Kubernetes:
Managing Containers the Google way

MiMUW, 09 Oct 2014

Jarek Kuśmierek
Engineering Manager

jdk@google.com

Google Cloud Platform

http://goo.gl/N7n6gF

Why Containers*?

1. Packaging
2. Efficiency and Speed
3. Security (?)

(*) Container = Docker flavor container

Packaging

Static application environment

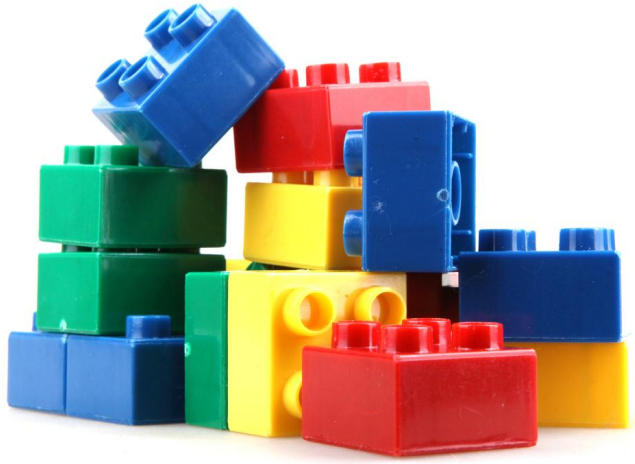
No stress deployment and update

Repeatable portable artifact

Develop here, run there
Pick your cloud solely on its merits

Loosely coupled
= easier to build and manage

Easier to build and manage
Compose applications from micro-services

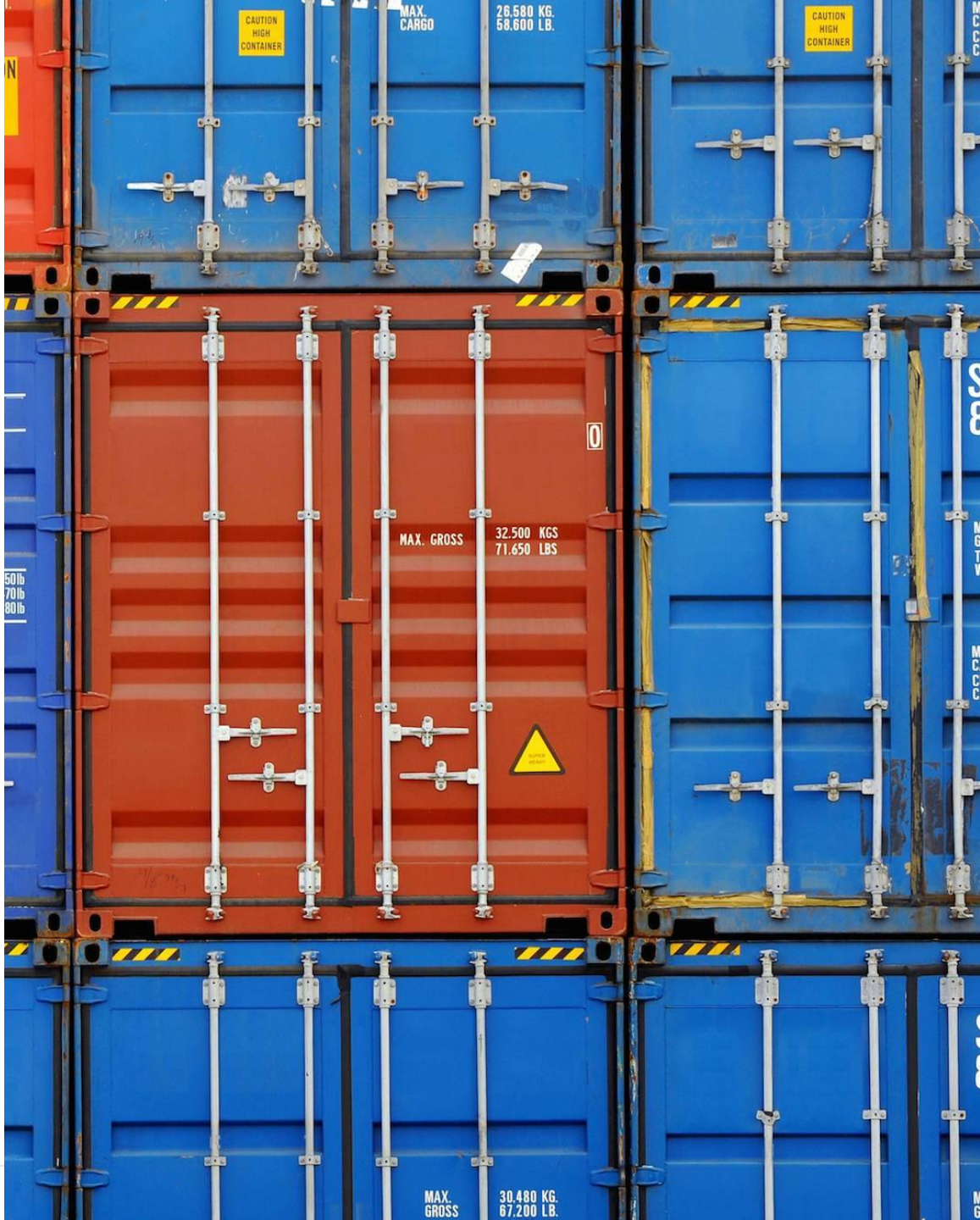


