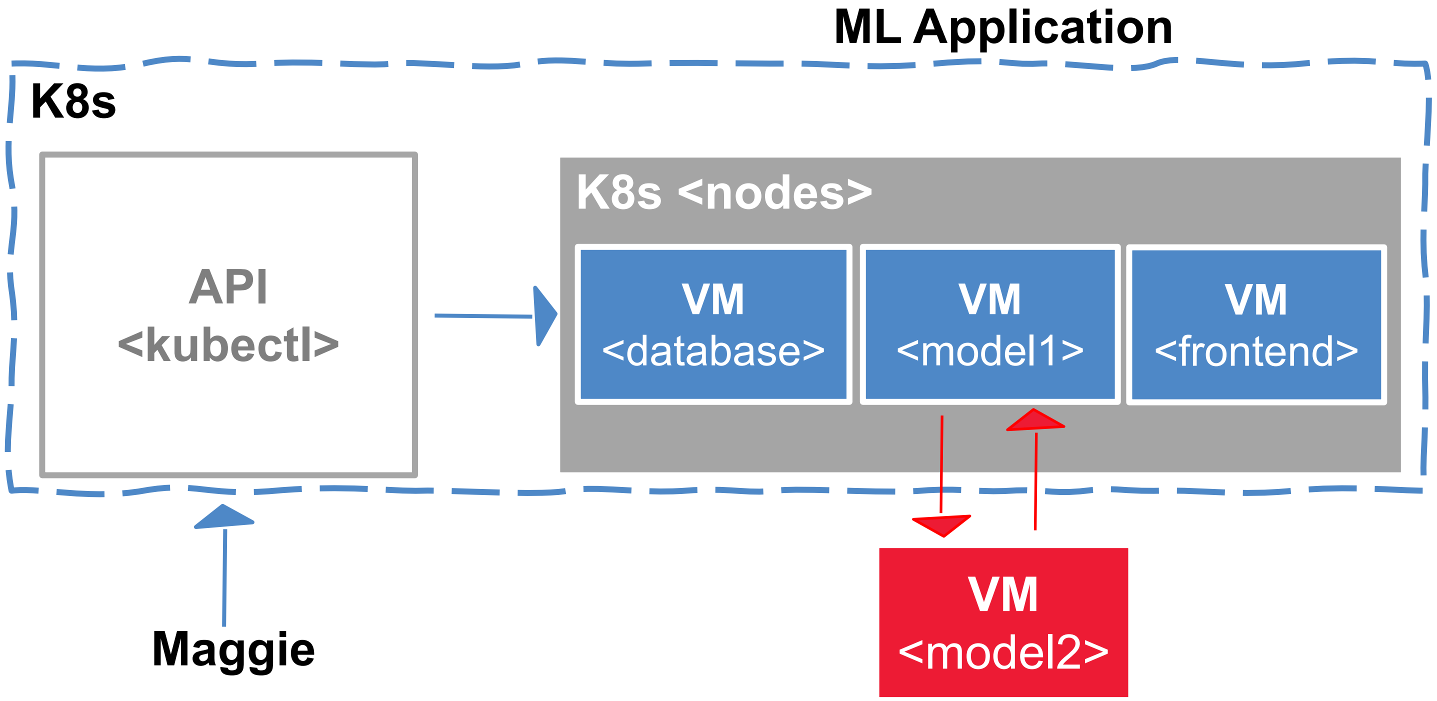
All these aspects relate to each other to speed up process that can reliably deploy software.



Advantages of using Kubernetes: **Velocity**

It is the speed with which you can respond to innovations developed by others

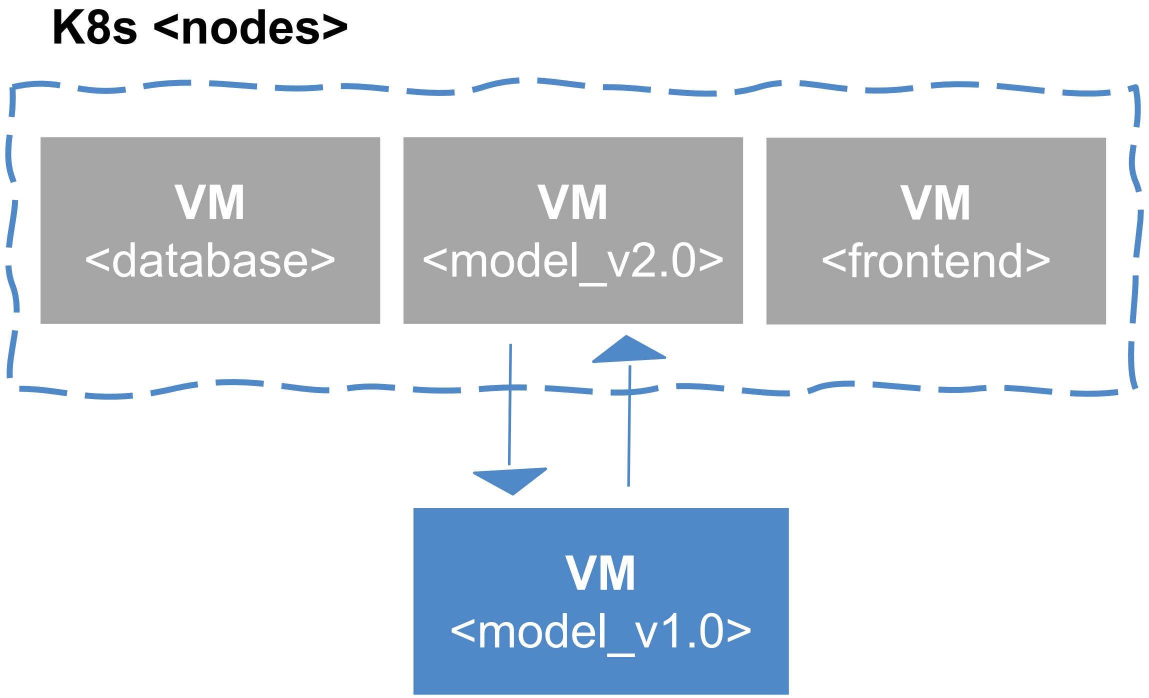
Velocity is measured not in terms of the number of things you can ship while **maintaining a highly available service**



