Orchestrating Containers

Kubernetes Basics

Eueung Mulyana

http://eueung.github.io/docker-stuff/kubernetes

CodeLabs |

Outline

Kubernetes

Short Introduction

What is Kubernetes?

The name Kubernetes originates from Greek, meaning "helmsman" or "pilot", and is the root of "governor" and "cybernetic".

K8s is an abbreviation derived by replacing the 8 letters "ubernete" with 8.

With Kubernetes you can deploy a full cluster of **multi-tiered** containers (frontend, backend, etc.) with a **single** configuration file and a **single** command (Ref).

Kubernetes is an open-source platform for **automating** deployment, scaling, and operations of **application containers** across clusters of hosts, providing container-centric infrastructure.

With Kubernetes, you are able to quickly and efficiently respond to customer demand:

- **Deploy** your applications quickly and predictably.
- **Scale** your applications on the fly.
- Seamlessly **roll out** new features.
- **Optimize** use of your hardware by using only the resources you need

Kubernetes is:

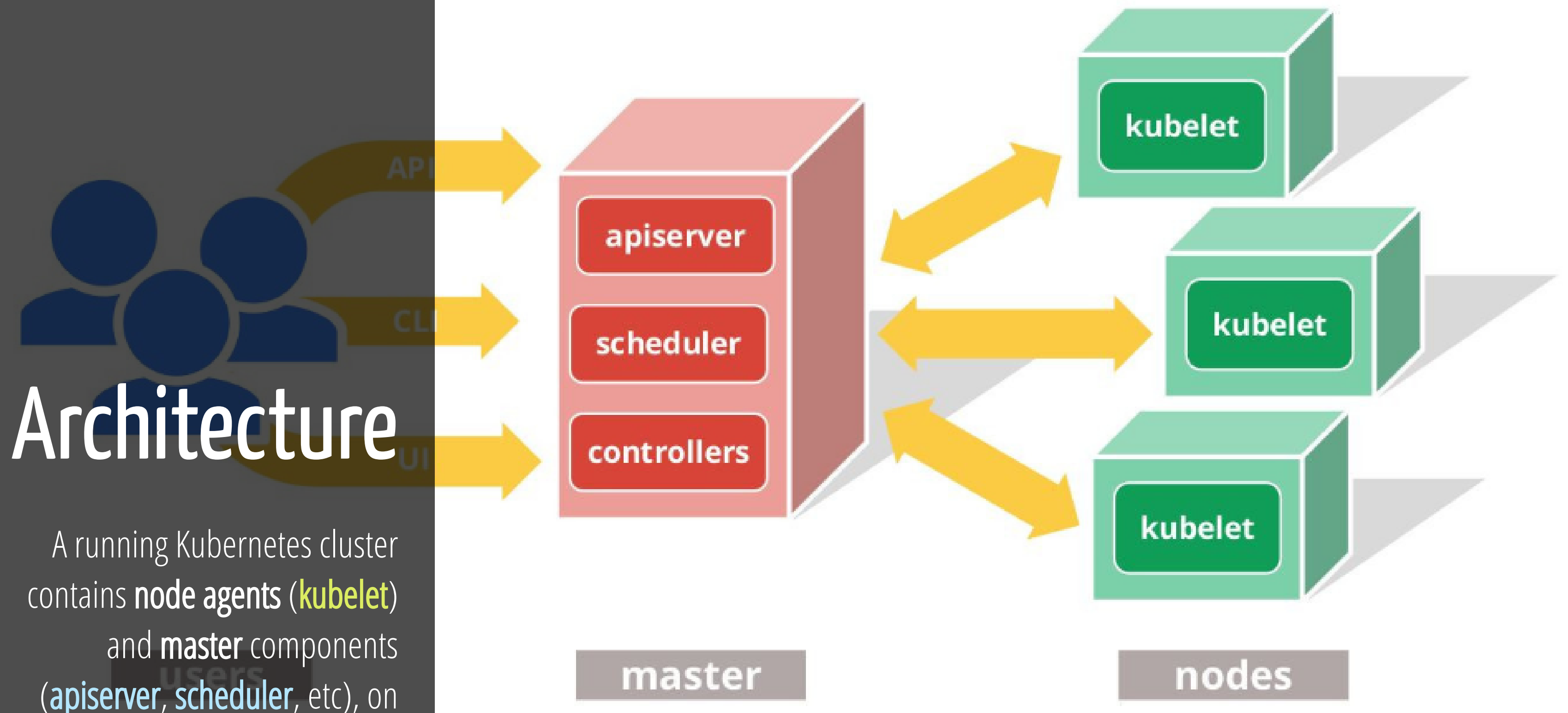
- **portable**: public, private, hybrid, multi-cloud
- **extensible**: modular, pluggable, hookable, composable
- **self-healing**: auto-placement, auto-restart, auto-replication, auto-scaling

Ref: kubernetes.io

Kubernetes

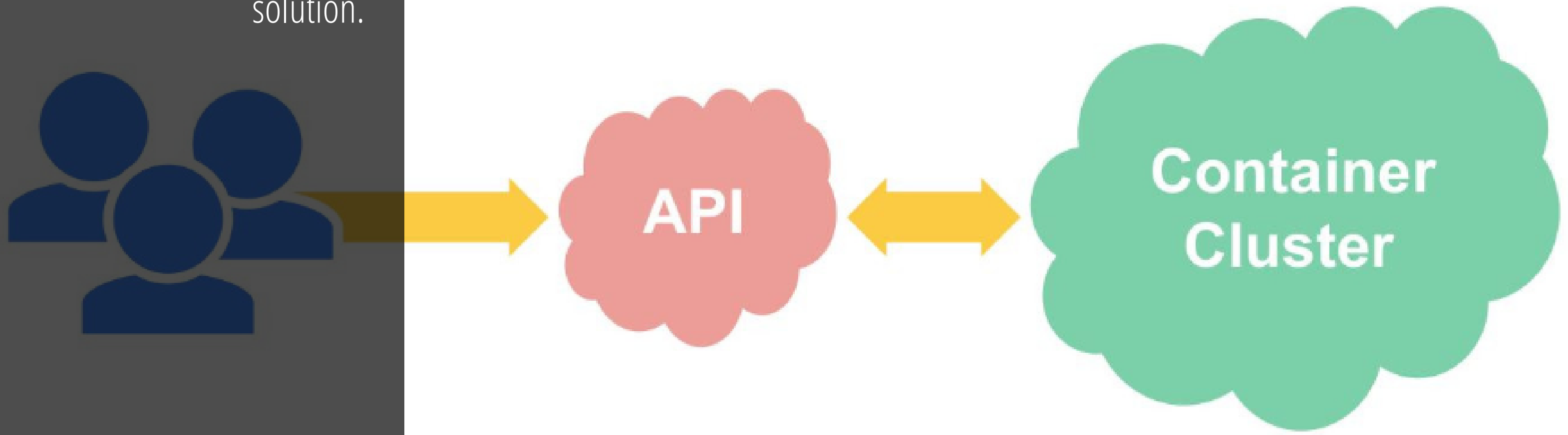
- Container orchestrator
- Runs and manages containers
- Supports multiple cloud and bare-metal environments
- Inspired and informed by Google's experiences and internal systems
- 100% Open source, written in Go
- Manage applications, not machines

Ref: Kubernetes Intro and Update @thockin



Architecture

A running Kubernetes cluster contains **node agents** (kubelet) and **master** components (apiserver, scheduler, etc), on top of a distributed storage solution.