Efficiency

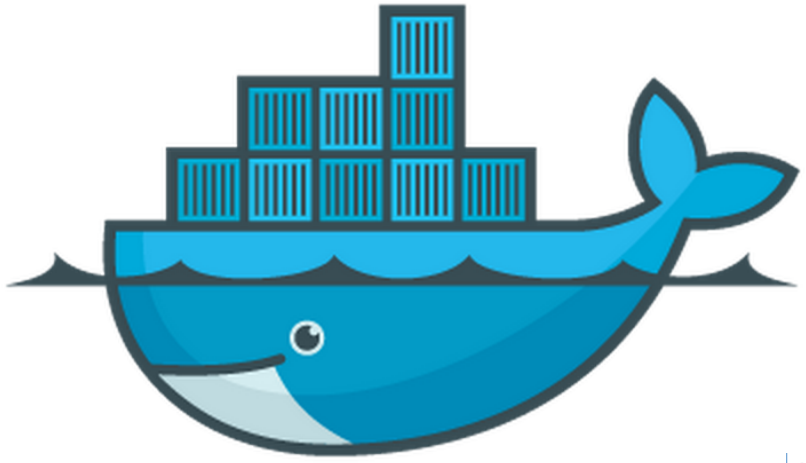
Everything at Google runs in a container.

- Resource isolation
- Predictability
- Quality of service
- Efficient overcommit
- Resource accounting

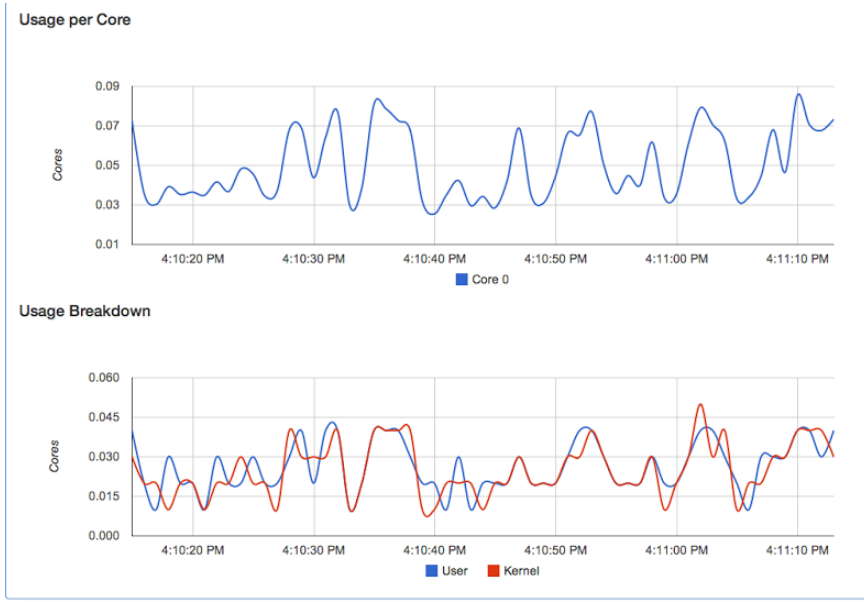
Google starts over 2 billion containers per week.

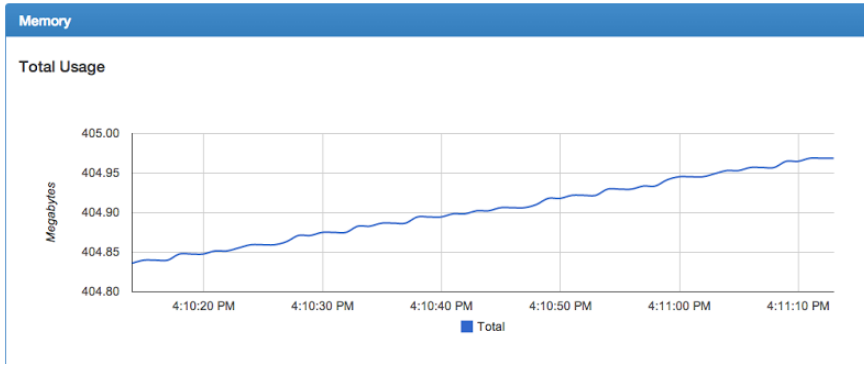


Docker



- 15000+ GitHub stars
- 600+ contributors
- 100's of projects built on top of Docker
  - UIs, mini-PaaS, ...
  - github.com/google/cadvisor
  - github.com/GoogleCloudPlatform/heapster
- Very popular GitHub project
  - Most popular Go project of all time





