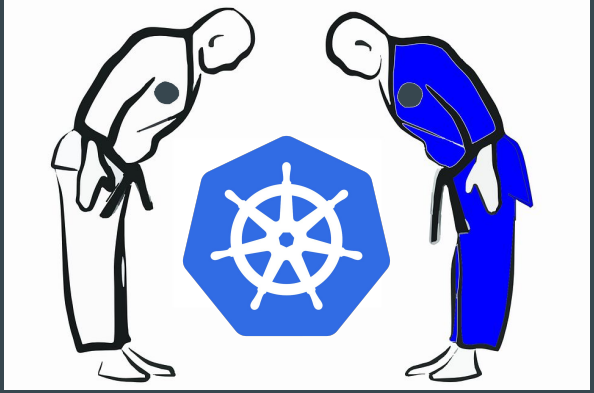
Barry Tarlton
Michael Frayer
Mark Ramsey



Docker & Kubernetes Dojo

•••

github.com/javaplus/DockerKubesDojo

Please Verify Your installations by following the
“Testing your Installations”
at the link above.

Who are we and what is a DOJO?

Michael Frayer

Mark Ramsey

Barry Tarlton

•••

A dōjō is a hall or space for immersive learning or meditation. This is traditionally in the field of martial arts, but has been seen increasingly in other fields, such as meditation and software development. The term literally means "place of the Way" in Japanese.

Tiny Containers: Exploring the World of Docker and Kubernetes with a Raspberry Pi Cluster

•••

Friday, 1:30-2:30
(Nationwide Sponsor Session)
Room: Cypress

How Exploding Birthday Cakes and Other CRAZY Projects Come to Life

...

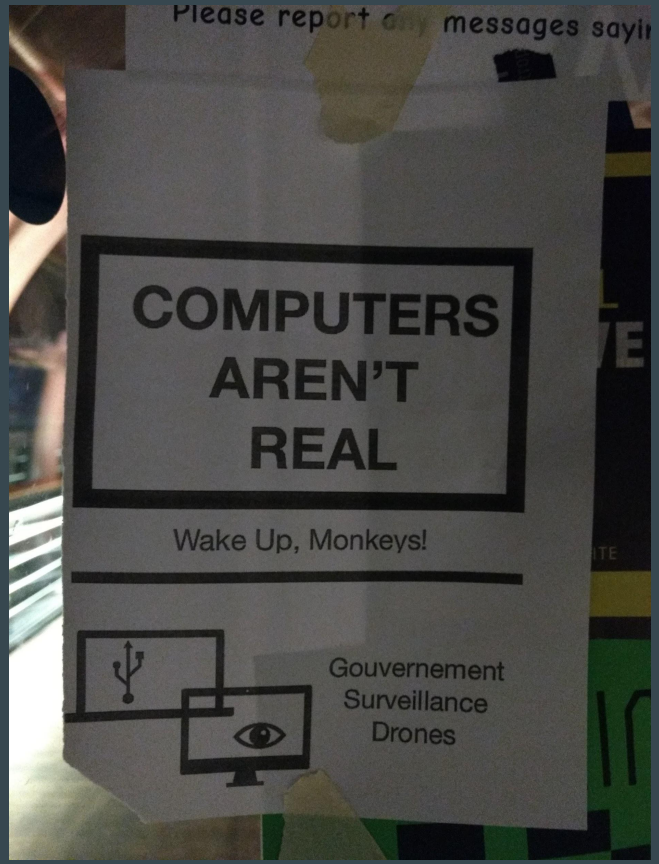
Friday 9:45 AM - 10:45 AM
Room: **Mangrove**