K8s Node

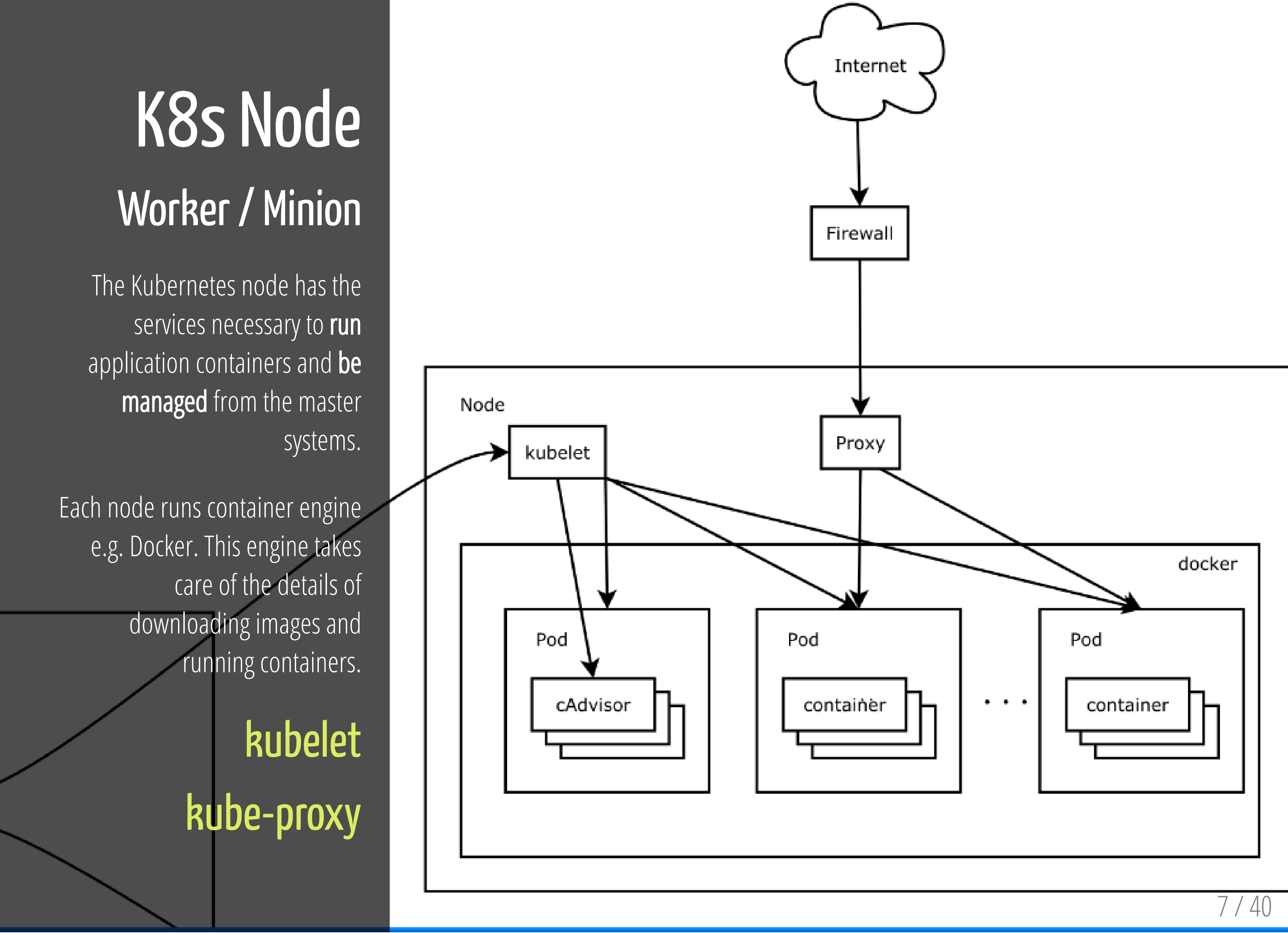
Worker / Minion

The Kubernetes node has the services necessary to **run** application containers and **be managed** from the master systems.

Each node runs container engine e.g. Docker. This engine takes care of the details of downloading images and running containers.

kubelet

kube-proxy



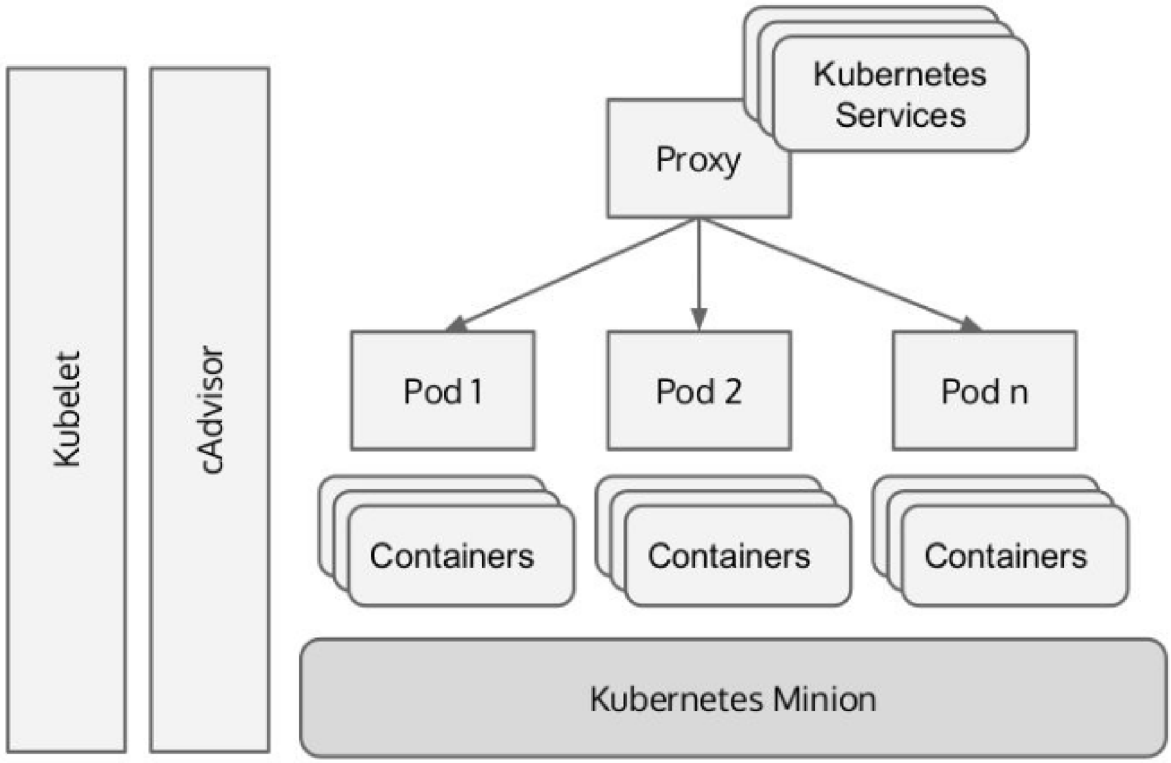
K8s Node

kubelet

The kubelet manages **pods** and their containers, their images, their volumes, etc.

kube-proxy

Each node also runs a simple network proxy and load balancer. This reflects **services** as defined in the Kubernetes API (apiserver) on each node and can do simple TCP and UDP stream forwarding (round robin) across a set of backends.



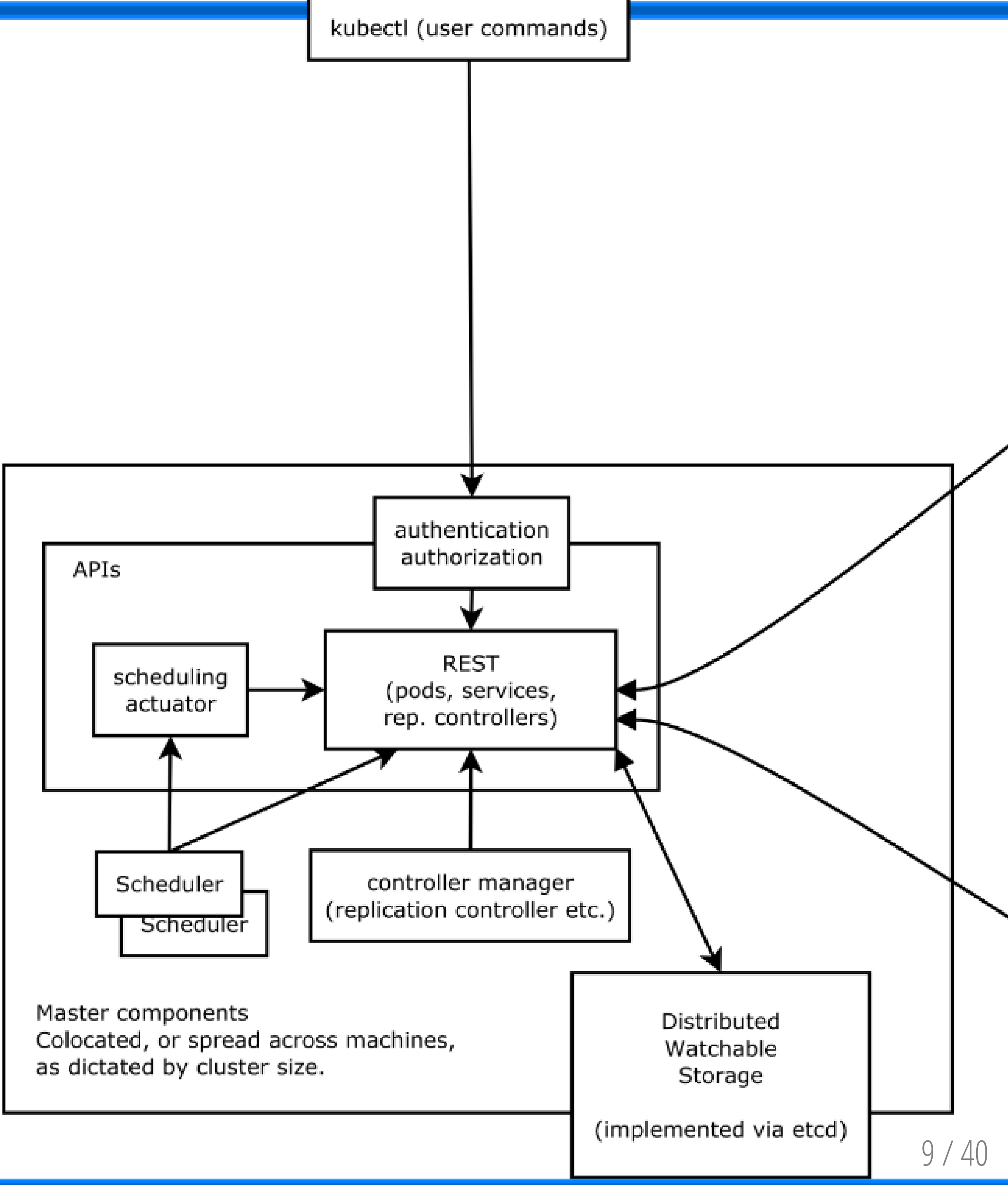
K8s Master

Control Plane

The Kubernetes control plane is split into a set of components. Currently they all run on a single master node. These components work together to provide a unified view of the cluster.

etcd

All persistent master state is stored in an instance of etcd. This provides a great way to store configuration data reliably. With watch support, coordinating components can be notified very quickly of changes.