Kubernetes

κυβερνήτης: *Greek for “pilot” or “helmsman of a ship”*

the open source cluster manager from Google

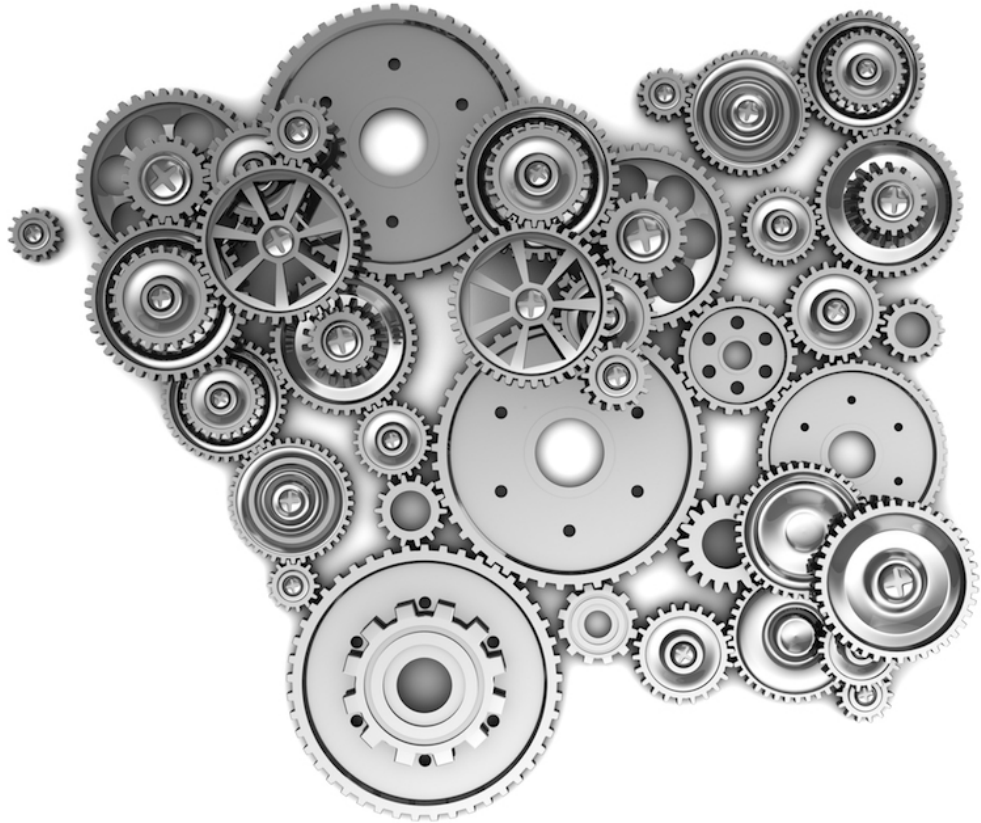
Kubernetes

Inspired by Google's systems and experience:
Manage Containers, not Machines

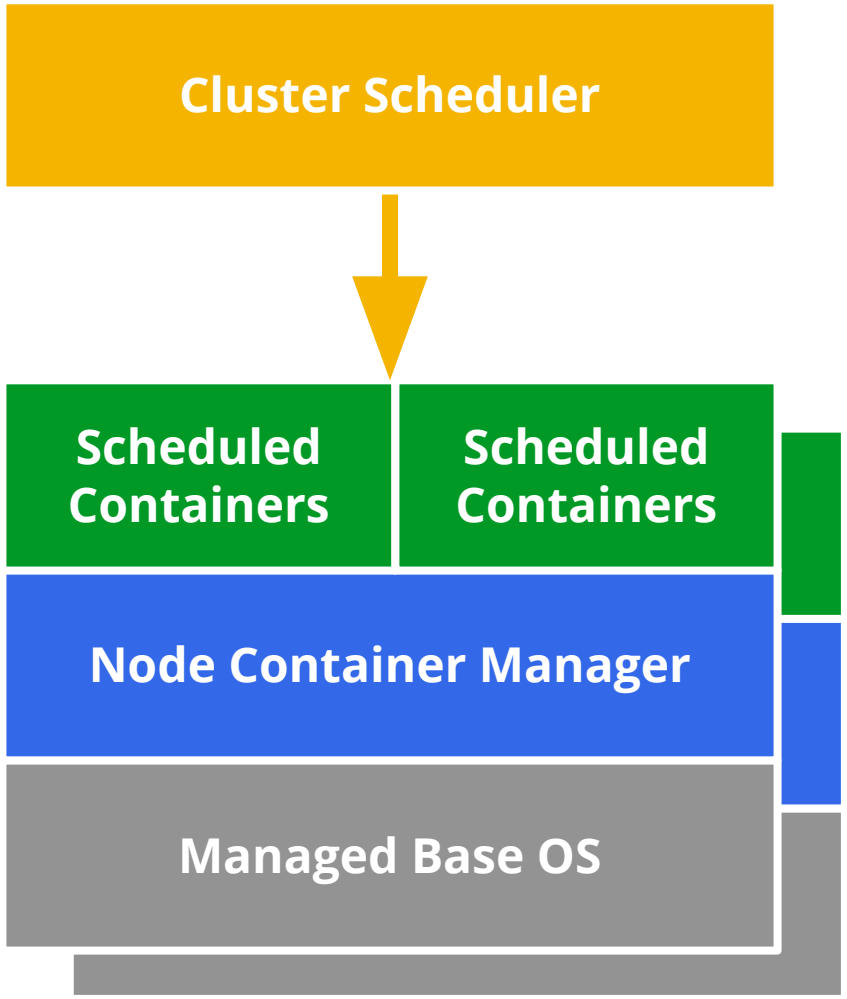
Efficient & Robust:
- optimized packing
- active monitoring
- self healing