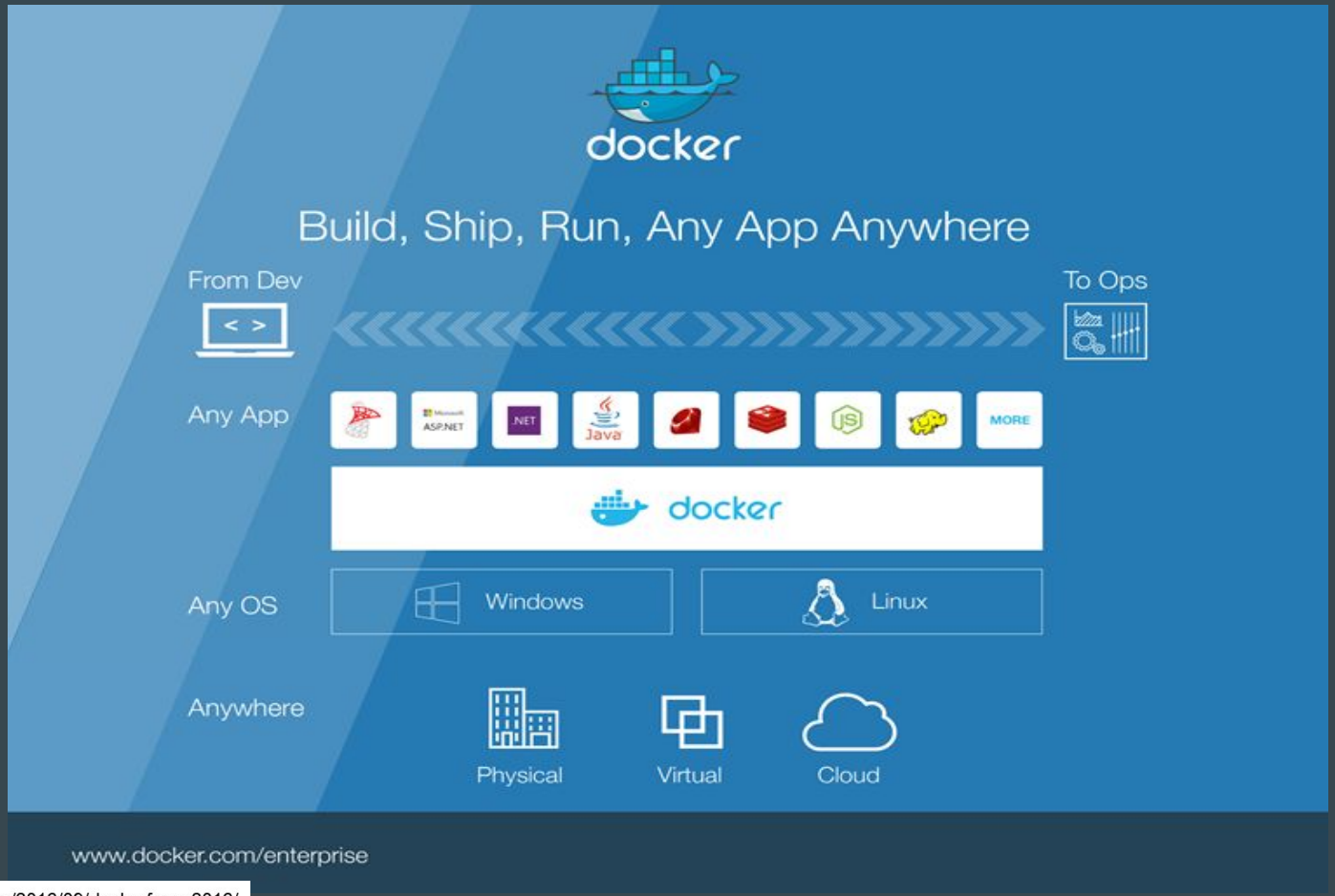
Technical Agenda

- What is Docker?
- Do Docker Stuff...
- Do More Docker Stuff...
- Why Kubernetes?
- Do Kubernetes Stuff...
- Do more Kubernetes Stuff...
- Do even more Kubernetes Stuff... and maybe some Docker stuffs

Real Agenda

- What is Docker?
- Running Docker Containers
- Create a Docker Image
- Why Kubernetes?
- Running containers in Kubernetes
- Declarative Configuration with Kubernetes
- Using Docker for Tooling
- Working With Environment Variables
- Services, Ingress, Readiness, etc...