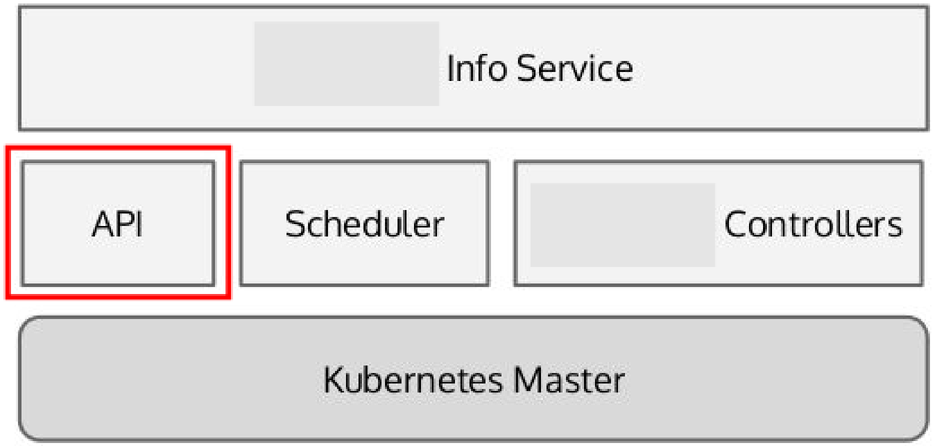

kubectl (user commands)
authentication authorization
APIs
scheduling actuator
REST (pods, services, rep. controllers)
Scheduler
Scheduler
controller manager (replication controller etc.)
Master components
Colocated, or spread across machines,
as dictated by cluster size.
Distributed Watchable Storage
(implemented via etcd)

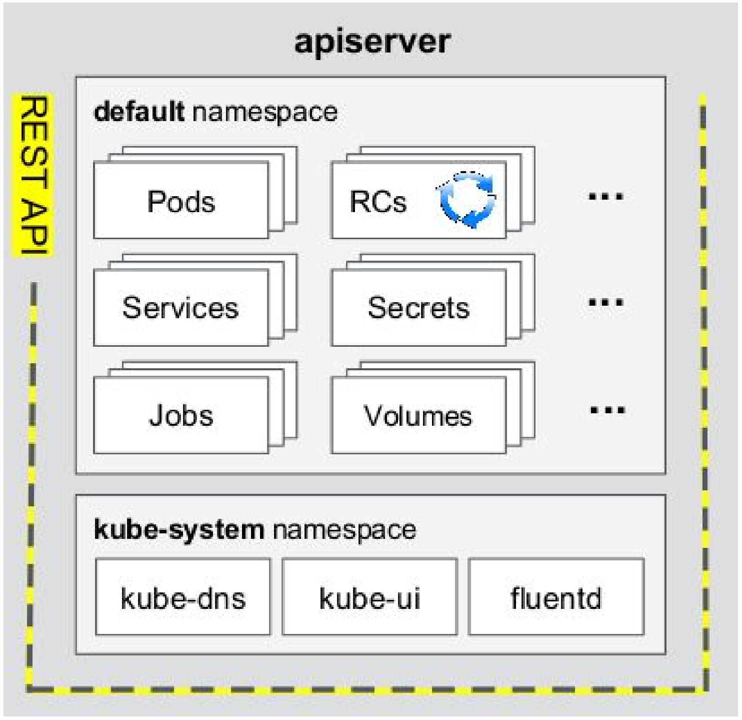
K8s Master

etcd

API Server

The apiserver serves up the Kubernetes API. It is intended to be a CRUD-y server, with most/all business logic implemented in separate components or in plug-ins. It mainly processes REST operations, validates them, and updates the corresponding objects in etcd (and eventually other stores).





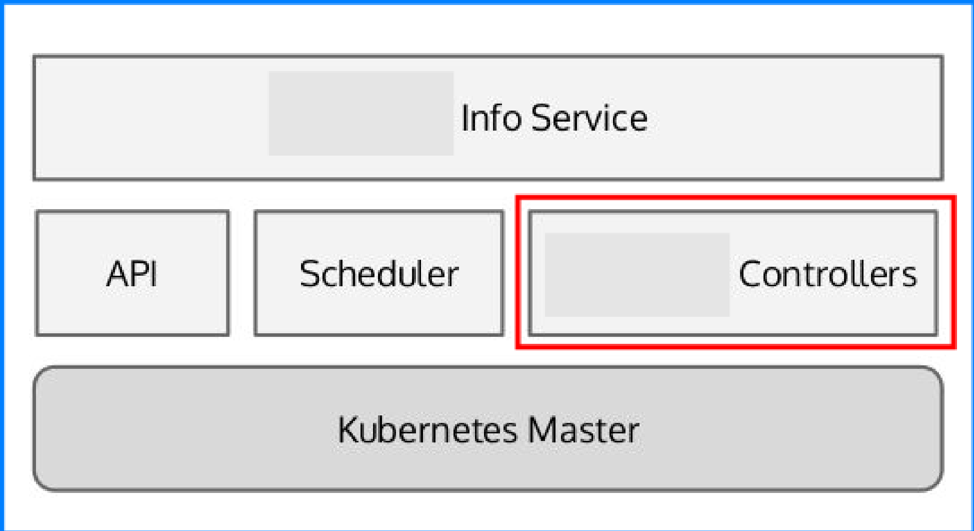
Scheduler

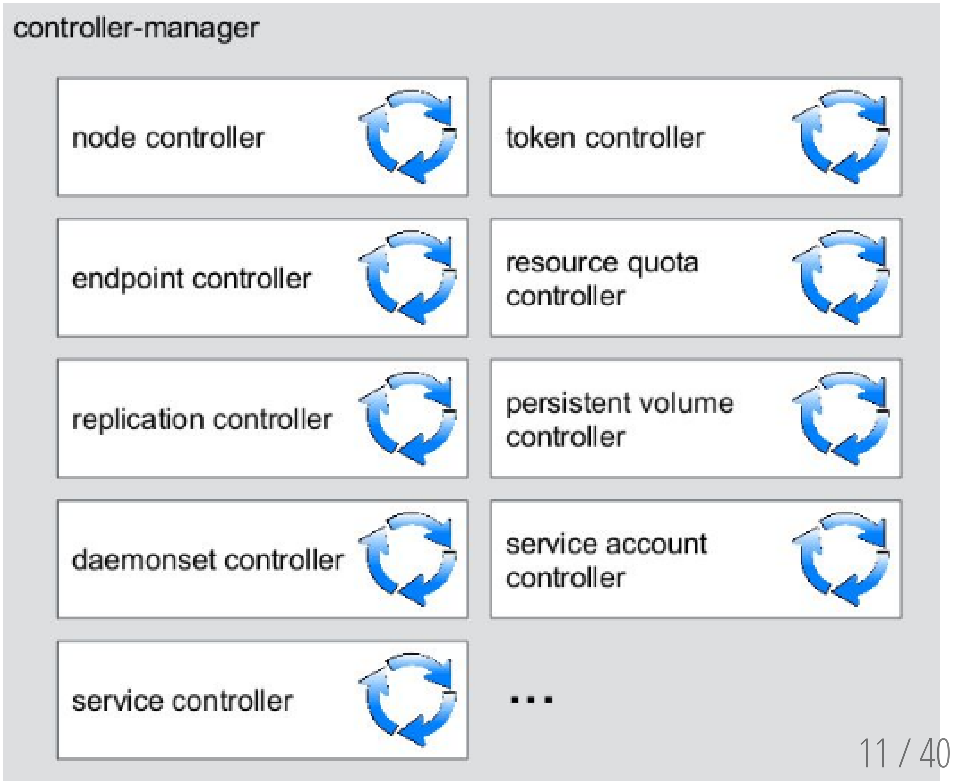
The scheduler binds unscheduled pods to nodes via the **/binding** API. The scheduler is pluggable, support for multiple cluster schedulers and user-provided schedulers in the plan.

Controller Manager

All other cluster-level functions are currently performed by the Controller Manager.

For instance, Endpoints objects are created and updated by the **endpoints controller**, and nodes are discovered, managed, and monitored by the **node controller**. These could eventually be split into separate components to make them independently pluggable. The **replicationcontroller** is a mechanism that is layered on top of the simple pod API.