Organizationally Scalable:
- Enable micro-services
- active scaling

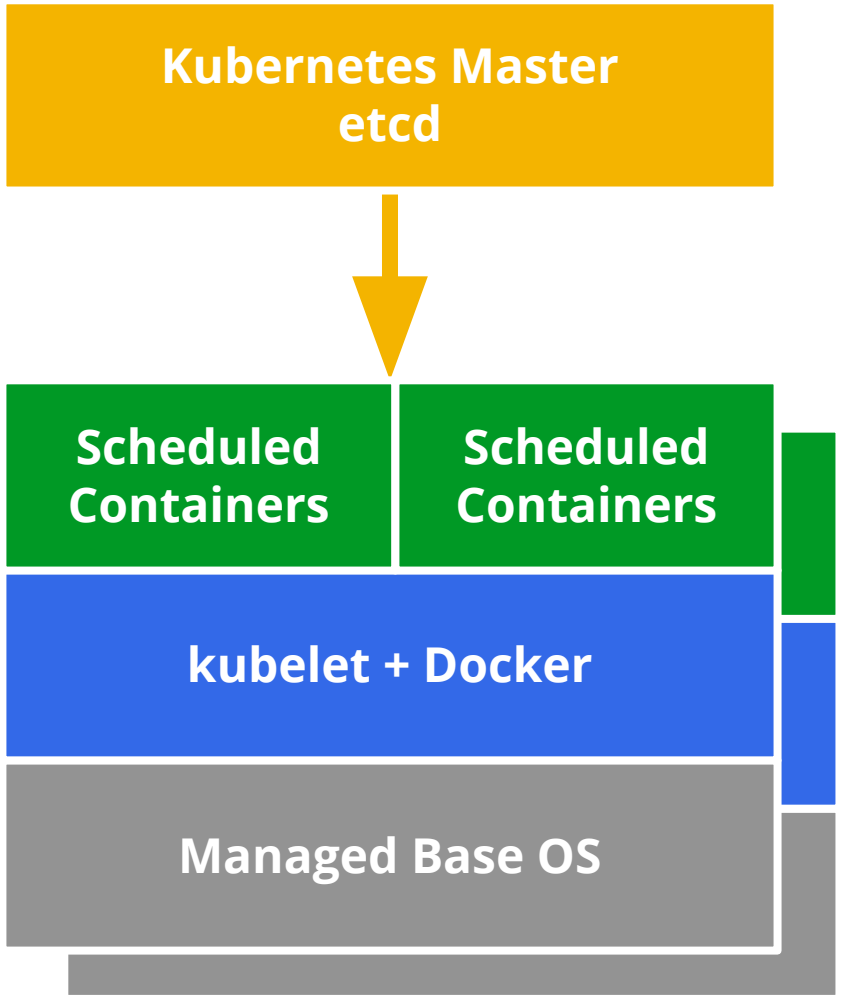
Modern Open Source:
- Extensible & portable, can run anywhere
- Written in Go
- Hosted on github
- Apache 2.0 licensed



