INTRO TO KUBERNETES AND SERVERLESS

TWO SIDES OF THE DEPLOYMENT COIN

WES ROLNICK, NICK PRUSKO

INTRODUCTION

- Who this talk is intended for

INTRODUCTION

- If you want to follow along:
  - Create a free Google Cloud account
  - Log into: https://labs.play-with-k8s.com/

INTRODUCTION

- How we used to deploy
- The container revolution

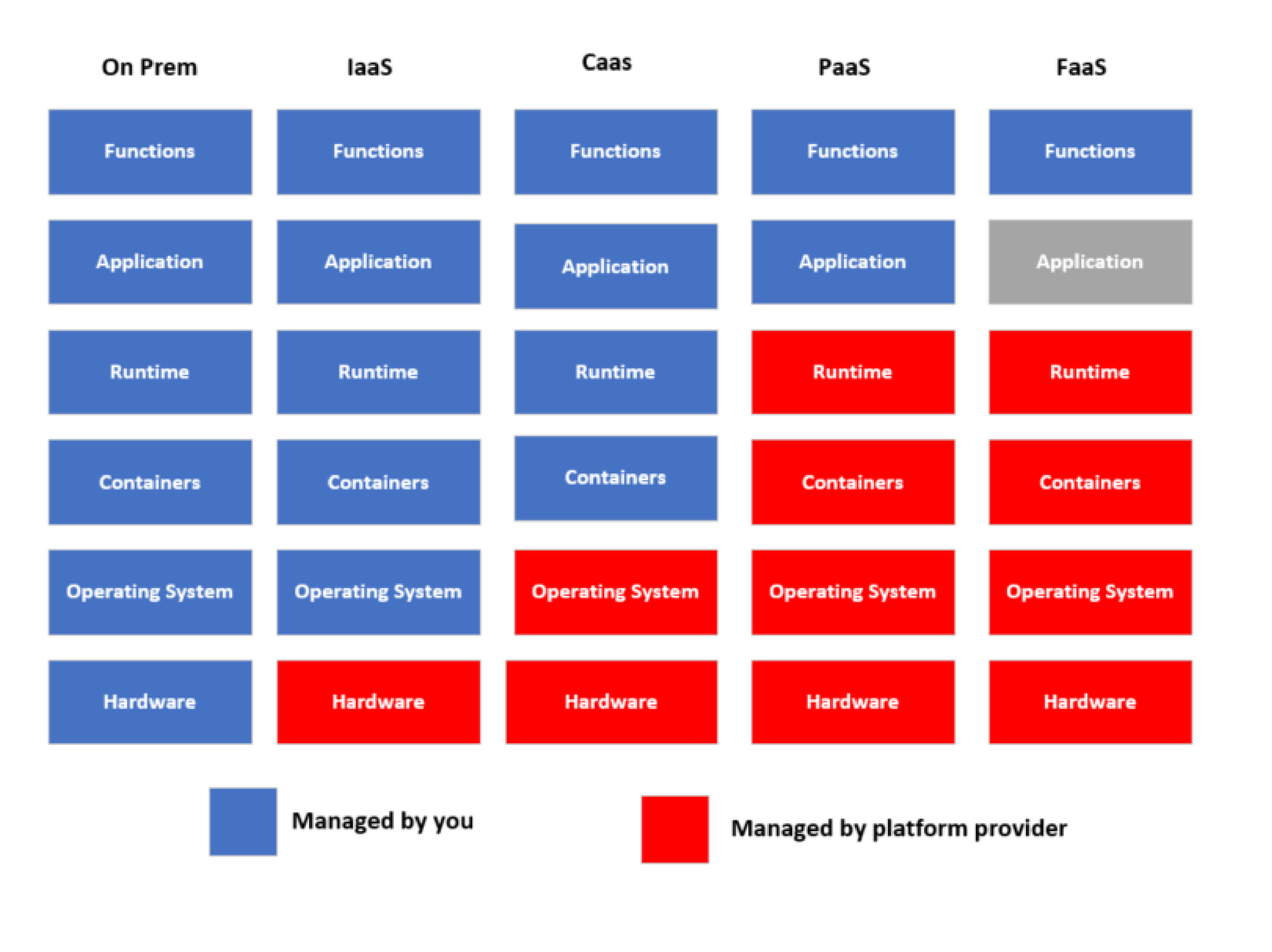
INTRODUCTION

- Kubernetes: Deploy containerized applications across managed infrastructure

- Serverless: Deploy code to cloud infrastructure which will handle resource management

KUBERNETES AND SERVERLESS

Kubernetes	Serverless
+ Highly configurable	+ Quick Implementation
+ Complete control of data	+ Low cost start
+ Control of resources	+ Scales easily
- Implementation Complexity	- May be more expensive at scale
- High up-front cost	- Platform dependent



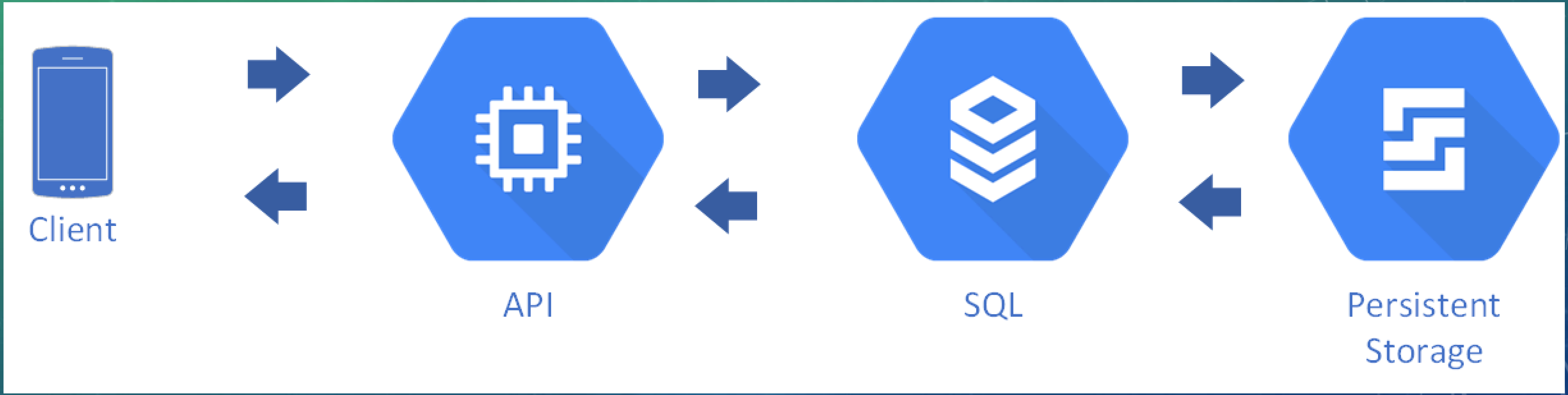
On Prem	IaaS	Caas	PaaS	FaaS
🟦 Functions	🟦 Functions	🟦 Functions	🟦 Functions	🟦 Functions
🟦 Application	🟦 Application	🟦 Application	🟦 Application	▨ Application
🟦 Runtime	🟦 Runtime	🟦 Runtime	🟥 Runtime	🟥 Runtime
🟦 Containers	🟦 Containers	🟦 Containers	🟥 Containers	🟥 Containers
🟦 Operating System	🟦 Operating System	🟥 Operating System	🟥 Operating System	🟥 Operating System
🟦 Hardware	🟥 Hardware	🟥 Hardware	🟥 Hardware	🟥 Hardware