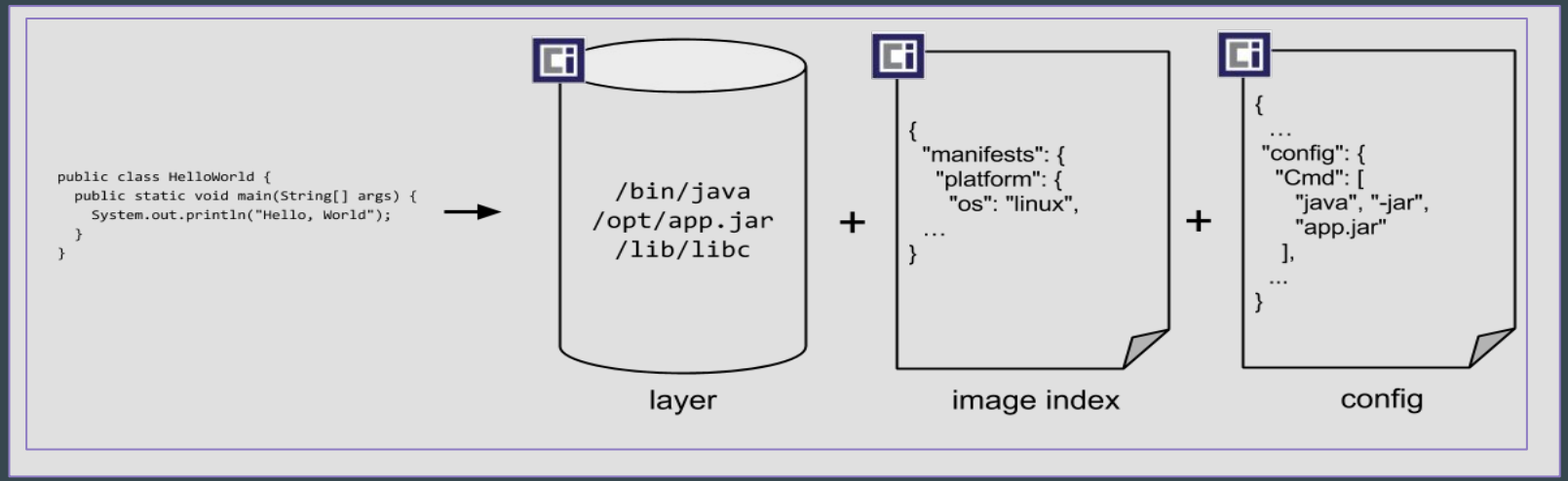


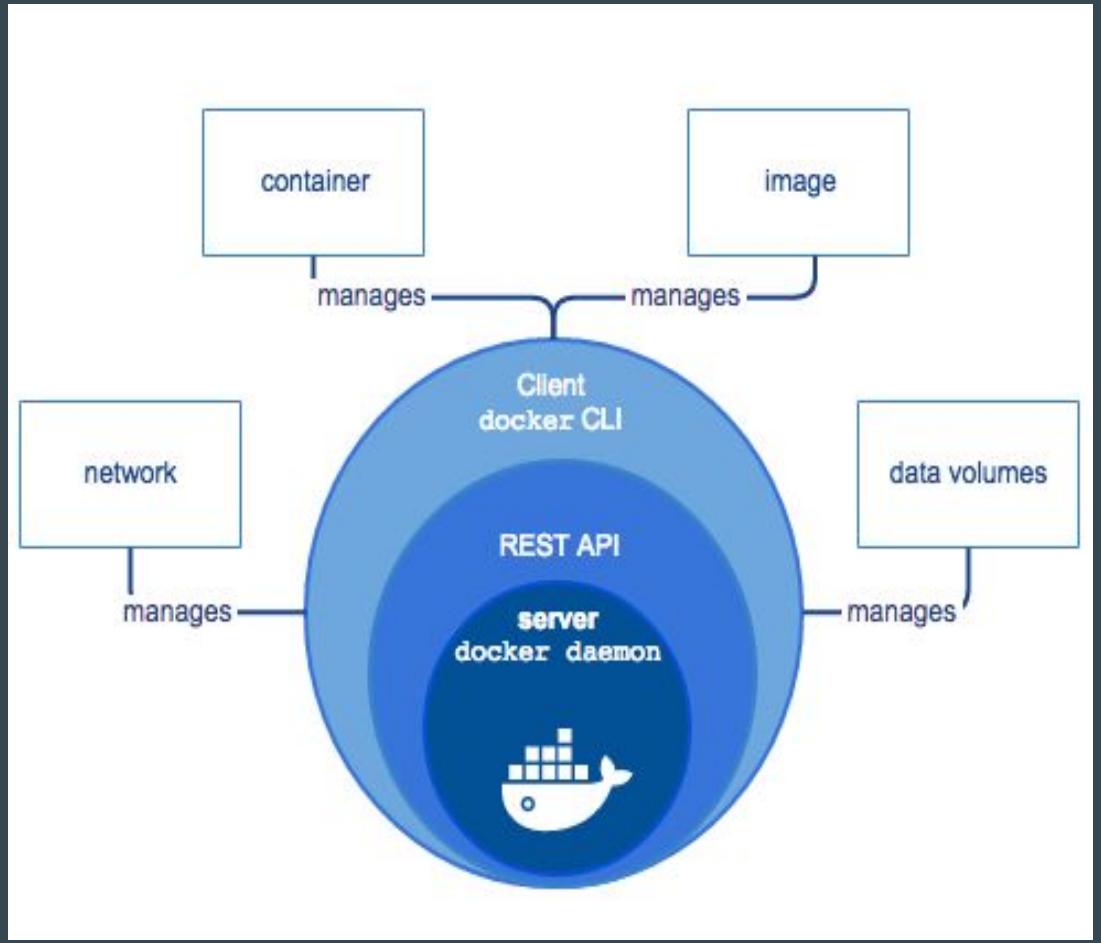


https://blog.docker.com/2016/09/dockerforws2016/

What is Docker?