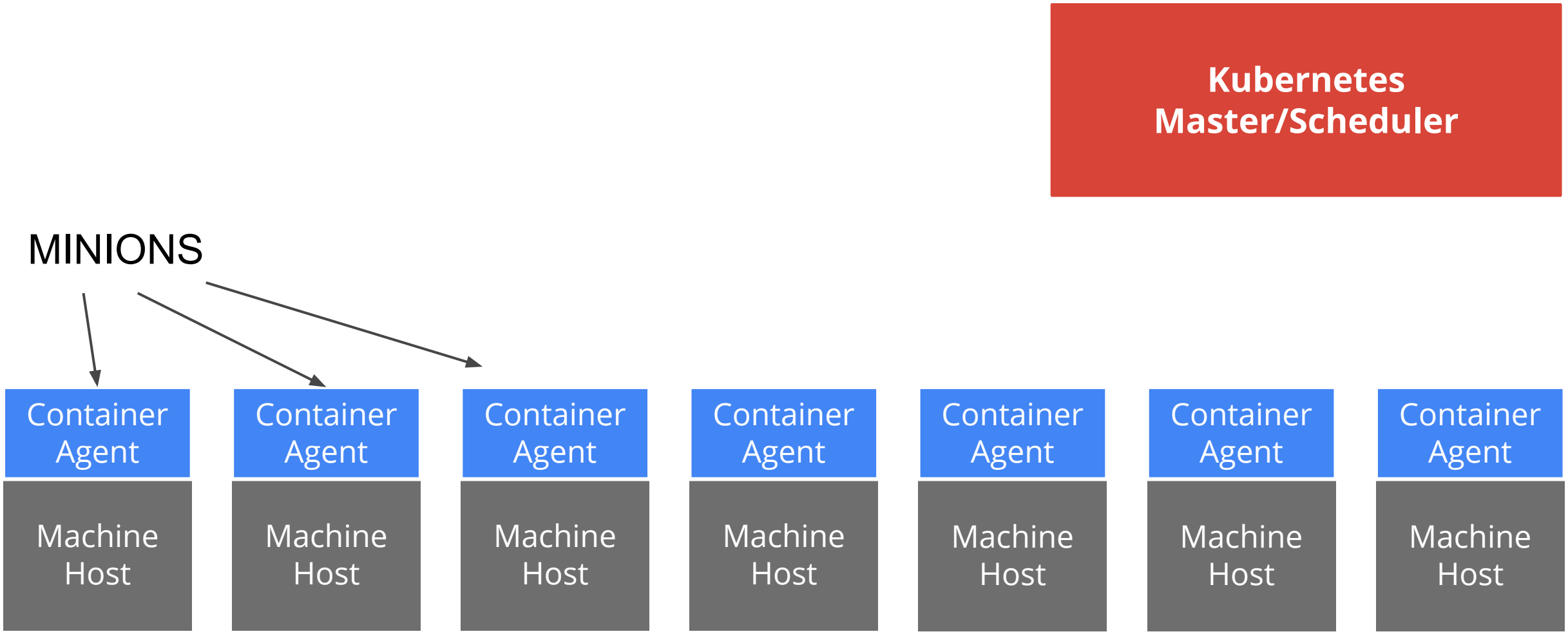
Simplified Cluster Management Stack



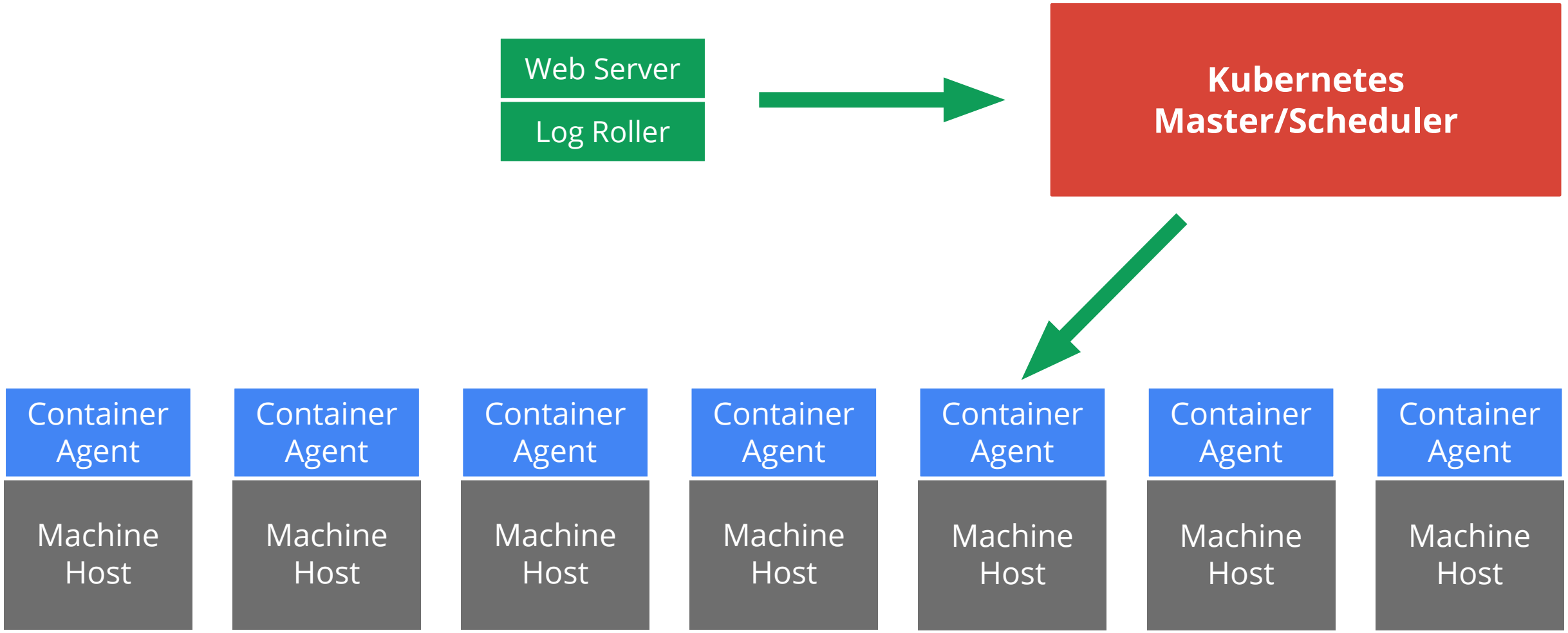
Kubernetes Container Stack