🟦 Managed by you

🟥 Managed by platform provider

THE DEMO PROJECT

- We will be using a simple Python REST API.
- MySQL database for storing data.
- Google Cloud Serverless for serverless deployment (sign up for a trial account if you want to follow along)
- http://sharksareawesome.com is the demo site.
- Also log into https://labs.play-with-k8s.com/
- Code is available on Github: https://github.com/syncrisis/kubernetesdemo


Client
API
SQL
Persistent Storage

WHAT IS KUBERNETES

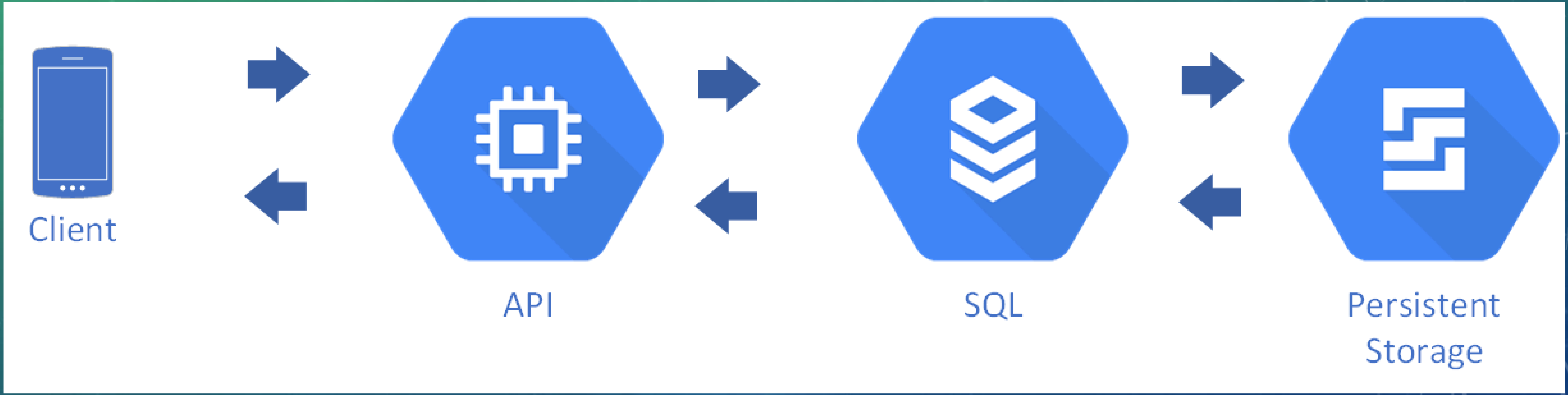
- Production ready container management
- Manages container lifecycles
- Manages updates and deployments

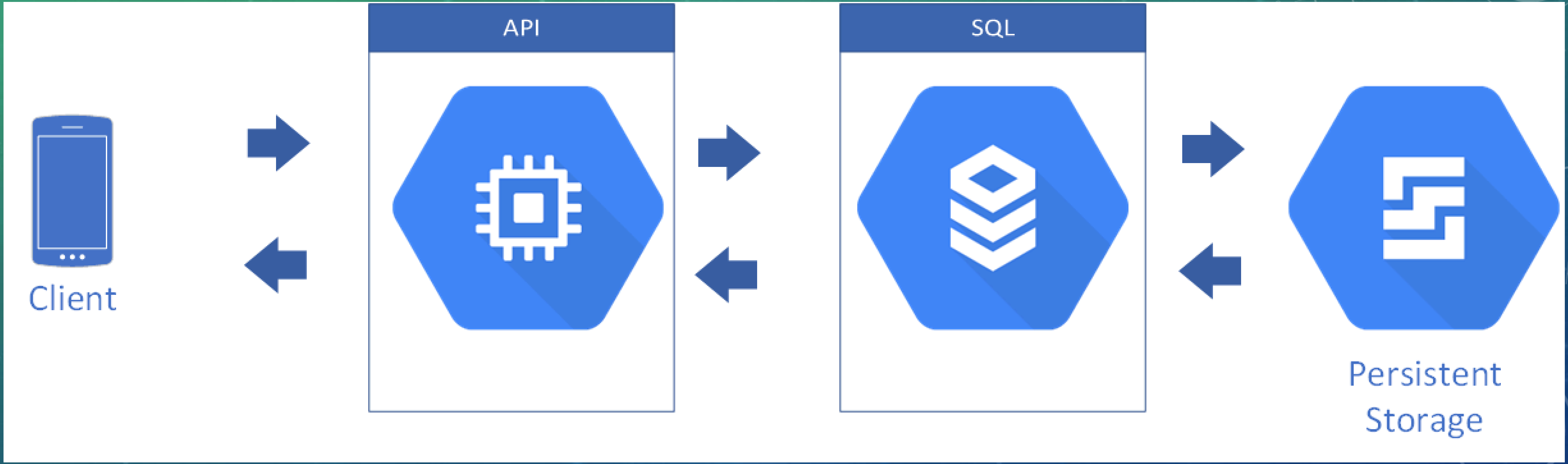
KUBERNETES TERMS

- Cluster: A collection of nodes that are being managed by Kubernetes
- Node: A VM or physical machine that contains one or more containers
- Master Node: The node that coordinates everything in the cluster.