- Image Format Specification - how to specify the manifest, filesystem layers, and configuration of an image
- Image Distribution Specification - how to distribute images
- Container Runtime Specification - how to execute a container on a specific OS





Docker Architecture

- Daemon
- REST API
- Client

NOTE ON LABS

- They are NOT a race!
- Take your time!
- Understand what you are doing!
- Ask questions!
  - That's why we are here!
- GLHF! (Good Luck Have Fun!)

Time for Our FIRST LAB

- Do Lab1
- “Intro to Docker and Containers”

- All links to the labs can be found:
- github.com/javaplus/DockerKubesDojo
- Scroll down into the readme to the “~~Labs~~” section

Lab 1 Debrief

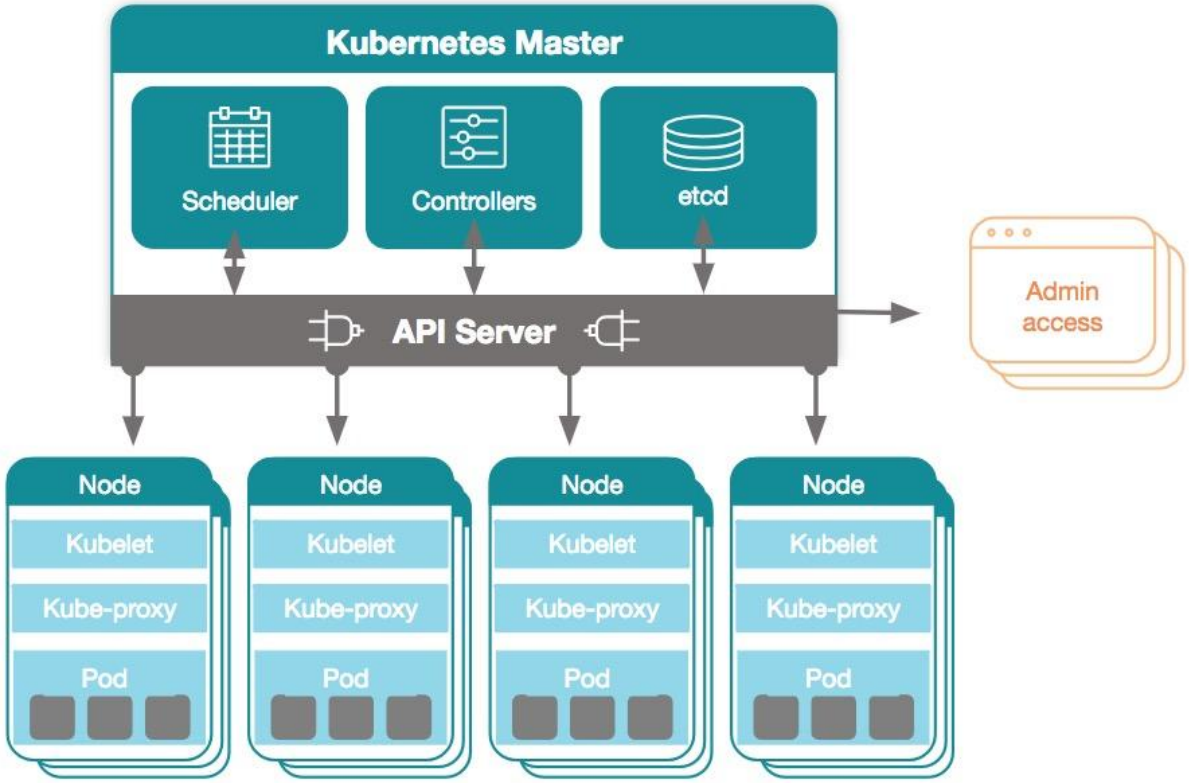
- Q: Where did the hello-world and nginx image definitions come from?
- A: hub.docker.com (technically: registry-1.docker.io)
  - Official images have no prefix
  - Your images would be prefixed with your username or org
  - <registry host name>/<sourceImage>:<tag>

LAB2

- Do Lab2
- “Docker with Cloud Native App”

Why Kubernetes?

Kubernetes Architecture



Take what runs well locally on a single machine with "docker run …" and do the same reliably and securely across a distributed cluster of potentially hundreds of machines

"Container Scheduling & Orchestration"

Resource Utilization
Configuration
Volume Management
Health Checks
Networking

and so much more ...

https://blog.newrelic.com/engineering/what-is-kubernetes/

Lab 3 (Kubernetes finally!)