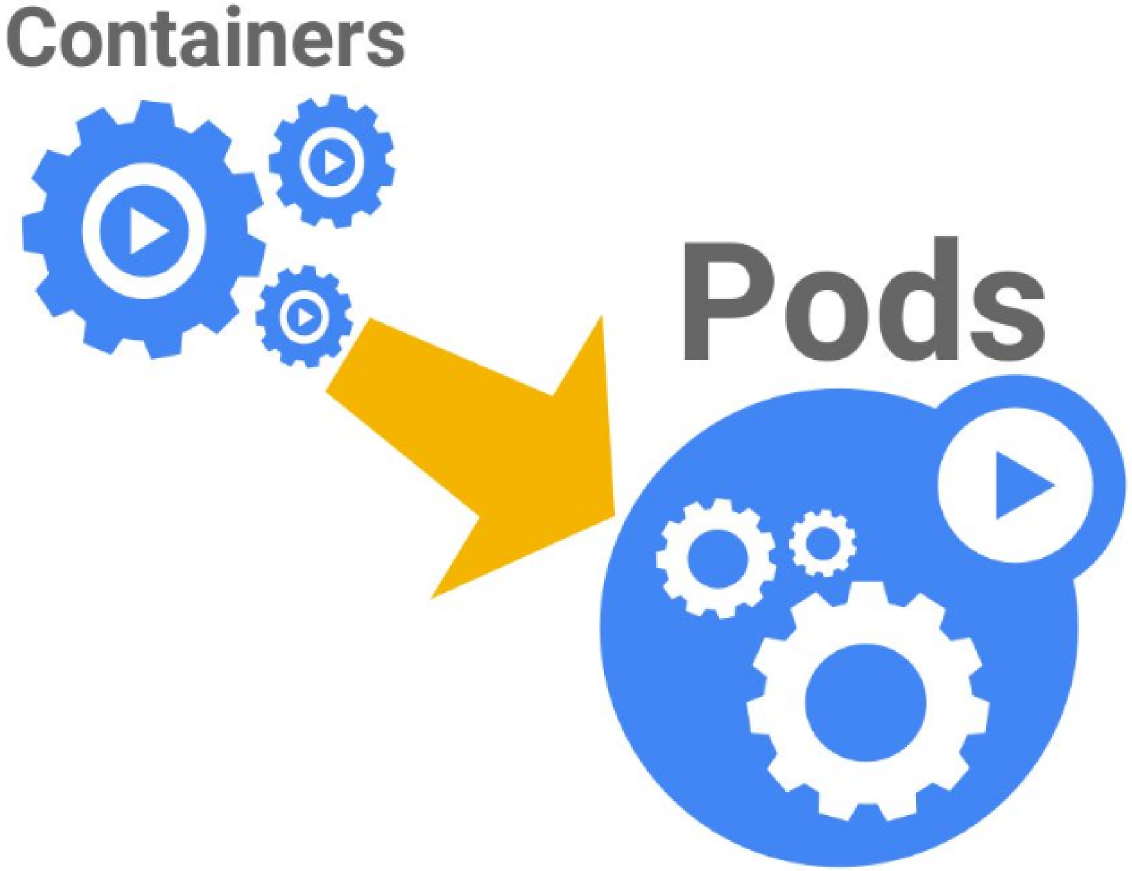


Kubernetes

Parts & Components

Containers

Base Asset



Containers

Pods

A pod is a co-located group of containers ...

Containers

Pods

Pods with **Volumes**

A pod is a co-located group of containers and volumes.

A volume is a directory, possibly with some data in it, which is accessible to a Container as part of its filesystem.

Kubernetes volumes build upon Docker Volumes, adding provisioning of the volume directory and/or device.

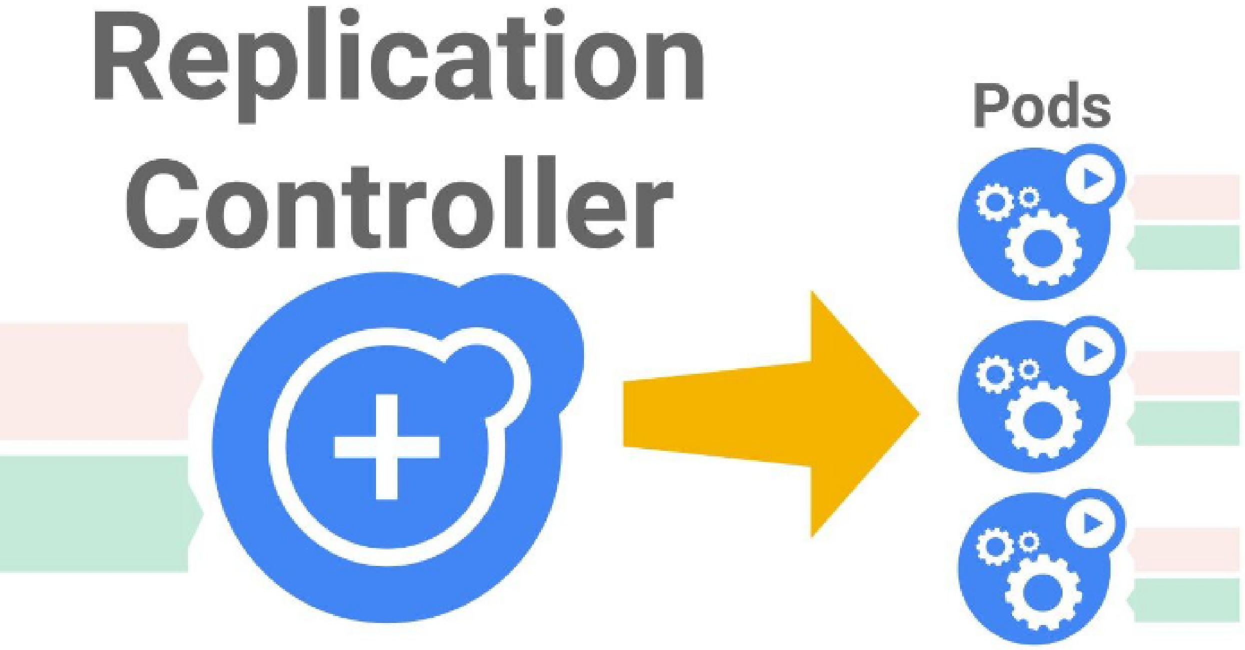
A label is a key/value pair that is attached to a resource, such as a pod, to convey a user-defined identifying attribute.

Labels can be used to organize and to select subsets of resources.

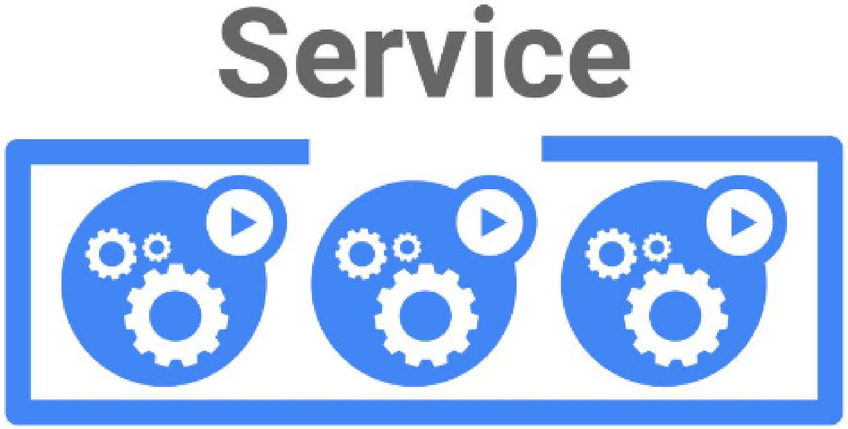
Replication Controller

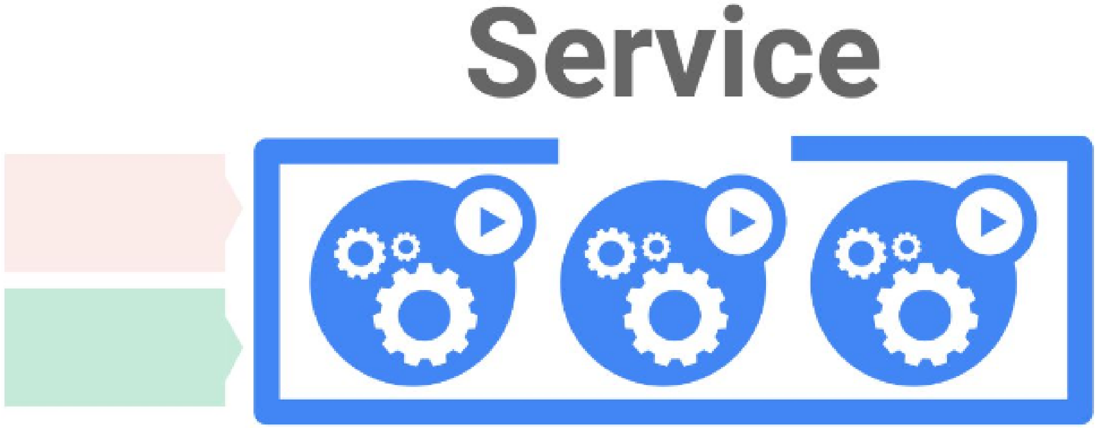
A replication controller ensures that a specified number of pod replicas are running at any one time.

It both allows for easy scaling of replicated systems and handles re-creation of a pod when the machine it is on reboots or otherwise fails.