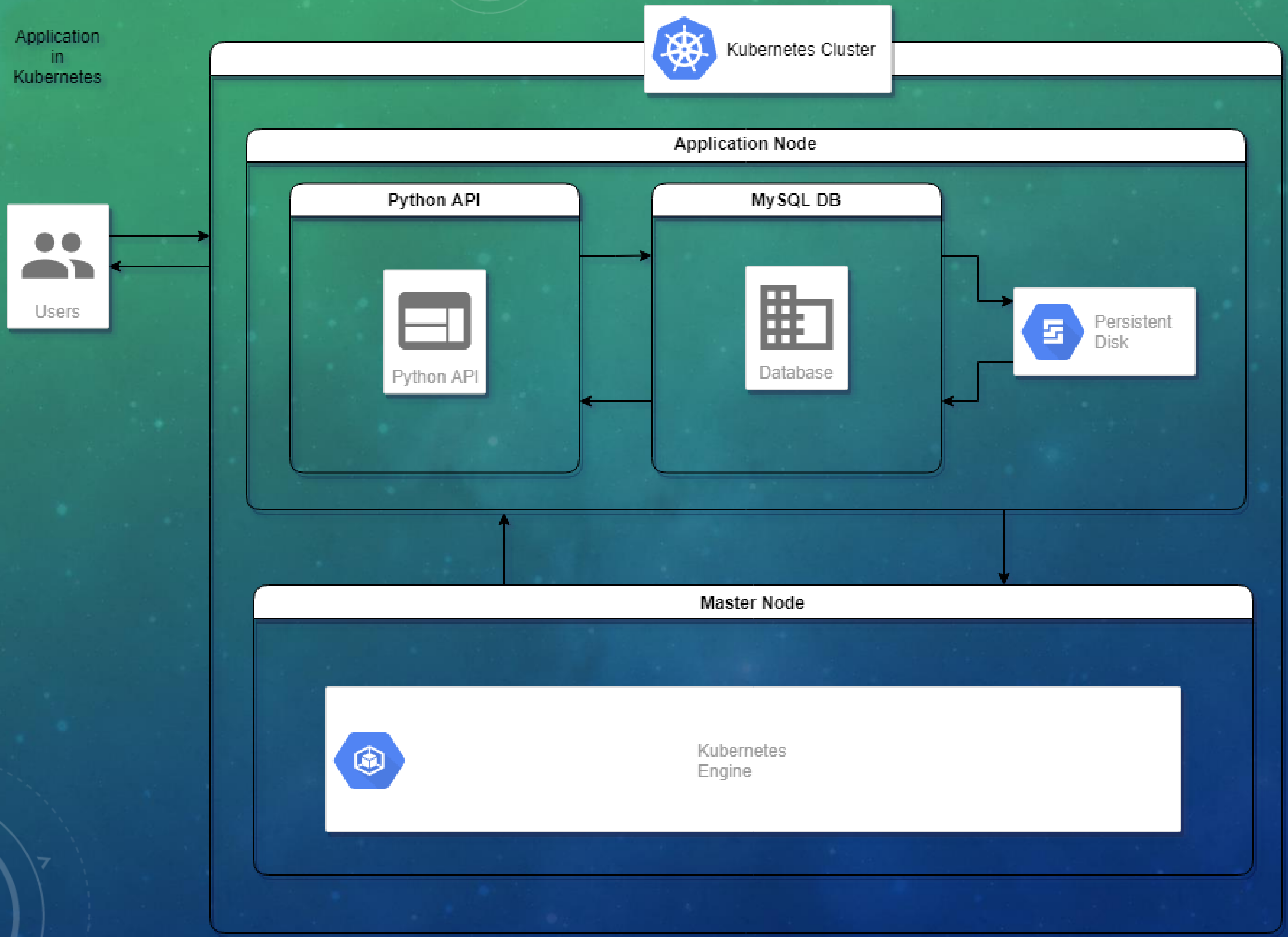
KUBERNETES OBJECTS

- Pod: A running process on the cluster. Can be an application container (sometimes multiple containers).
- Service: Defines policies for accessing Pods.
- Deployment: Describes the desired state for how pods should be deployed across a cluster.