- Do lab3
- “Kubernetes Intro”

Kubernetes Deployments

Kubernetes Deployments

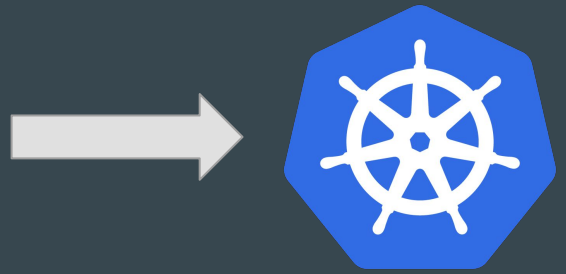
```
docker run \
  --name my-app \
  --env DATABASE=jdbc:postgresql://localhost/test \
  --volume $(pwd)/conf:/usr/local/etc
  my-app:1.0.0
```

Take what runs well locally on a single machine with "docker run …" and do the same reliably and securely across a distributed cluster of potentially hundreds of machines

Deployment

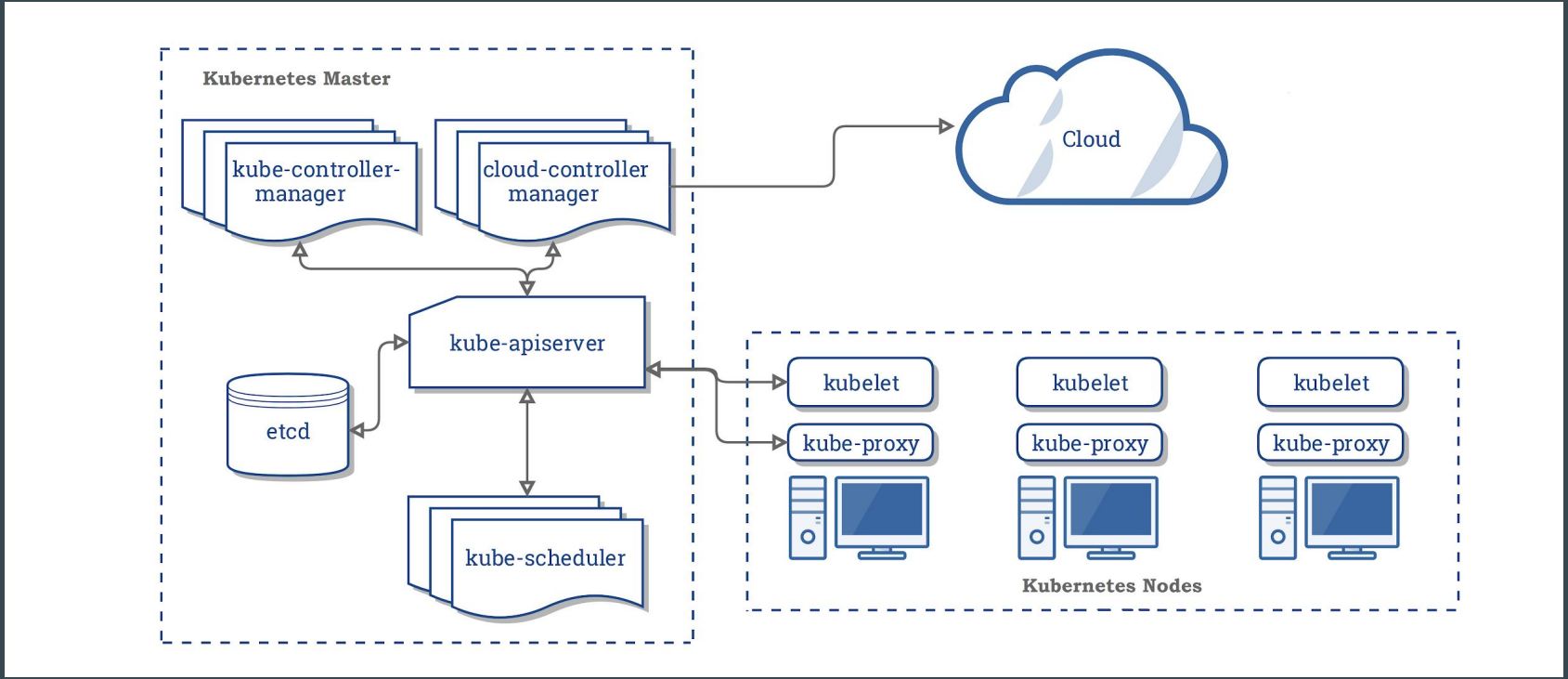
Pod

- Containers
- CPU / Memory
- Env Vars
- Volumes
- Health checks



Kubernetes API Server

Kubernetes API Server



https://kubernetes.io/docs/concepts/overview/components/

Kubernetes API Server

```
kubectl <verb> <resource-noun> <resource-name>
```

```
> kubectl get deployments cn-demo -v 6

I1227 12:10:48.658948    3925 round_trippers.go:443] GET
https://kubernetes.docker.internal:6443/apis/extensions/v1beta1/namespaces/default/deployments/cn-demo 200 OK in
15 milliseconds

NAME      READY   UP-TO-DATE   AVAILABLE   AGE
cn-demo   1/1     1            1           2m17s
```