Creating labeled pods with a (labeled) RC

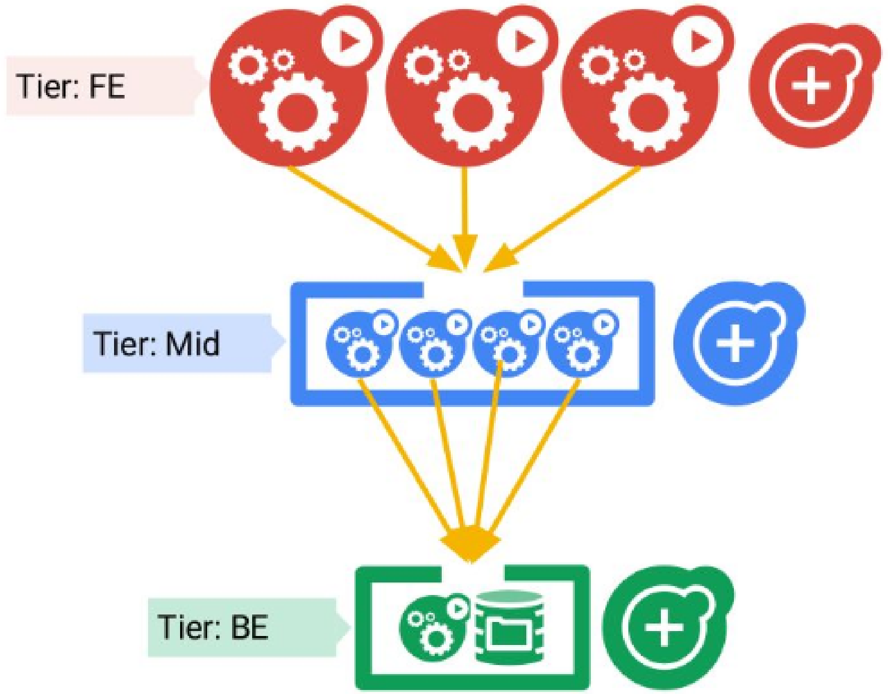




Services & labeled Services

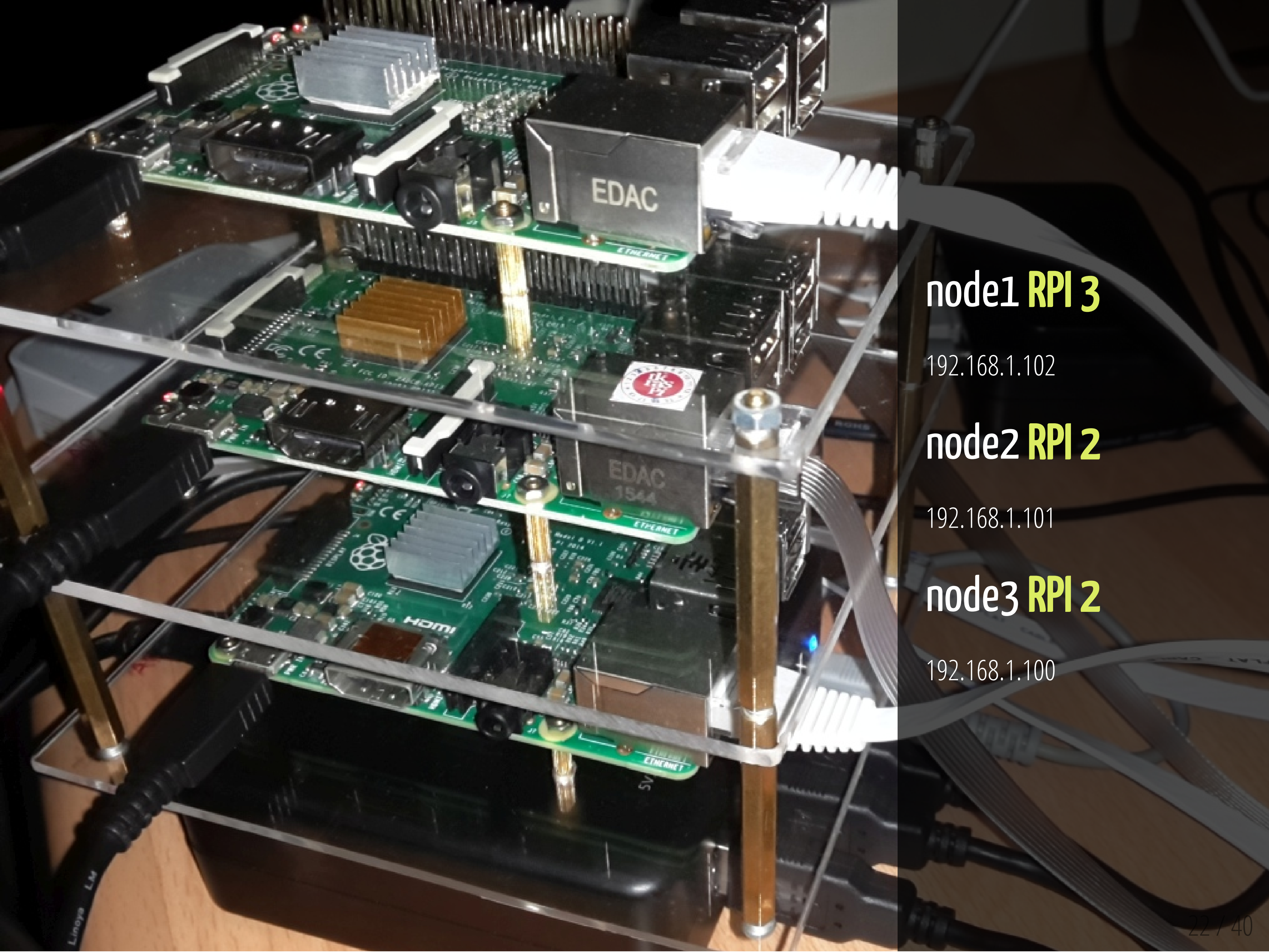
A service defines a set of pods and a means by which to access them, such as single stable IP address and corresponding DNS name.