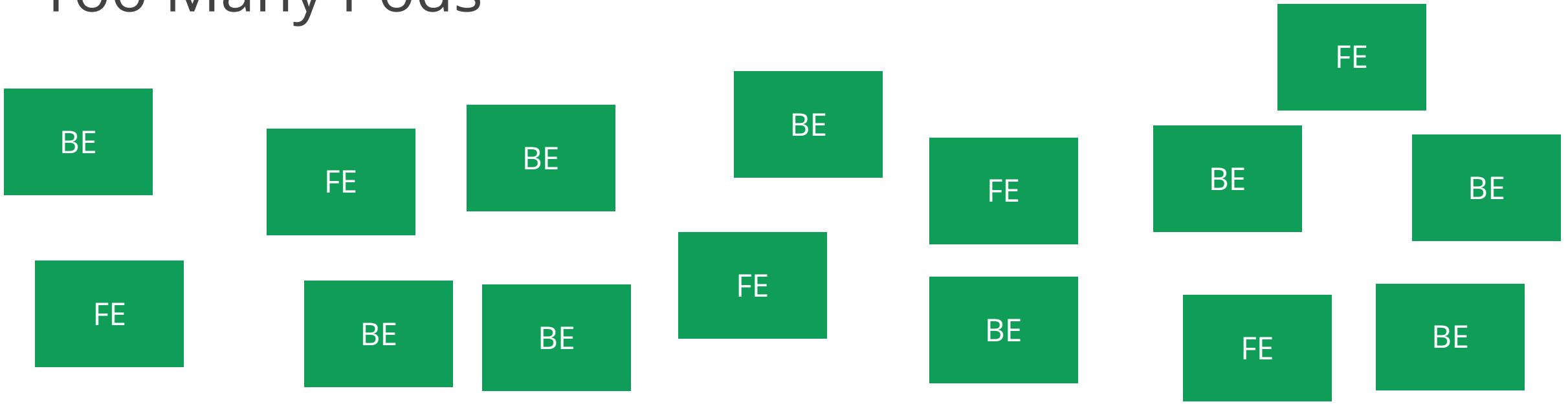
Kubernetes


Kubernetes Master/Scheduler
MINIONS
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host