Other varieties (**`kubectl help`** to see all commands)

```
> kubectl get …
> kubectl edit …
> kubectl delete …
```

Lab 4 Let's DO it Together

- We'll walk through Lab4 together!
- "Kubernetes Infrastructure as Code"

Kubernetes API Server

GET https://kubernetes.docker.internal:6443/apis/extensions/v1beta1/namespaces/default/deployments/cn-demo

```
> kubectl get deployments cn-demo -o yaml

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  labels:
    run: cn-demo
  name: cn-demo
  namespace: default
spec:
  …
  …
  …
```

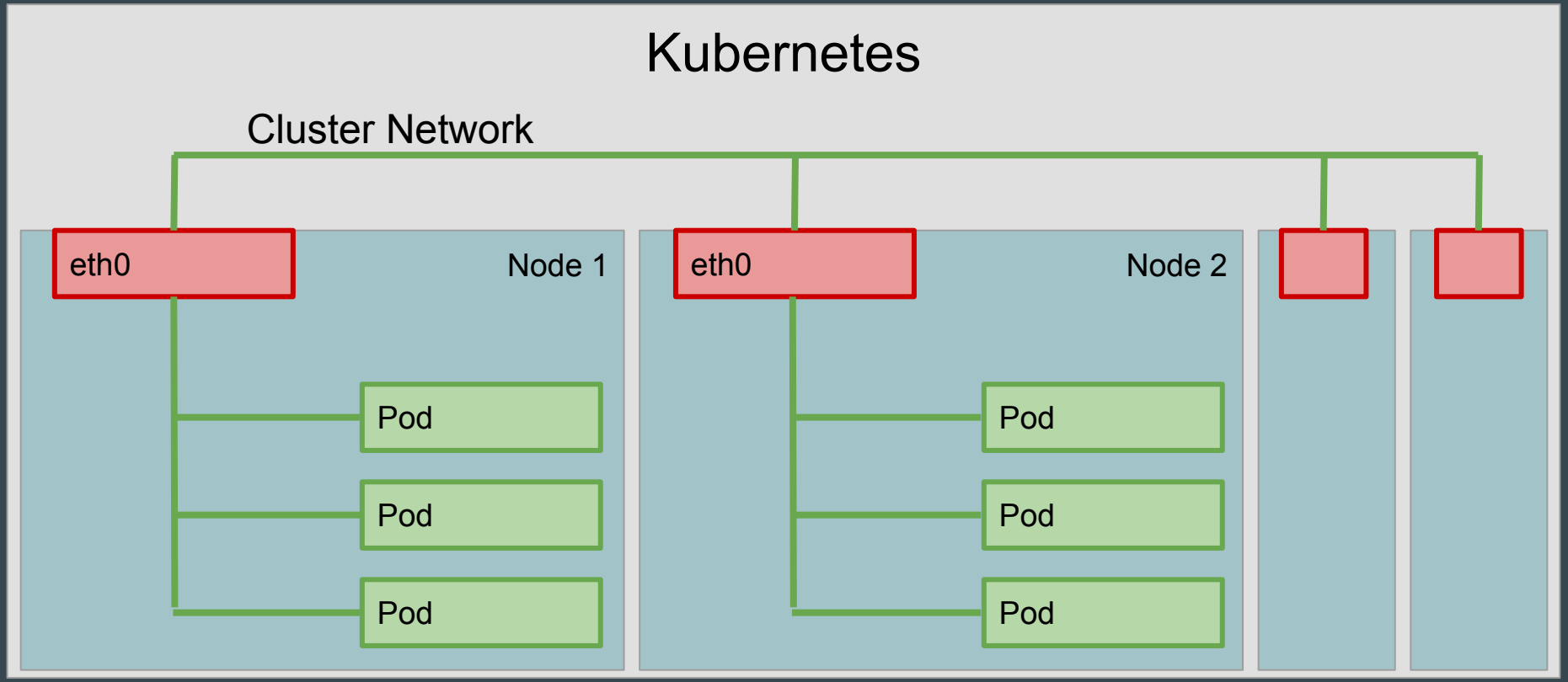
```
> kubectl get deployments cn-demo -o json

{
    "apiVersion": "extensions/v1beta1",
    "kind": "Deployment",
    "metadata": {
        "annotations": {
           "deployment.kubernetes.io/revision": "1"
        },
        "labels": {
            "run": "cn-demo"
        },
        "name": "cn-demo",
        "namespace": "default",
    },
    "spec": …
}
```

Kubernetes Networking

- Why the need for “kubectl port-forward”?