Client
API
SQL
Persistent Storage


Client
API
SQL
Persistent Storage

Application in Kubernetes



Kubernetes Cluster

Users

Application Node

Python API

Python API

My SQL DB

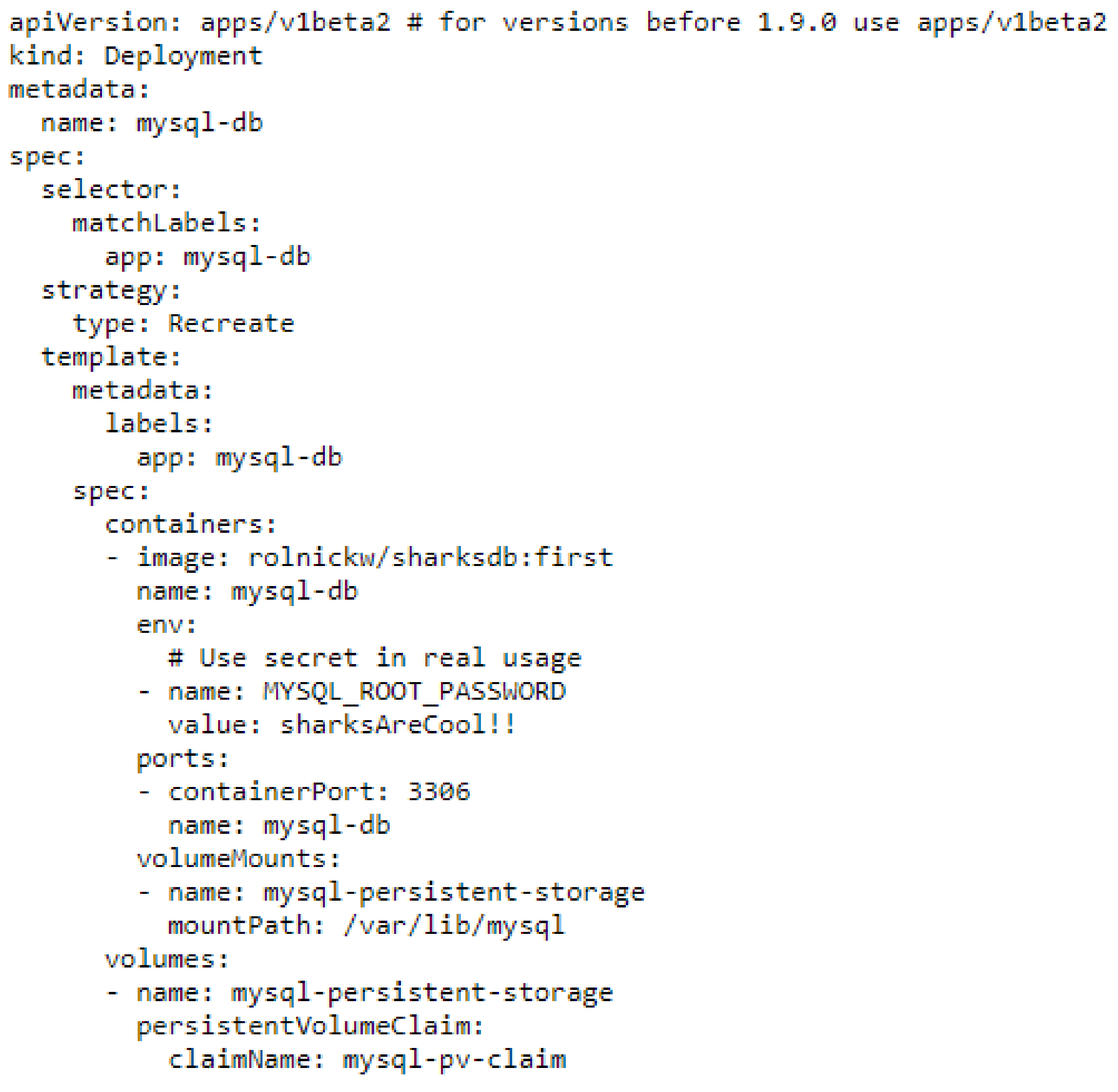
Database

Persistent Disk

Master Node

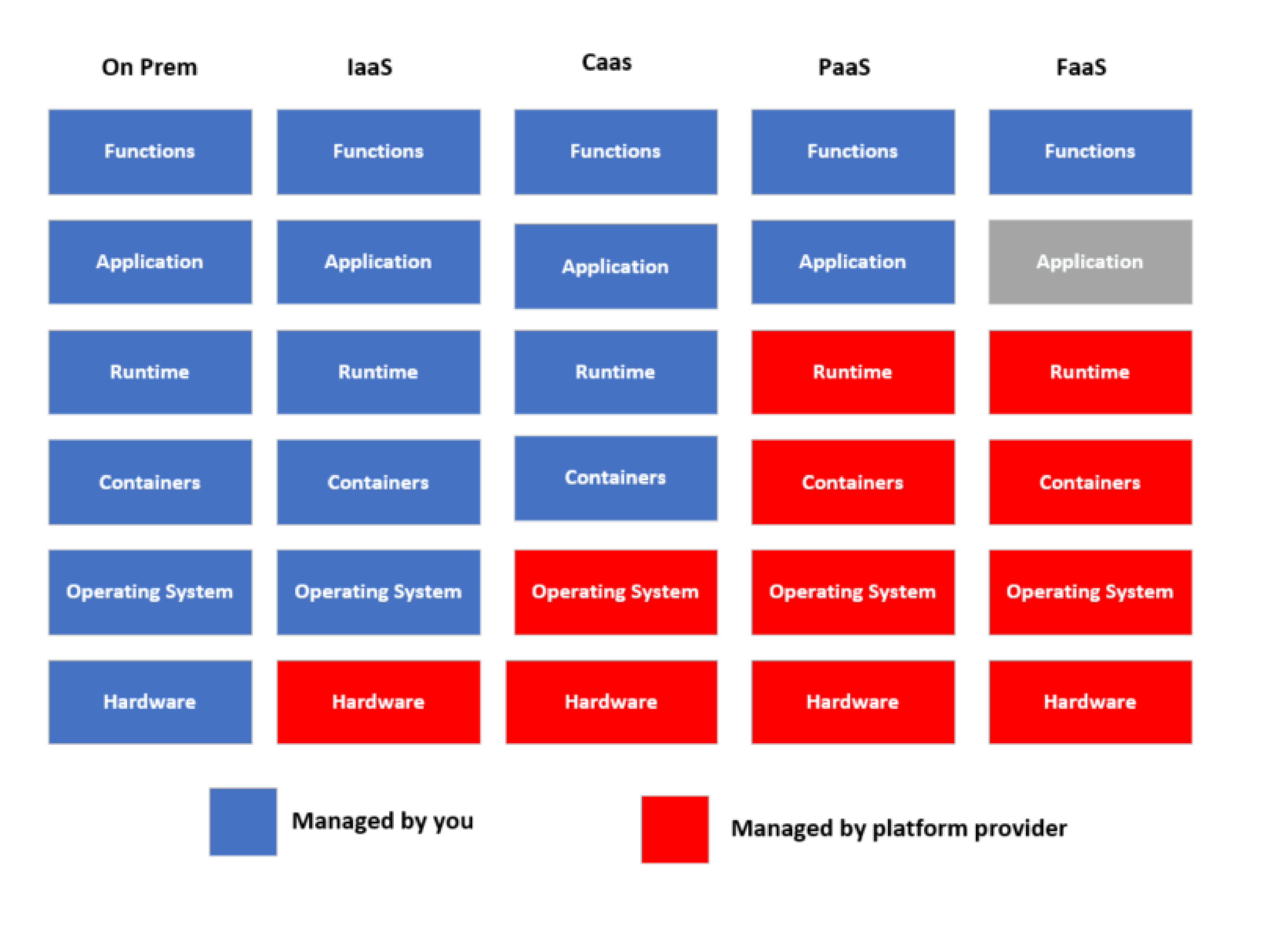
Kubernetes Engine

YAML!



```yaml
apiVersion: apps/v1beta2 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: mysql-db
spec:
  selector:
    matchLabels:
      app: mysql-db
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql-db
    spec:
      containers:
      - image: rolnickw/sharksdb:first
        name: mysql-db
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: sharksAreCool!!
        ports:
        - containerPort: 3306
          name: mysql-db
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```

DEMO



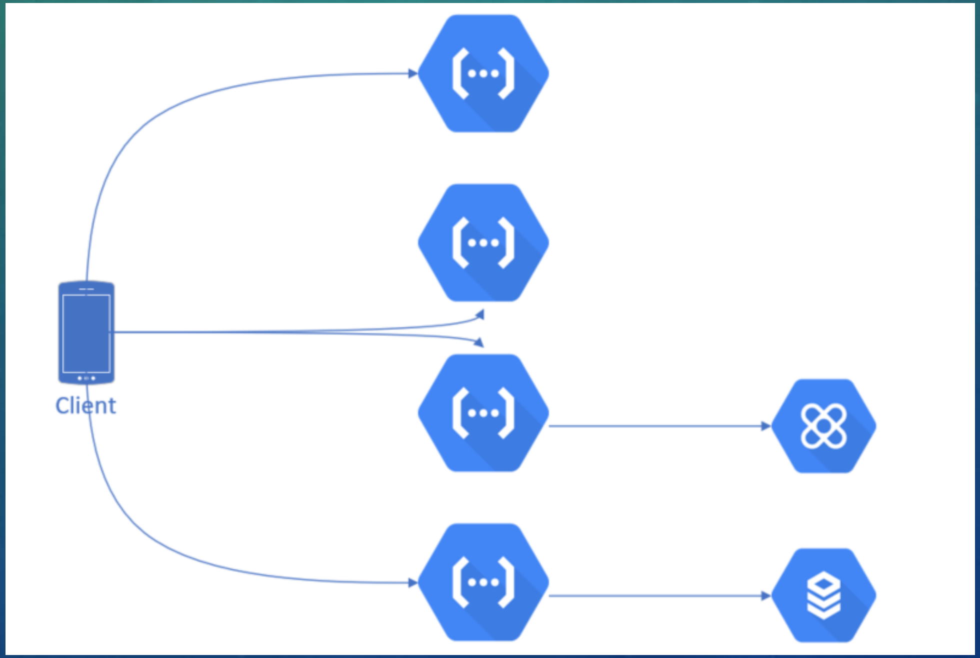
On Prem	IaaS	Caas	PaaS	FaaS
🟦 Functions	🟦 Functions	🟦 Functions	🟦 Functions	🟦 Functions
🟦 Application	🟦 Application	🟦 Application	🟦 Application	⬜ Application
🟦 Runtime	🟦 Runtime	🟦 Runtime	🟥 Runtime	🟥 Runtime
🟦 Containers	🟦 Containers	🟦 Containers	🟥 Containers	🟥 Containers
🟦 Operating System	🟦 Operating System	🟥 Operating System	🟥 Operating System	🟥 Operating System