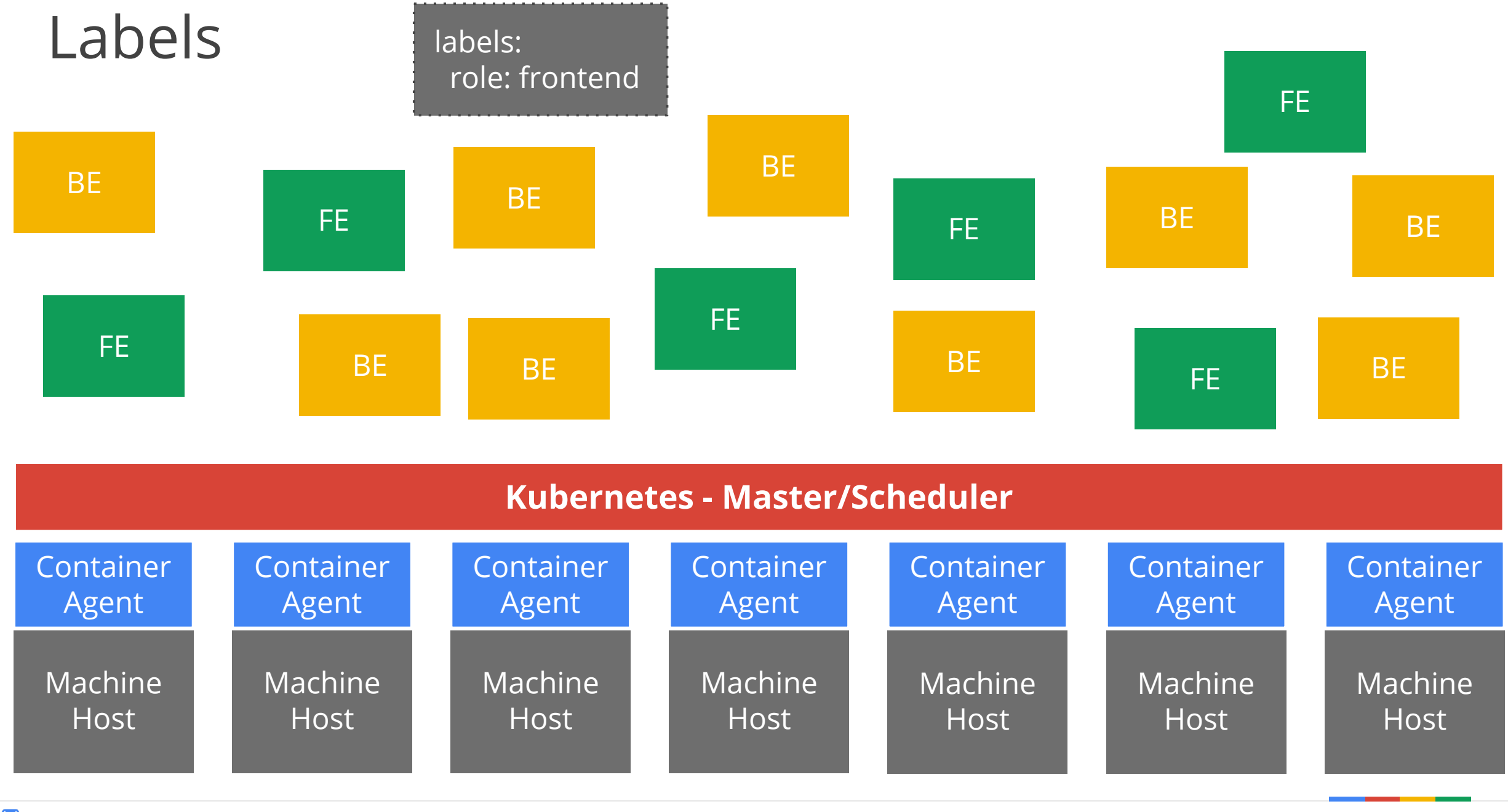
Pods


Web Server
Log Roller
Kubernetes Master/Scheduler
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host