- Kubernetes Service Types
  - ClusterIP
  - NodePort
  - LoadBalancer



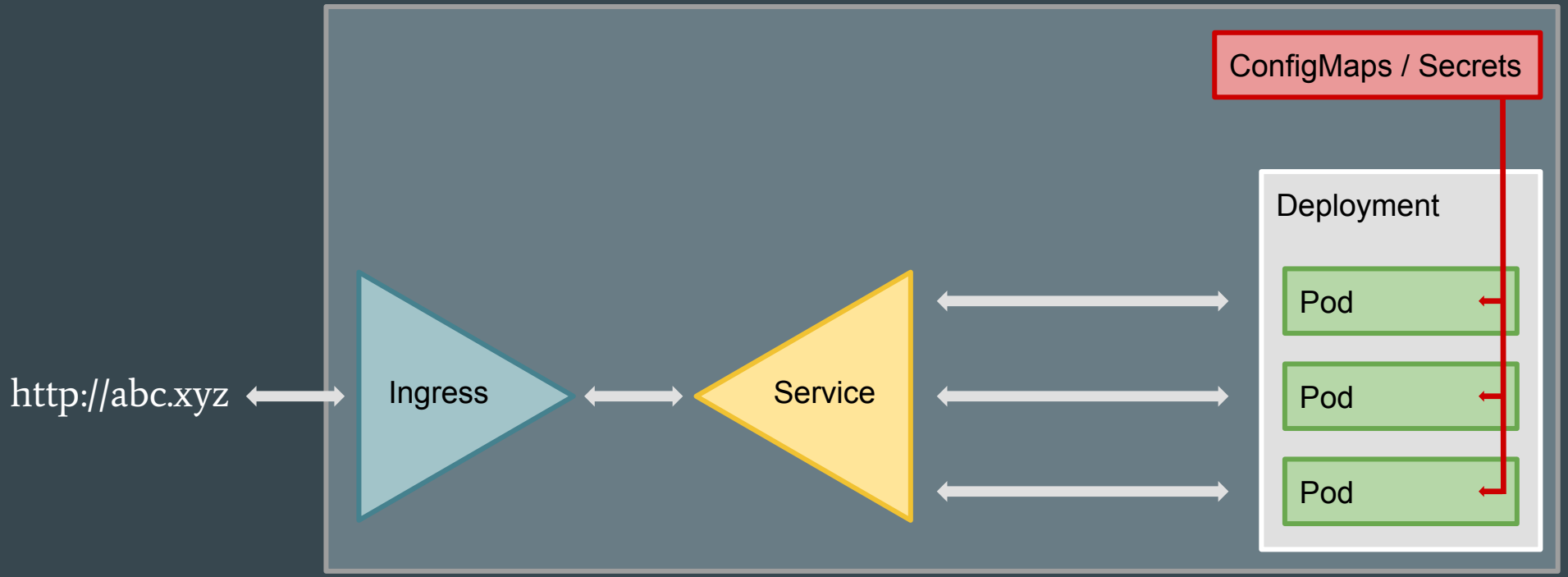
Lab 5 & 6 & 7

- Do Lab 5
- “Docker for Tooling”
- Lab 6
- “Kubernetes Environment Variables”
- Lab 7
- “Kubernetes Override Starting Command”