🟦 Hardware	🟥 Hardware	🟥 Hardware	🟥 Hardware	🟥 Hardware

🟦 Managed by you

🟥 Managed by platform provider

WHAT IS SERVERLESS?

- Allows your code to run without the headache of server management, maintenance, scaling strategies. Back end infrastructure becomes invisible



WHY WOULD I?

- Allows developers to focus on development rather than infrastructure, small teams can do big things
- Speeds up timelines to bring ideas to market
- Reduction in cost

LANGUAGES

Google Cloud

node JS®

python™

amazon web services™

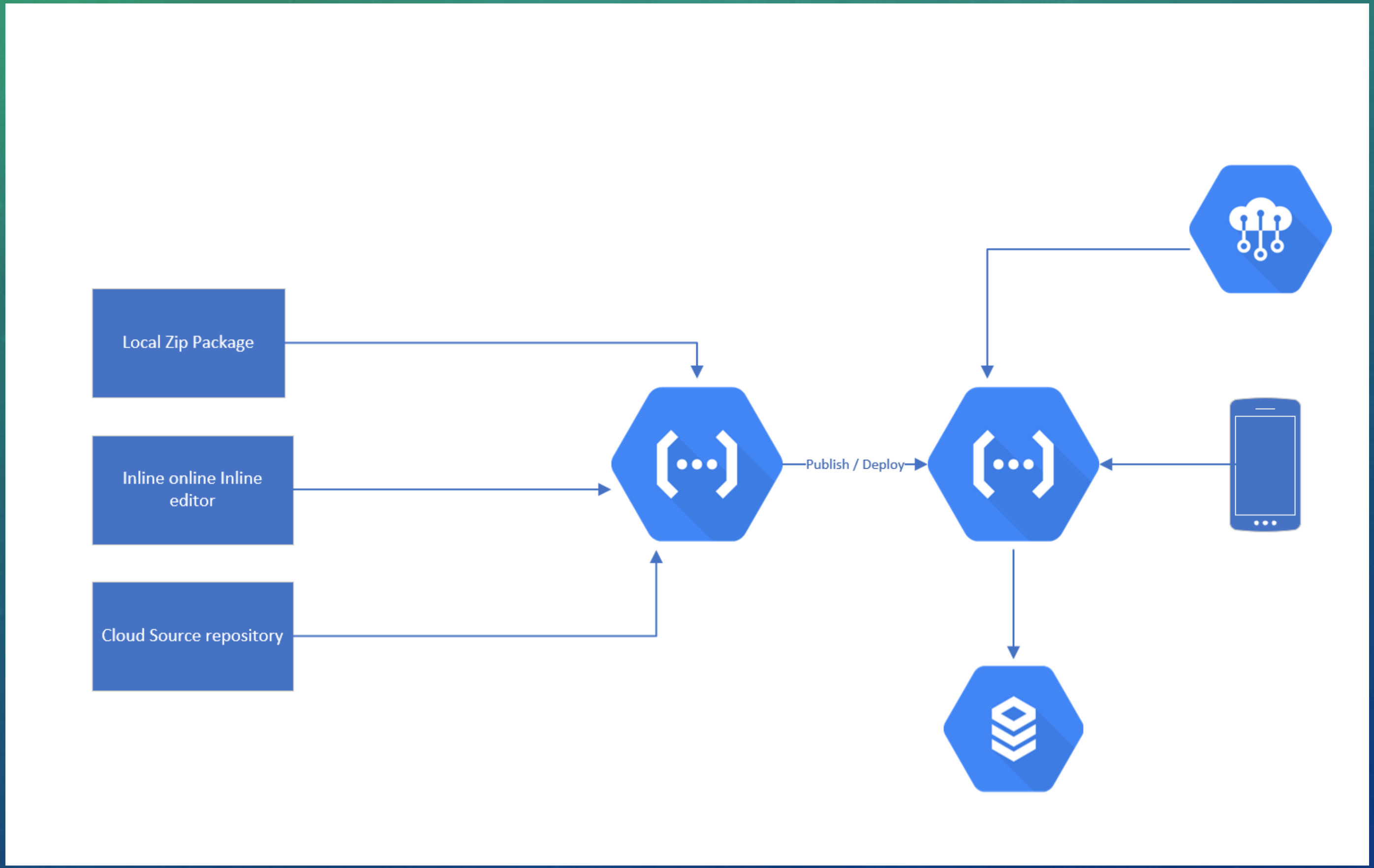
Java™

C#

EVENTS & TRIGGERS

- Events & triggers provide hooks into your cloud environment
  - HTTP – GET / POST / PUT / PATCH / DELETE
  - Storage – Event Driven functions from your platform
  - Pub / Sub -
  - Firebase events

HOW DOES IT ALL WORK?



DEMO

THE FUTURE

- Kubernetes + Serverless!

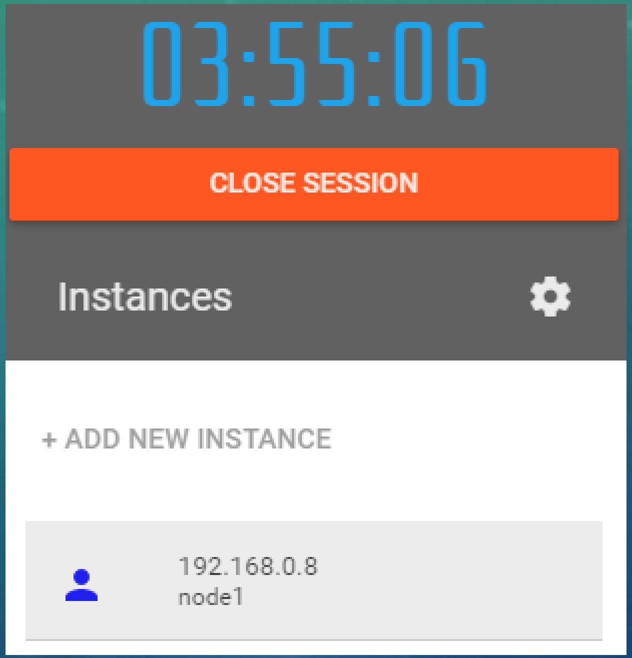
CONCLUSION

- It really depends…

KUBERNETES PLAYGROUND DEMO

GO TO HTTPS://LABS.PLAY-WITH-K8S.COM/

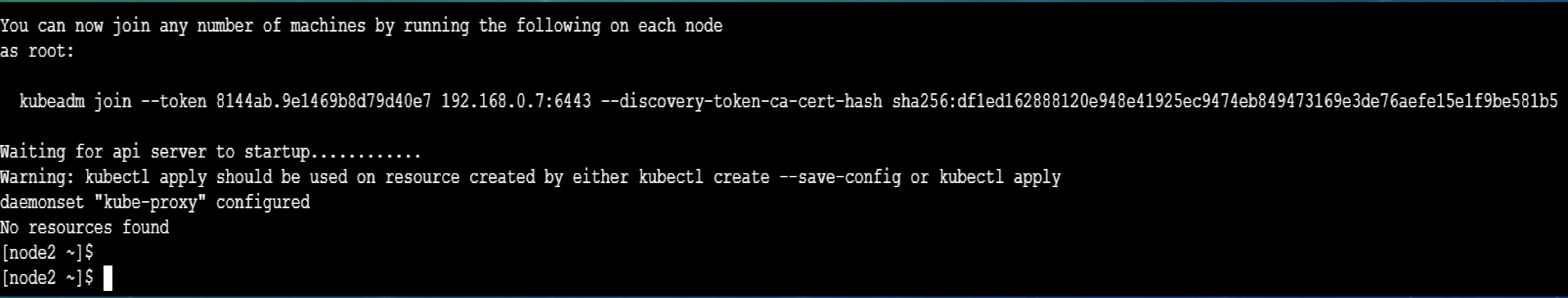
1. Click Add New Instance:



HTTPS://LABS.PLAY-WITH-K8S.COM/

2. Copy the “Initializes cluster master node:” line:

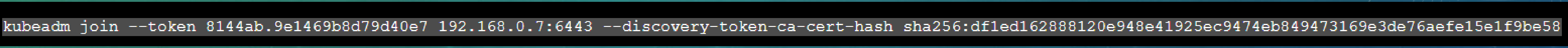
   kubeadm init --apiserver-advertise-address $(hostname -i)