Advantages of using Kubernetes: **Velocity**

Velocity is enabled by:

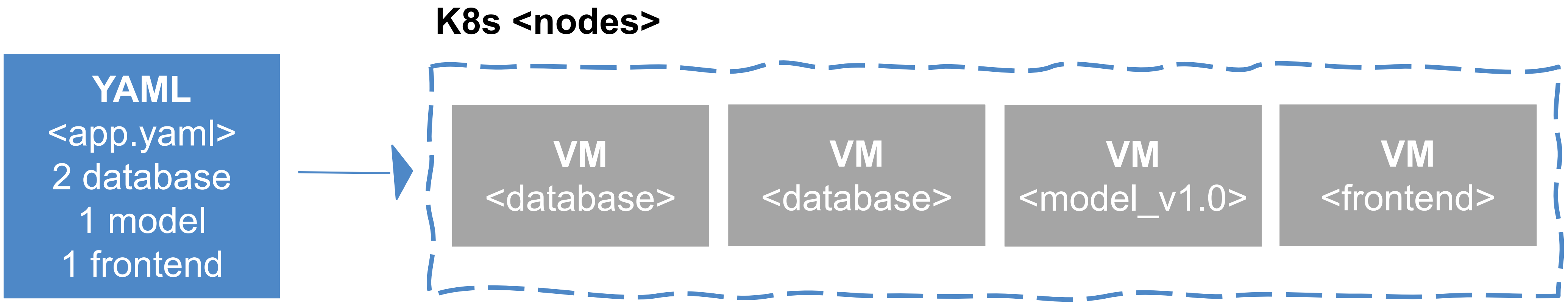
- **Immutable system:** you can't change running container, but you create a new one and replace it in case of failure (allows for keeping track of the history and load older images)



Advantages of using Kubernetes: **Velocity**

Velocity is enabled by:

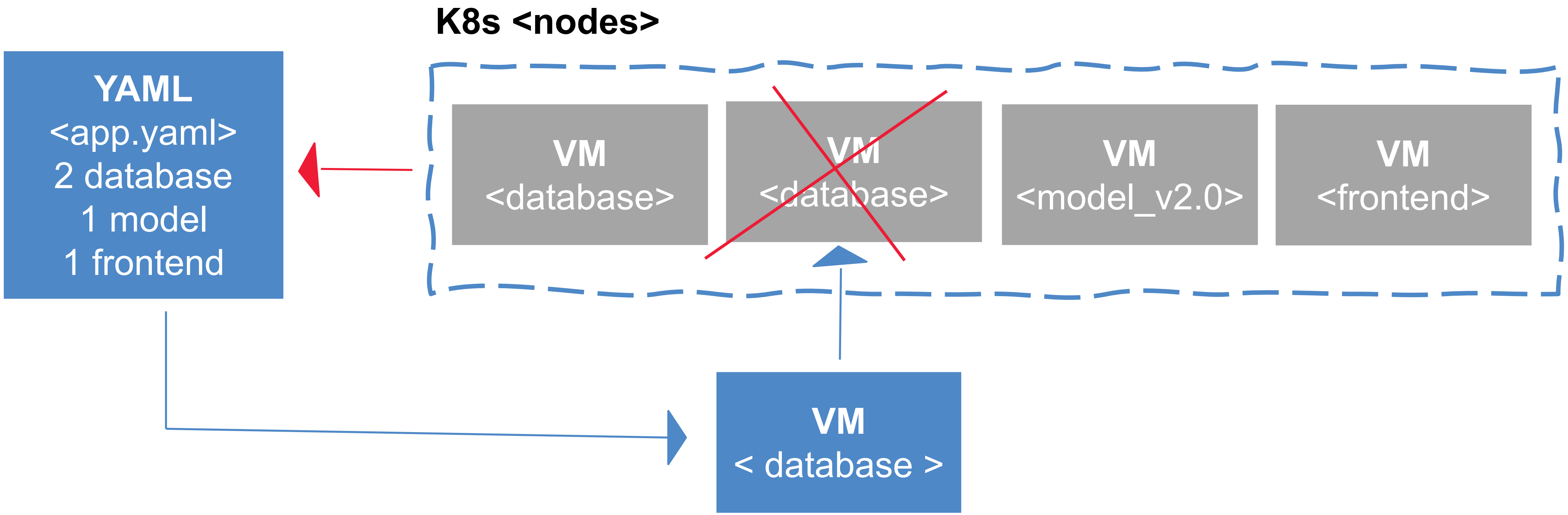
- **Declarative configuration:** you can define the desired state of the system restating the previous declarative state to go back. ***Imperative*** configuration are defined by the execution of a series of instructions, but not the other way around.