Put it all together



Kubernetes

Getting Started



node1 RPI 3

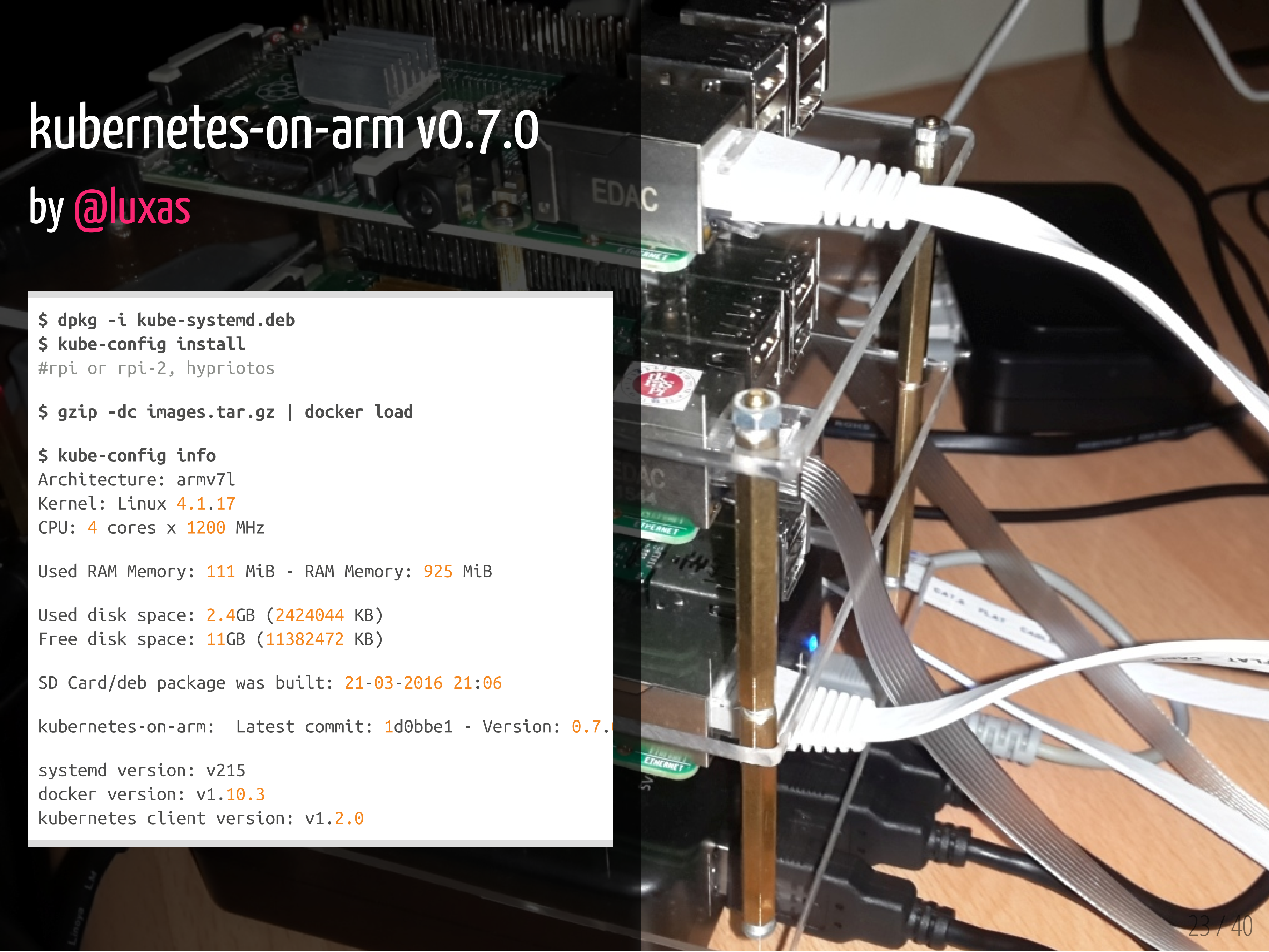
192.168.1.102

node2 RPI 2

192.168.1.101

node3 RPI 2

192.168.1.100



kubernetes-on-arm v0.7.0

by @luxas

```
$ dpkg -i kube-systemd.deb
$ kube-config install
#rpi or rpi-2, hypriotos

$ gzip -dc images.tar.gz | docker load

$ kube-config info
Architecture: armv7l
Kernel: Linux 4.1.17
CPU: 4 cores x 1200 MHz

Used RAM Memory: 111 MiB - RAM Memory: 925 MiB

Used disk space: 2.4GB (2424044 KB)
Free disk space: 11GB (11382472 KB)

SD Card/deb package was built: 21-03-2016 21:06

kubernetes-on-arm:  Latest commit: 1d0bbe1 - Version: 0.7.

systemd version: v215
docker version: v1.10.3
kubernetes client version: v1.2.0
```

```
$ docker images
REPOSITORY                   TAG           IMAGE ID        CREATED
kubernetesonarm/etcd         0.7.0         3550c0c4f205    8 weeks ago
kubernetesonarm/etcd         latest        3550c0c4f205    8 weeks ago
kubernetesonarm/grafana      0.7.0         8ea26f4ef5a3    8 weeks ago
kubernetesonarm/grafana      latest        8ea26f4ef5a3    8 weeks ago
kubernetesonarm/influxdb     0.7.0         447951a687c6    8 weeks ago
kubernetesonarm/influxdb     latest        447951a687c6    8 weeks ago
kubernetesonarm/heapster     0.7.0         1d034674fc0e    8 weeks ago
kubernetesonarm/heapster     latest        1d034674fc0e    8 weeks ago
kubernetesonarm/loadbalancer 0.7.0         6b262f2fd318    8 weeks ago
kubernetesonarm/loadbalancer latest        6b262f2fd318    8 weeks ago
kubernetesonarm/registry     0.7.0         4c9c964f89e9    8 weeks ago
kubernetesonarm/registry     latest        4c9c964f89e9    8 weeks ago
kubernetesonarm/exechealthz  0.7.0         57f77542ac5c    8 weeks ago
kubernetesonarm/exechealthz  latest        57f77542ac5c    8 weeks ago
kubernetesonarm/kube2sky     0.7.0         770cccac4236    8 weeks ago
kubernetesonarm/kube2sky     latest        770cccac4236    8 weeks ago
kubernetesonarm/skydns       0.7.0         30ea4958f939    8 weeks ago
kubernetesonarm/skydns       latest        30ea4958f939    8 weeks ago
kubernetesonarm/pause        0.7.0         05edc969256e    8 weeks ago
kubernetesonarm/pause        latest        05edc969256e    8 weeks ago
kubernetesonarm/hyperkube    0.7.0         380def049467    8 weeks ago
kubernetesonarm/hyperkube    latest        380def049467    8 weeks ago
kubernetesonarm/flannel      0.7.0         8a35c629399f    8 weeks ago
kubernetesonarm/flannel      latest        8a35c629399f    8 weeks ago
hypriot/rpi-swarm            latest        c298de062190    10 weeks ago
```

Master

```
$ kube-config enable-master
Disabling k8s if it is running
Checks so all images are present
Transferring images to system-docker, if necessary
Copying kubernetesonarm/etcd to system-docker
...
Kubernetes master services enabled

$ docker ps
CONTAINER ID        IMAGE                         COMMAND                  CREATED
7bf736d84451        kubernetesonarm/hyperkube     "/hyperkube controlle"   2 minutes a
5560f06bcf47        kubernetesonarm/hyperkube     "/hyperkube proxy --m"   2 minutes a
0fb169567946        kubernetesonarm/hyperkube     "/hyperkube scheduler"   2 minutes a
11c03a40412c        kubernetesonarm/hyperkube     "/hyperkube apiserver"   2 minutes a
ede858b35dfa        kubernetesonarm/pause         "/pause"                 2 minutes a
5e38dac4fd19        kubernetesonarm/hyperkube     "/hyperkube kubelet -"   3 minutes a
```

Worker

```
$ kube-config enable-worker 192.168.1.102
Disabling k8s if it is running
Using master ip: 192.168.1.102
Checks so all images are present
Transferring images to system-docker, if necessary
Copying kubernetesonarm/flannel to system-docker
Created symlink from /etc/systemd/system/multi-user.target.wants/flannel.service to
Starting worker components in docker containers
Created symlink from /etc/systemd/system/multi-user.target.wants/k8s-worker.service
Kubernetes worker services enabled
```

Check

```
$ kubectl get nodes
NAME             STATUS    AGE
192.168.1.100    Ready     26s
192.168.1.101    Ready     7m
192.168.1.102    Ready     4h

$ kubectl cluster-info
Kubernetes master is running at http://localhost:8080
```

Test #1

```
$ kubectl run my-nginx --image=luxas/nginx-test  --replicas=3 --expose --port=80
service "my-nginx" created
deployment "my-nginx" created

$ kubectl get pods
NAME                            READY     STATUS              RESTARTS    AGE
k8s-master-192.168.1.102        4/4       Running             1           5h
my-nginx-3795026575-9w8mw       0/1       ContainerCreating   0           14s
my-nginx-3795026575-miz3d       0/1       ContainerCreating   0           14s
my-nginx-3795026575-zy2d4       0/1       ContainerCreating   0           14s

$ kubectl get svc
NAME          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes    10.0.0.1      <none>         443/TCP    5h
my-nginx      10.0.0.129    <none>         80/TCP     36s

$ kubectl get deployments
NAME         DESIRED     CURRENT     UP-TO-DATE     AVAILABLE     AGE
my-nginx     3           3           3              1             1m

$ curl 10.0.0.129
<p>WELCOME TO NGINX</p>
```

Test #2

```
$ kubectl run hello-kube --image=hypriot/rpi-nano-httpd  --port=80
deployment "hello-kube" created

$ kubectl expose deployment hello-kube --type="LoadBalancer" --external-ip="192.168.

service "hello-kube" exposed

$ kubectl get pods -o wide
NAME                              READY     STATUS             RESTARTS    AGE       NODE
hello-kube-1079346743-2knj3       0/1       ImagePullBackOff   0           55s       192.
k8s-master-192.168.1.102          4/4       Running            1           8h        192.

$ kubectl get deployments
NAME          DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
hello-kube    1          1          1             1            1m

$ kubectl get pods -o wide
NAME                              READY     STATUS     RESTARTS    AGE       NODE
hello-kube-1079346743-2knj3       1/1       Running    0           1m        192.168.1.100
k8s-master-192.168.1.102          4/4       Running    1           8h        192.168.1.102

$ kubectl get svc
NAME          CLUSTER-IP    EXTERNAL-IP       PORT(S)    AGE
hello-kube    10.0.0.197    ,192.168.1.102    80/TCP     44s
kubernetes    10.0.0.1      <none>            443/TCP    8h

$ curl 10.0.0.197
<html><head><title>Pi armed with Docker by Hypriot</title>
  <body style="width: 100%; background-color: black;">
    <div id="main" style="margin: 100px auto 0 auto; width: 800px;">
      <img src="pi_armed_with_docker.jpg" alt="pi armed with docker" style="width: 8
    </div>
</body></html>
```