```
You can now join any number of machines by running the following on each node
as root:

  kubeadm join --token 8144ab.9e1469b8d79d40e7 192.168.0.7:6443 --discovery-token-ca-cert-hash sha256:df1ed162888120e948e41925ec9474eb849473169e3de76aefe15e1f9be581b5

Waiting for api server to startup............
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
daemonset "kube-proxy" configured
No resources found
[node2 ~]$
[node2 ~]$
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

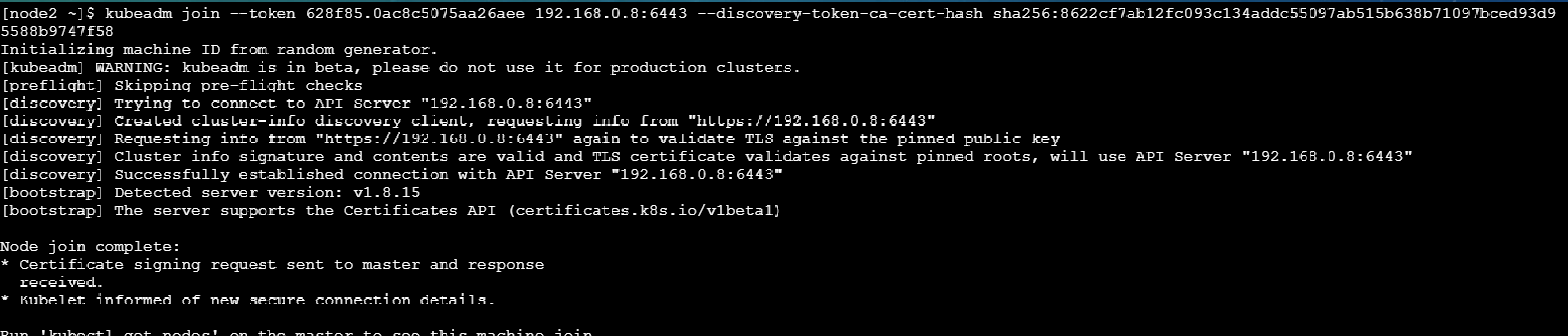
3. Copy the entire line that starts kubeadm join –token dkfjkldjfkdj



```
kubeadm join --token 8144ab.9e1469b8d79d40e7 192.168.0.7:6443 --discovery-token-ca-cert-hash sha256:df1ed162888120e948e41925ec9474eb849473169e3de76aefe15e1f9be58
```