Advantages of using Kubernetes: **Velocity**

Velocity is enabled by:

- **Online self-healing systems:** k8s takes actions to ensure that the current state matches the desired state (as opposed to an operator fixing the repair)



Advantages of using Kubernetes: **Abstraction**

Kubernetes allows to build, deploy, and manage your application in a way that is portable across a wide variety of environments. The move to application-oriented container APIs like Kubernetes has two concrete benefits:

- **separation**: developers from specific machines
- **portability**: simply a matter of sending the declarative config to a new cluster

Advantages of using Kubernetes: **Efficiency**

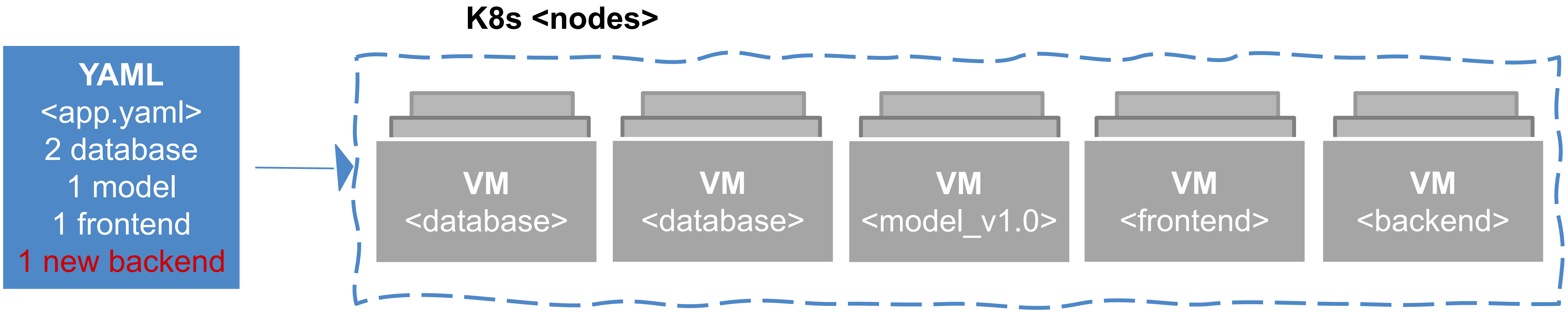
There are concrete economic benefit to the abstraction because tasks from multiple users can be packed tightly onto fewer machines:

- **Consume less energy** (ratio of the useful to the total amount)
- **Limit costs of running a server** (power usage, cooling requirements, datacenter space, and raw compute power)
- **Create quickly a developer's test environment** as a set of containers
- **Reduce cost of development instances in your stack**, liberating resources to develop others that were cost-prohibitive

Advantages of using Kubernetes: **Scaling**

As your product grows, it's inevitable that you will need to scale:

- Software
- Infrastructure
- Team/s that develop it



Advantages of using Kubernetes: **Scaling**

Kubernetes provides numerous advantages to address scaling:

- **Decoupled architectures**: each component is separated from other components by defined APIs and *service load balancers*.
- **Easy scaling for applications and clusters**: simply changing a number in a configuration file, k8s takes care of the rest (part of declarative).
- **Scaling development teams with microservices**: small team is responsible for the design and delivery of a service that is consumed by other small teams (optimal group size: 2 pizzas team).

Advantages of using Kubernetes: **Scaling**

Kubernetes provides numerous **abstractions** and APIs that help building these decoupled microservice architectures:

- **Pods** can group together container images developed by different teams into a single deployable unit (similar to docker-compose)
- **Other services to isolate** one microservice from another such (e.g. load balancing, naming, and discovery)
- **Namespaces** control the interaction among services
- **Ingress** combine multiple microservices into a single externalized API (easy-to-use frontend)

K8s provides full spectrum of solutions between doing it “the hard way” and a fully managed service

THANK YOU