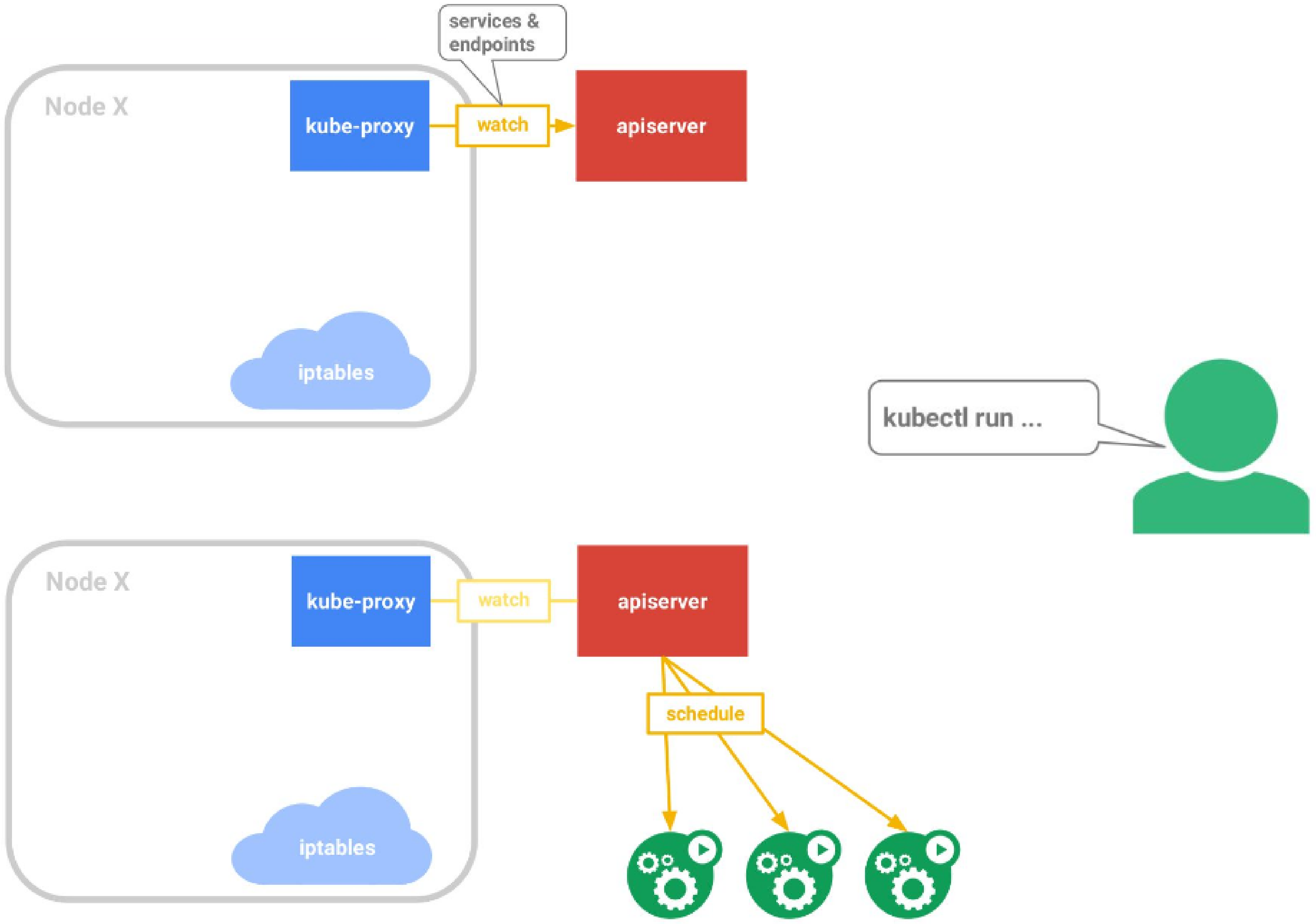
Test #3

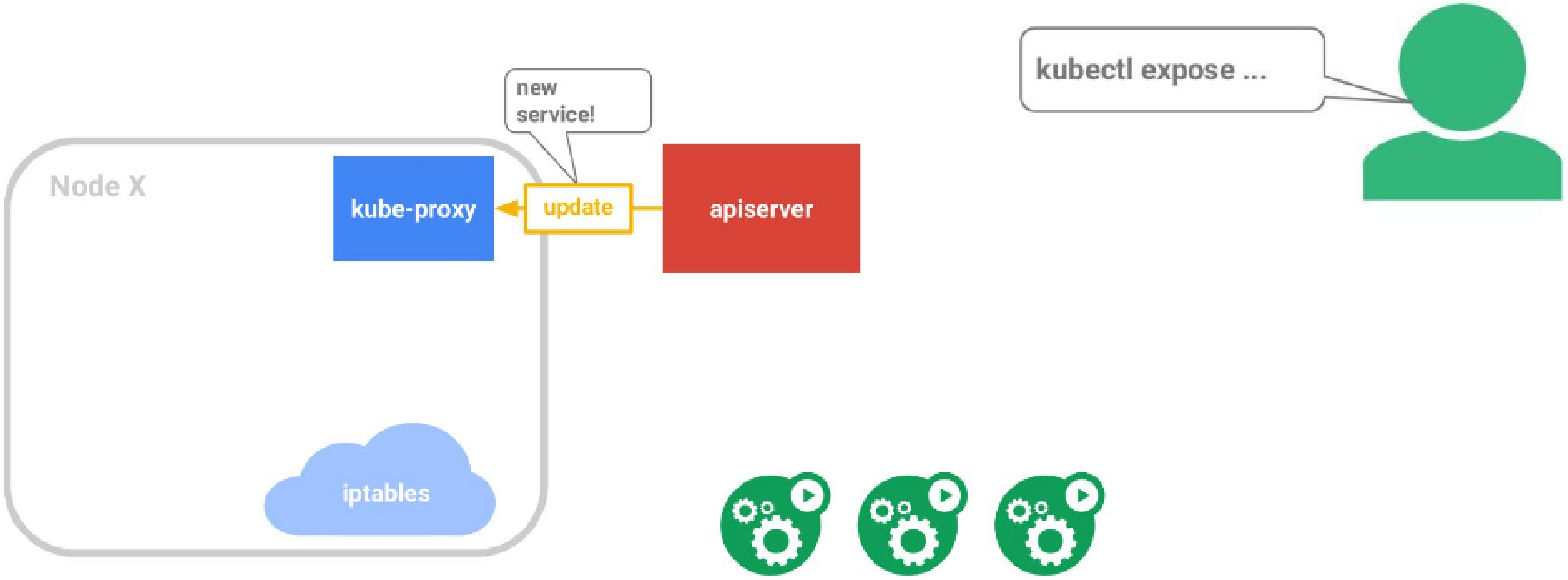
```
$ kubectl run hello-kube --image=hypriot/rpi-nano-httpd  --replicas=2 --port=80
deployment "hello-kube" created

$ kubectl expose deployment hello-kube --type="LoadBalancer" --external-ip="192.168.
service "hello-kube" exposed

$ kubectl get svc
NAME         CLUSTER-IP    EXTERNAL-IP       PORT(S)    AGE
hello-kube   10.0.0.221    ,192.168.1.102    80/TCP     10s
kubernetes   10.0.0.1      <none>            443/TCP    8h

$ kubectl get pods -o wide
NAME                           READY     STATUS    RESTARTS    AGE       NODE
hello-kube-1079346743-44yig    1/1       Running   0           1m        192.168.1.100
hello-kube-1079346743-upzxy    1/1       Running   0           1m        192.168.1.101
k8s-master-192.168.1.102       4/4       Running   1           8h        192.168.1.102

$ curl 10.0.0.221

$ curl 192.168.1.102
<html><head><title>Pi armed with Docker by Hypriot</title>
  <body style="width: 100%; background-color: black;">
    <div id="main" style="margin: 100px auto 0 auto; width: 800px;">
      <img src="pi_armed_with_docker.jpg" alt="pi armed with docker" style="width: 8
    </div>
</body></html>
```

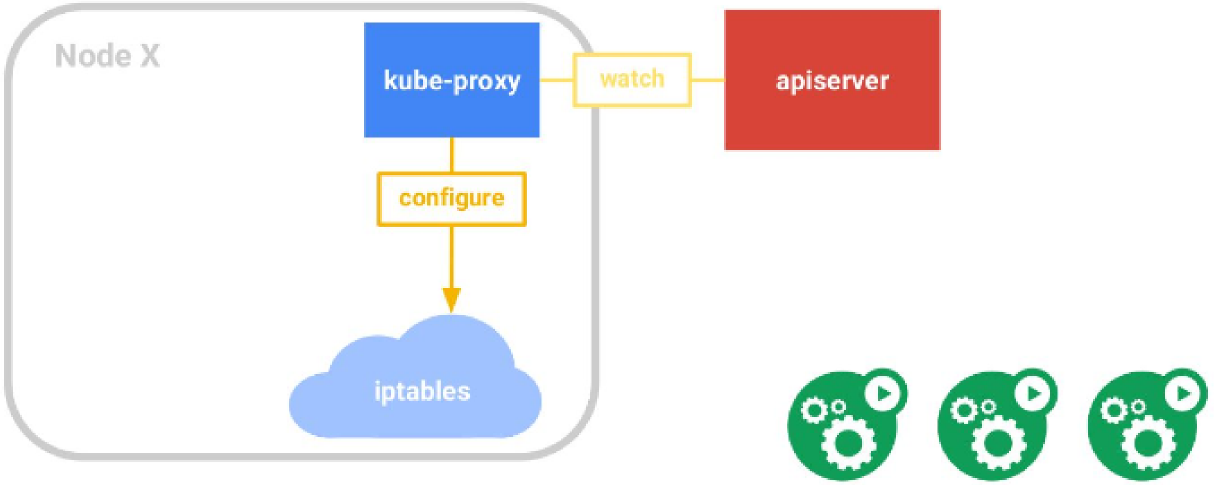
Test #4

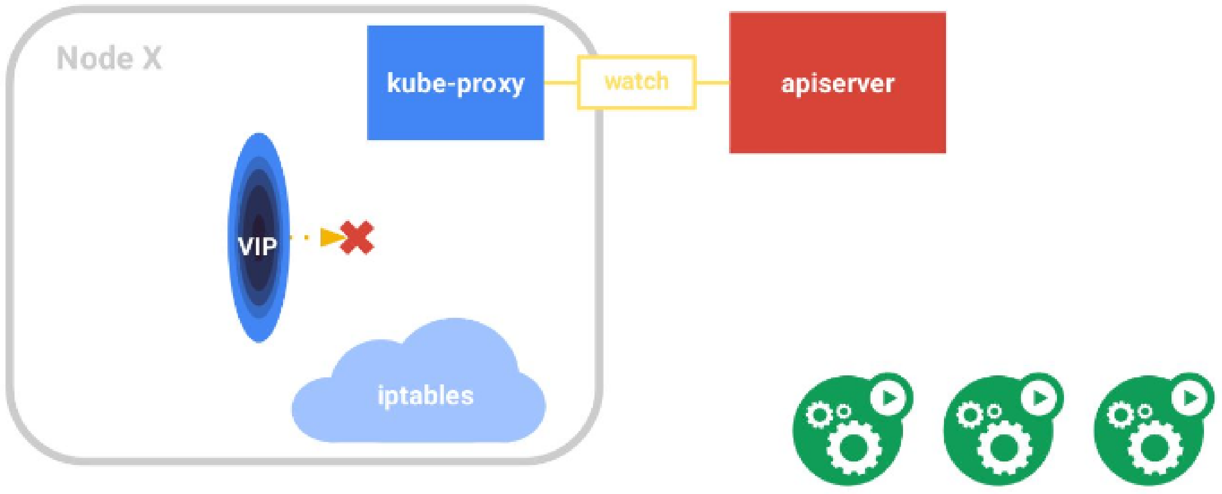
```
$ kubectl run hello-kube --image=hypriot/rpi-nano-httpd  --replicas=3 --port=80
deployment "hello-kube" created

$ kubectl expose deployment hello-kube --port=8300 --target-port=80 --type="LoadBala
service "hello-kube" exposed

$ kubectl get svc
NAME         CLUSTER-IP    EXTERNAL-IP     PORT(S)    AGE
hello-kube   10.0.0.124    ,192.168.1.102  8300/TCP   8s
kubernetes   10.0.0.1      <none>          443/TCP    7h

$ kubectl delete service,deployment hello-kube
```