4. Click Add New Instance Again

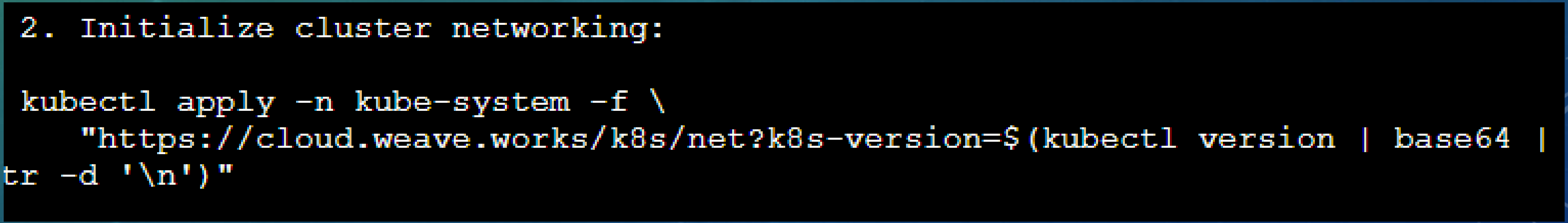
5. In the second instance paste the copied kubeadm join line:



```
[node2 ~]$ kubeadm join --token 628f85.0ac8c5075aa26aee 192.168.0.8:6443 --discovery-token-ca-cert-hash sha256:8622cf7ab12fc093c134addc55097ab515b638b71097bced93d9
5588b9747f58
Initializing machine ID from random generator.
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production clusters.
[preflight] Skipping pre-flight checks
[discovery] Trying to connect to API Server "192.168.0.8:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.0.8:6443"
[discovery] Requesting info from "https://192.168.0.8:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "192.168.0.8:6443"
[discovery] Successfully established connection with API Server "192.168.0.8:6443"
[bootstrap] Detected server version: v1.8.15
[bootstrap] The server supports the Certificates API (certificates.k8s.io/v1beta1)

Node join complete:
* Certificate signing request sent to master and response
  received.
* Kubelet informed of new secure connection details.

Run 'kubectl get nodes' on the master to see this machine join.
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

6. Switch back to your first instance (the master)
7. Copy the “Initialize cluster networking command:



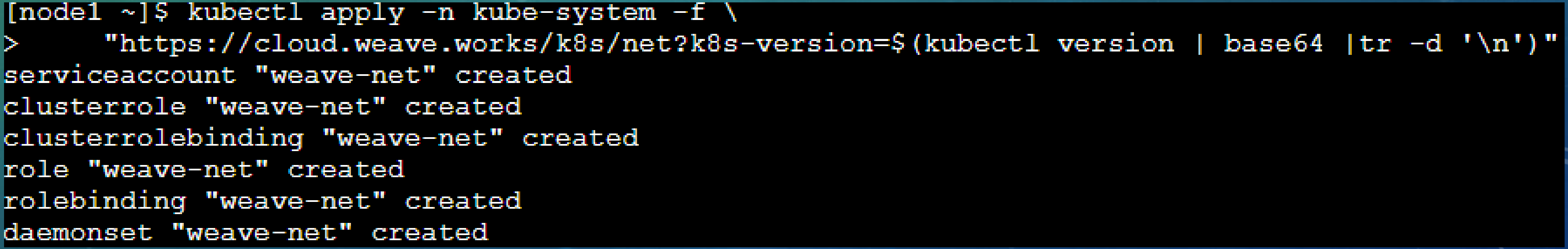
```
 2. Initialize cluster networking:

 kubectl apply -n kube-system -f \
    "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 |
tr -d '\n')"
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

8. Paste and run this in the master instance (the first instance):

   Kubectl apply –n kube-system –f \

   “https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 |tr -d ‘\n’)”

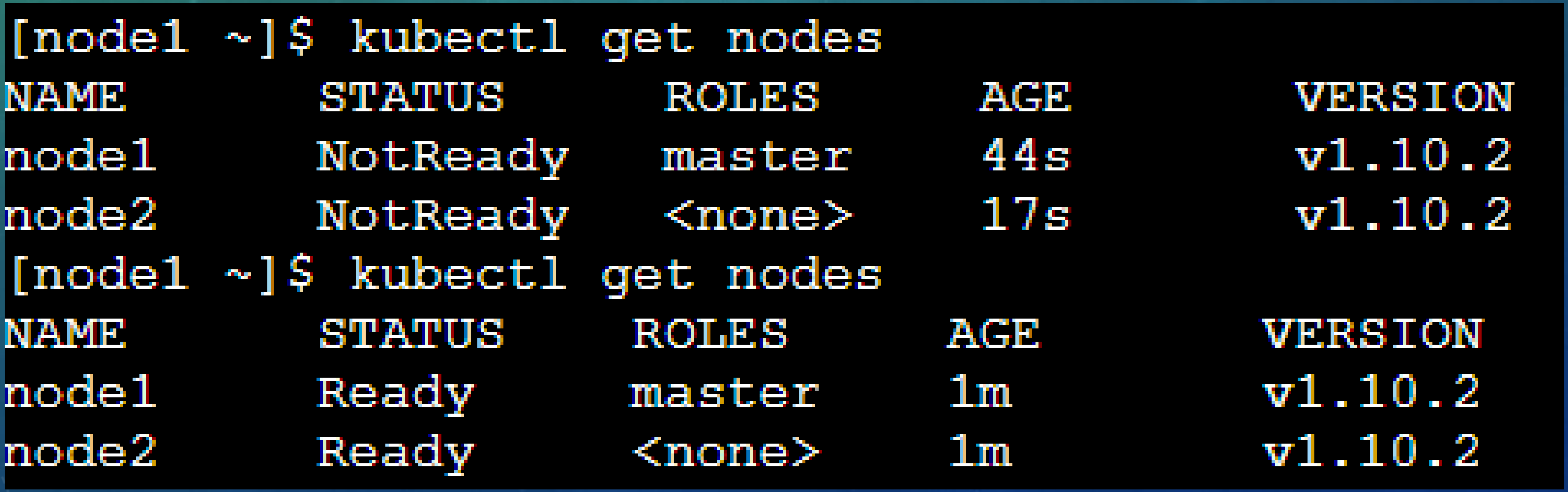


```
[node1 ~]$ kubectl apply -n kube-system -f \
>     "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 |tr -d '\n')"
serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
role "weave-net" created
rolebinding "weave-net" created
daemonset "weave-net" created
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

9. Verify that the nodes are configured and “Ready” by running:

   Kubectl get nodes

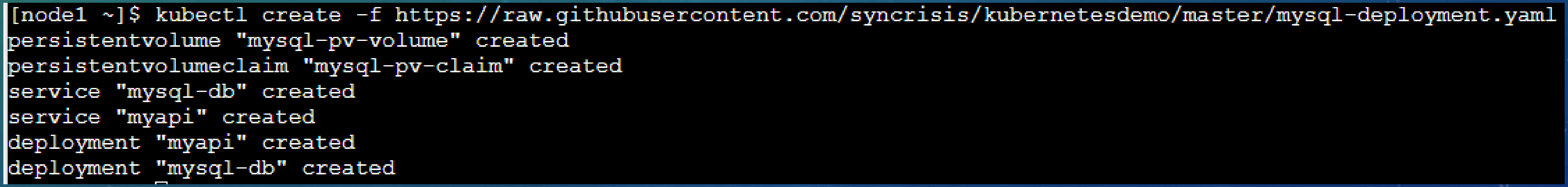