Kubernetes Services and Ingress

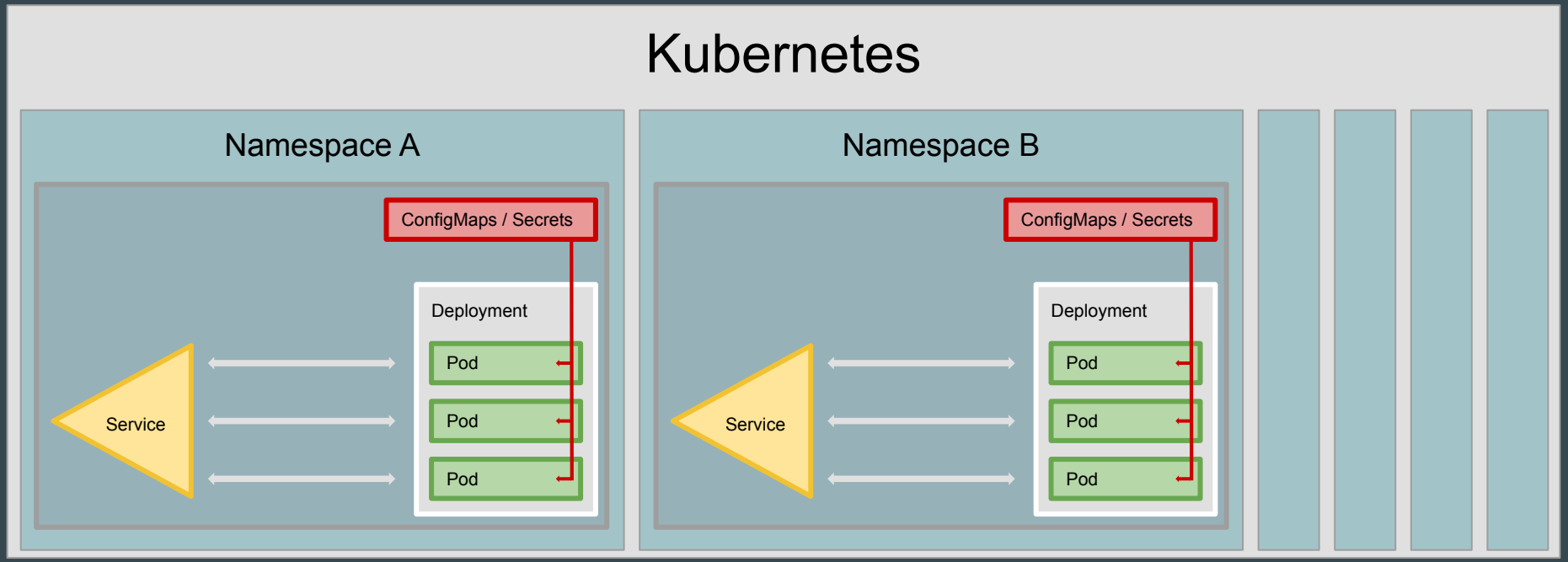
The big picture ...

- “Ingress” maps HTTP requests to Services
- “Services” load balance network traffic across Pods
- “Pods” define how to run your containers
- “Deployments” make sure your desired number of “Pods” are always running
- Configuration is stored in ConfigMaps and Secrets, and made available to the container



The big picture ... Namespaces

- Resource Quotas
  - CPU / Memory
  - Object counts
- Resource Isolation
  - Naming
  - Scheduling
- Security
  - RBAC
  - Network Policy



Lab 8

- Do Lab 8
- “Kubernetes Ingress”

Service to Pod selector

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: ClusterIP
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
```

Service to Pod selector

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  ...
  ...
  selector:
    run: nginx
    version: 1.0
  type: ClusterIP
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
    version: 1.0
  name: nginx
spec:
  containers:
  - image: nginx:1.16.0
  ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
    version: 2.0
  name: nginx
spec:
  containers:
  - image: nginx:1.17.2
  ...
```

Service to Pod selector

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  ...
  ...
  selector:
    run: nginx
    version: 2.0
  type: ClusterIP
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
    version: 1.0
  name: nginx
spec:
  containers:
  - image: nginx:1.16.0
  ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
    version: 2.0
  name: nginx
spec:
  containers:
  - image: nginx:1.17.2
  ...
```

Ingress to Service selector

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: blue-green
spec:
  rules:
  - host: blue-green.127.0.0.1.xip.io
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx
          servicePort: 80
```

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: ClusterIP
```

Labs 9 & 10

- Do Lab 9
- “Kubernetes Readiness”
-
- Do Lab 9
- “Kubernetes Readiness Part 2”

Where to go next

- Application Configuration
  - Environment Variables
  - ConfigMaps
  - Secrets
- Other resources
  - Docker Desktop
    - Simple to provision local Kubernetes environment
  - https://github.com/frayer/kubernetes-patterns
  - https://www.katacoda.com
  - Udemy CKAD Course is excellent!
    - https://www.udemy.com/course/certified-kubernetes-application-developer/
- Learn by doing

Final thoughts

- Stateful application workloads
  - Not impossible, but the techniques needed were not covered here
  - Watch out for data workloads that need privileged level access to kernel

Thank you!

Michael Frayer - @frayerm
Mark Ramsey - @mramsey24
Barry Tarlton - btarlton@gmail.com

http://github.com/javaplus/DockerKubesDojo