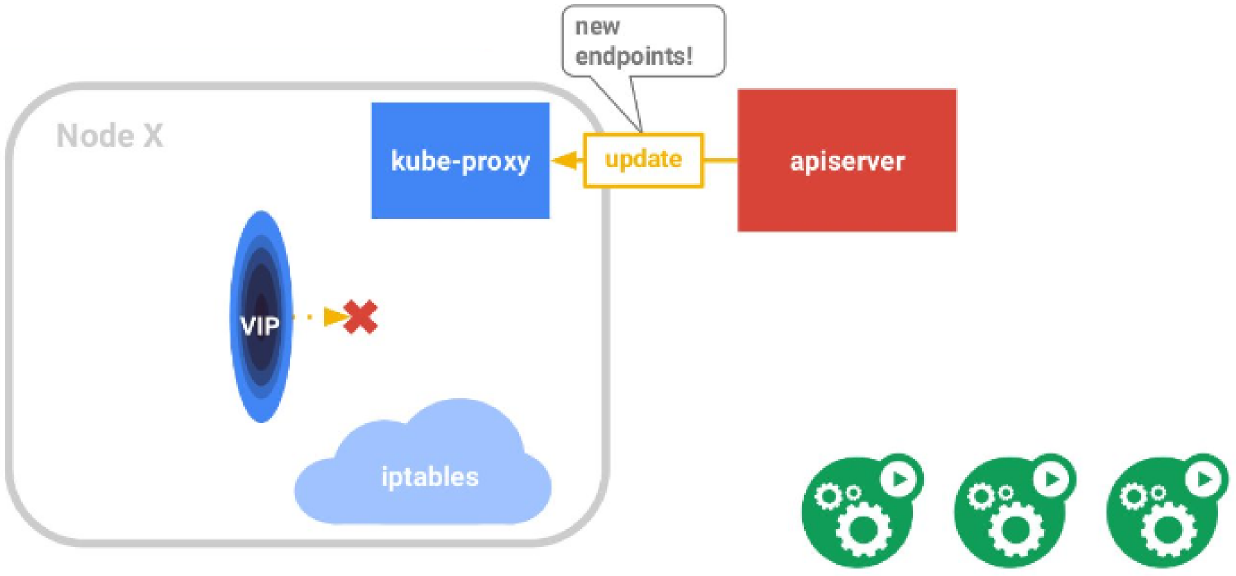
kubectl run + kubectl expose

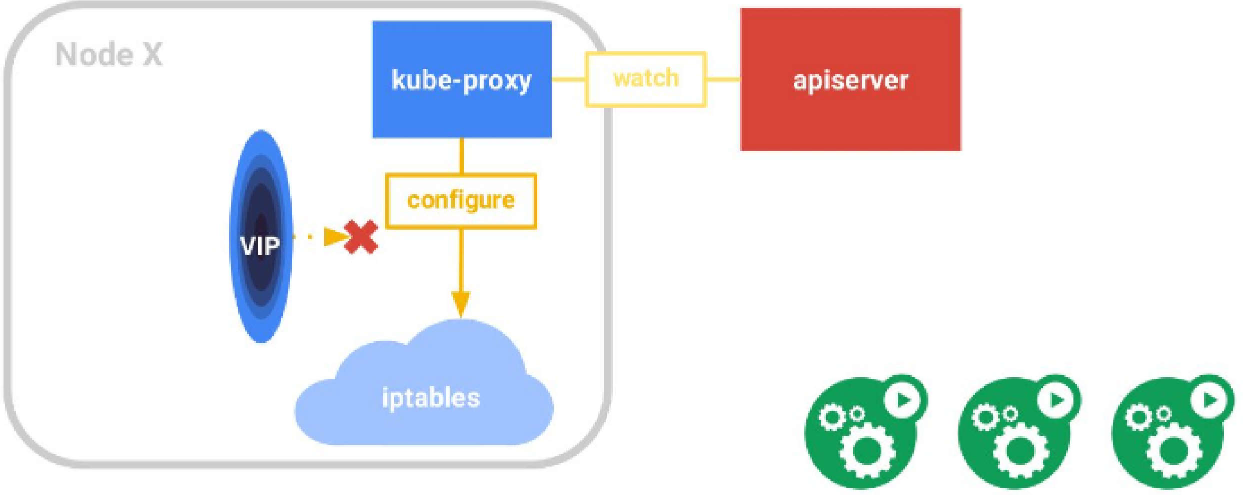

services & endpoints
Node X
kube-proxy
watch
apiserver
iptables
kubectl run ...
Node X
kube-proxy
watch
apiserver
schedule
iptables

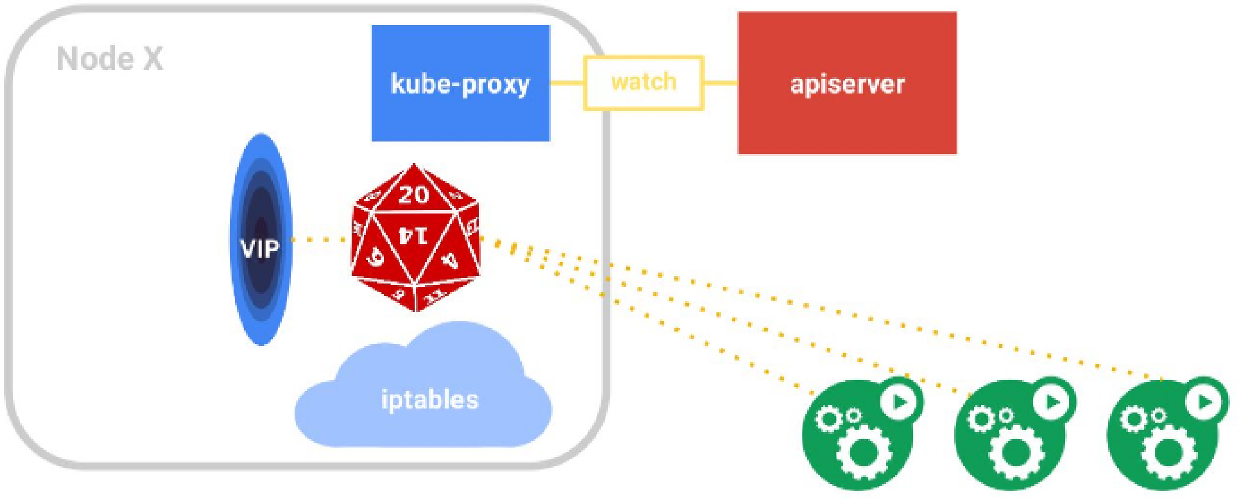

kubectl expose ...
new service!
Node X
kube-proxy
update
apiserver
iptables

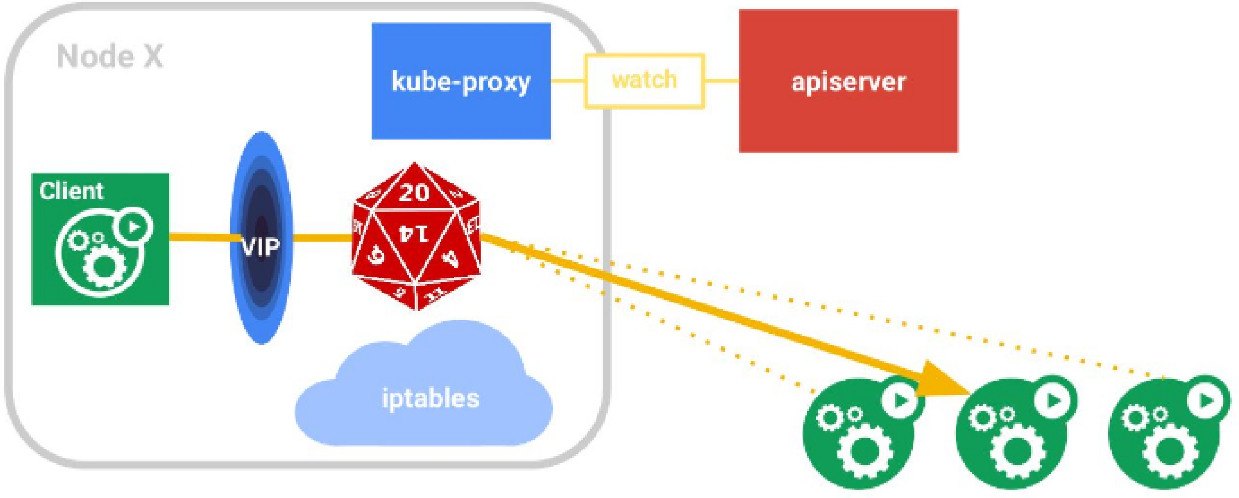

Node X
kube-proxy
watch
apiserver
configure
iptables


Node X
kube-proxy
watch
apiserver
VIP
iptables


new endpoints!
Node X
kube-proxy
update
apiserver
VIP
iptables


Node X
kube-proxy
watch
apiserver
configure
VIP
iptables


Node X
kube-proxy
watch
apiserver
VIP
iptables


Node X
kube-proxy
watch
apiserver
Client
VIP
iptables

Refs

Refs

1. Tim Hockin, Kubernetes: One Year Later
2. Ray Tsang @saturnism, Kubernetes with Java-based Microservices
3. Stefan Schimanski, Kubernetes Architecture & Introduction
4. @luxas - kubernetes-on-arm
5. Daniel Smith, What's new in Kubernetes
6. Kubernetes - What is Kubernetes?
7. Kubernetes - User Guide
8. Learn the Kubernetes Key Concepts
9. Kubernetes Intro and Update @thockin
10. kubernetes/architecture.md at release-1.2 - kubernetes/kubernetes
11. Kubernetes - User Guide
12. An Introduction to Kubernetes

END

Eueung Mulyana

http://eueung.github.io/docker-stuff/kubernetes

CodeLabs | Attribution-ShareAlike CC BY-SA