```
[node1 ~]$ kubectl get nodes
NAME     STATUS     ROLES     AGE       VERSION
node1    NotReady   master     44s      v1.10.2
node2    NotReady   <none>     17s      v1.10.2
[node1 ~]$ kubectl get nodes
NAME     STATUS    ROLES     AGE       VERSION
node1    Ready     master    1m        v1.10.2
node2    Ready     <none>    1m        v1.10.2
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

10. In the master node enter:

kubectl create –f https://raw.githubusercontent.com/syncrisis/kubernetesdemo/master/mysql-deployment.yaml

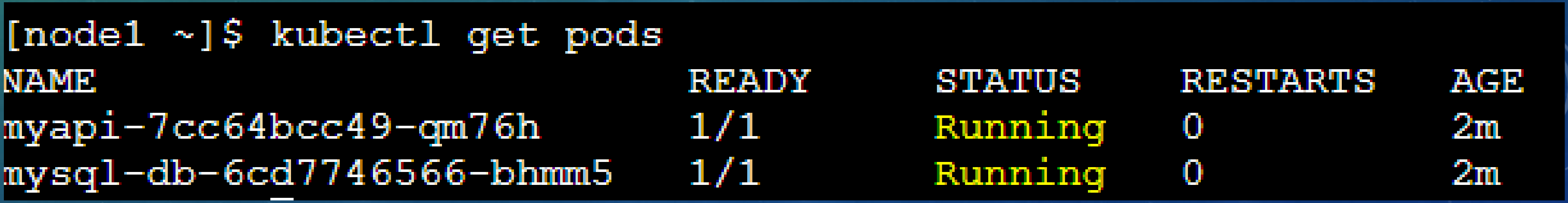


```
[node1 ~]$ kubectl create -f https://raw.githubusercontent.com/syncrisis/kubernetesdemo/master/mysql-deployment.yaml
persistentvolume "mysql-pv-volume" created
persistentvolumeclaim "mysql-pv-claim" created
service "mysql-db" created
service "myapi" created
deployment "myapi" created
deployment "mysql-db" created
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

11. Wait for the pods to finish creating and run

```
kubectl get pods
```

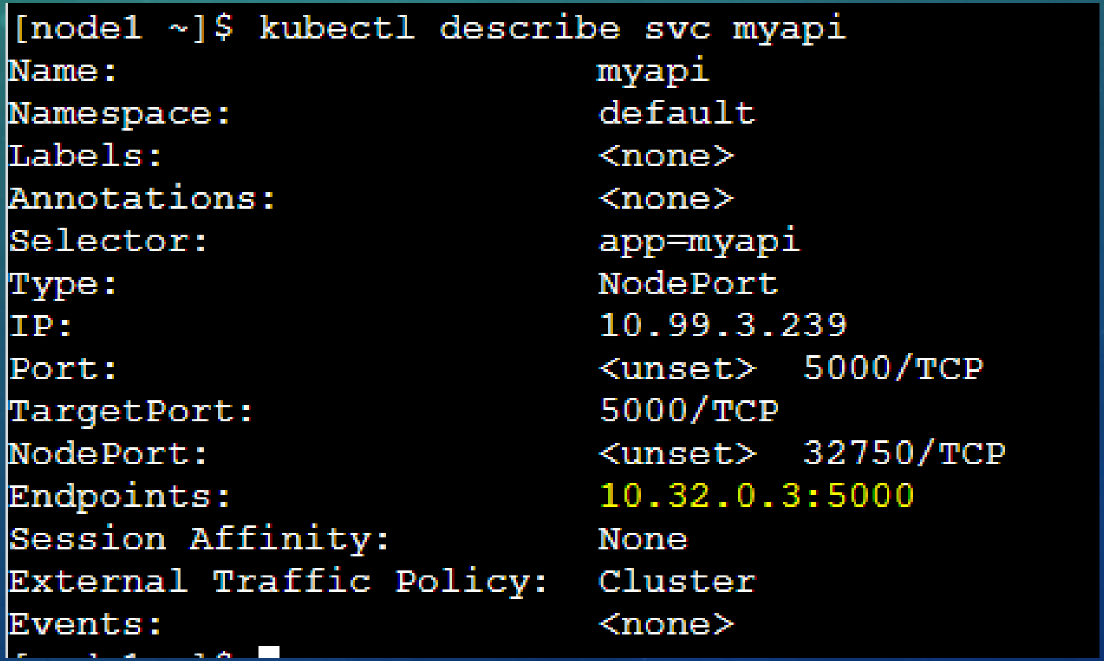


```
[node1 ~]$ kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
myapi-7cc64bcc49-qm76h        1/1       Running   0          2m
mysql-db-6cd7746566-bhmm5     1/1       Running   0          2m
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

12. Find the exposed API port by typing:

    kubectl describe svc myapi

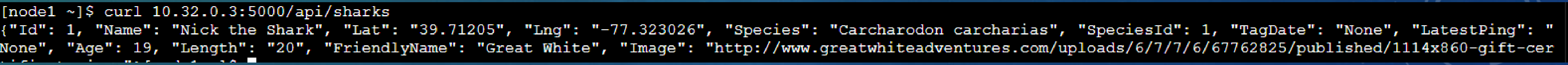


```
[node1 ~]$ kubectl describe svc myapi
Name:                     myapi
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=myapi
Type:                     NodePort
IP:                       10.99.3.239
Port:                     <unset>  5000/TCP
TargetPort:               5000/TCP
NodePort:                 <unset>  32750/TCP
Endpoints:                10.32.0.3:5000
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
[node1 ~]$ 
```

HTTPS://LABS.PLAY-WITH-K8S.COM/

13. Send a curl request to the API at the endpoint:

    curl 10.32.0.3:5000/**api/sharks** **(the service endpoint for the IP address)**



```
[node1 ~]$ curl 10.32.0.3:5000/api/sharks
{"Id": 1, "Name": "Nick the Shark", "Lat": "39.71205", "Lng": "-77.323026", "Species": "Carcharodon carcharias", "SpeciesId": 1, "TagDate": "None", "LatestPing": "
None", "Age": 19, "Length": "20", "FriendlyName": "Great White", "Image": "http://www.greatwhiteadventures.com/uploads/6/7/7/6/67762825/published/1114x860-gift-cer
```