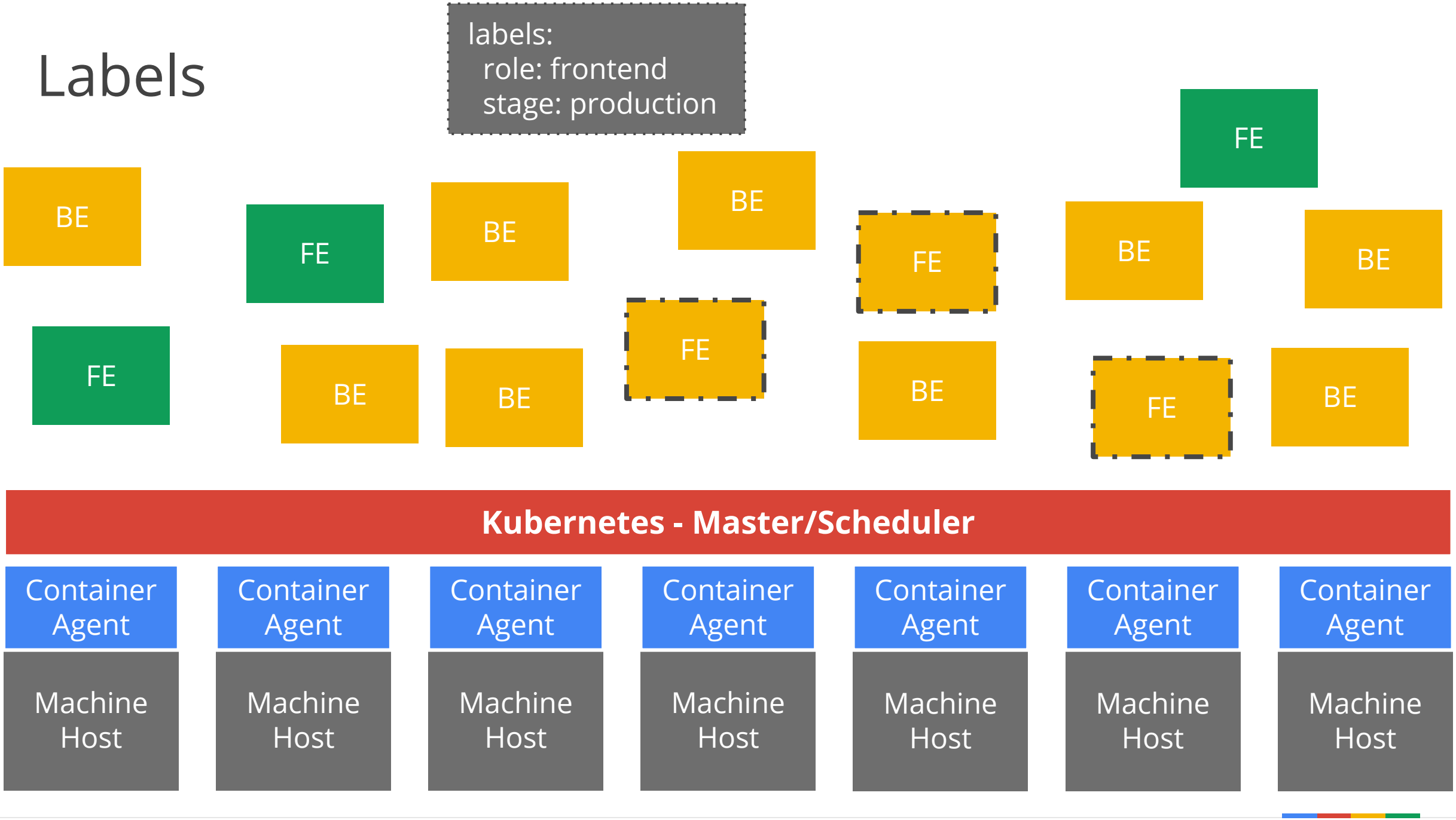
Too Many Pods



Labels


labels:
role: frontend
BE
FE
BE
BE
FE
FE
BE
FE
BE
BE
FE
BE
BE
FE
BE
Kubernetes - Master/Scheduler
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host

Labels


labels:
role: frontend
stage: production
BE
FE
BE
BE
FE
FE
BE
BE
FE
BE
BE
FE
BE
Kubernetes - Master/Scheduler
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host
Container Agent
Machine Host

Replication Controller

FE FE FE FE

```
replicas: 4
template:
  ...
labels:
  role: frontend
  stage: production
```

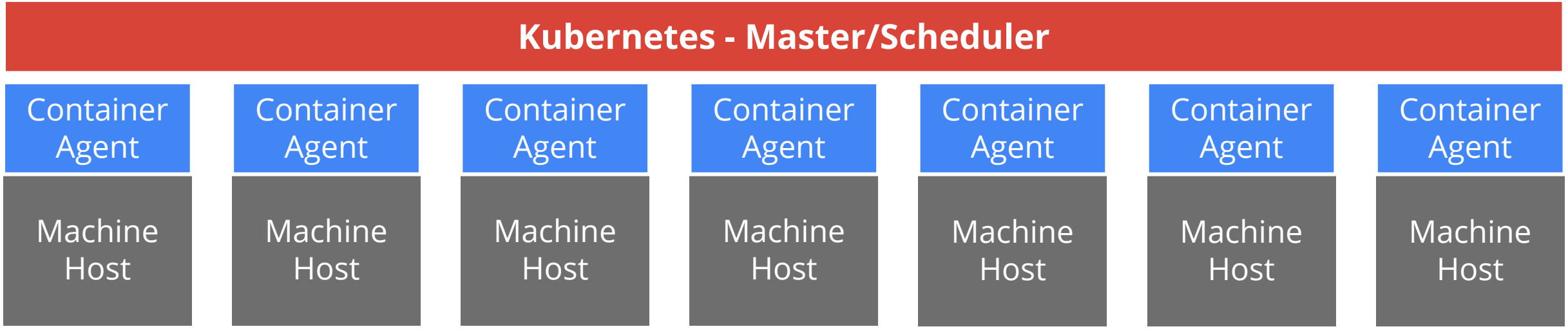
Kubernetes - Master/Scheduler

Container Agent	Container Agent	Container Agent	Container Agent	Container Agent	Container Agent	Container Agent
Machine Host	Machine Host	Machine Host	Machine Host	Machine Host	Machine Host	Machine Host

Replication Controller

FE

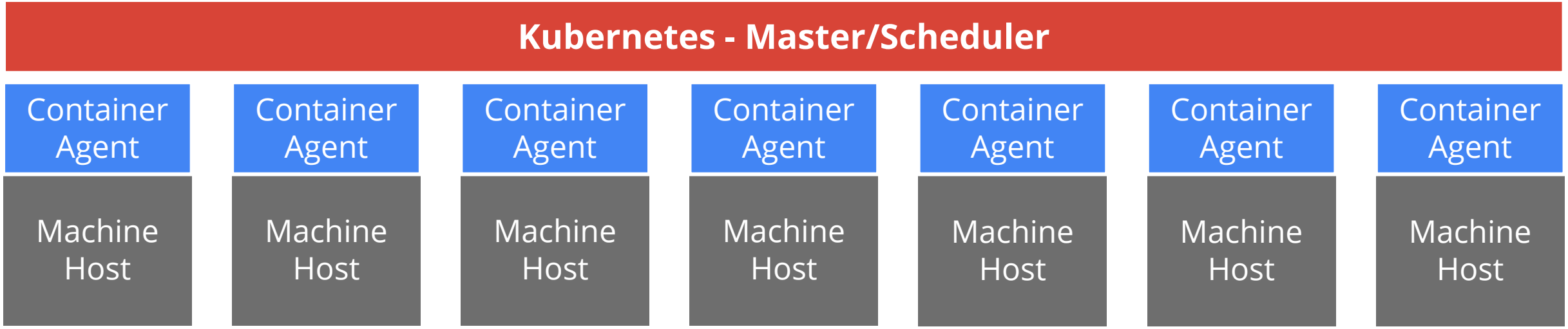
```
replicas: 1
template:
  ...
labels:
  role: frontend
  stage: production
```



Replication Controller

FE FE FE

```
replicas: 3
template:
  ...
labels:
  role: frontend
  stage: production
```

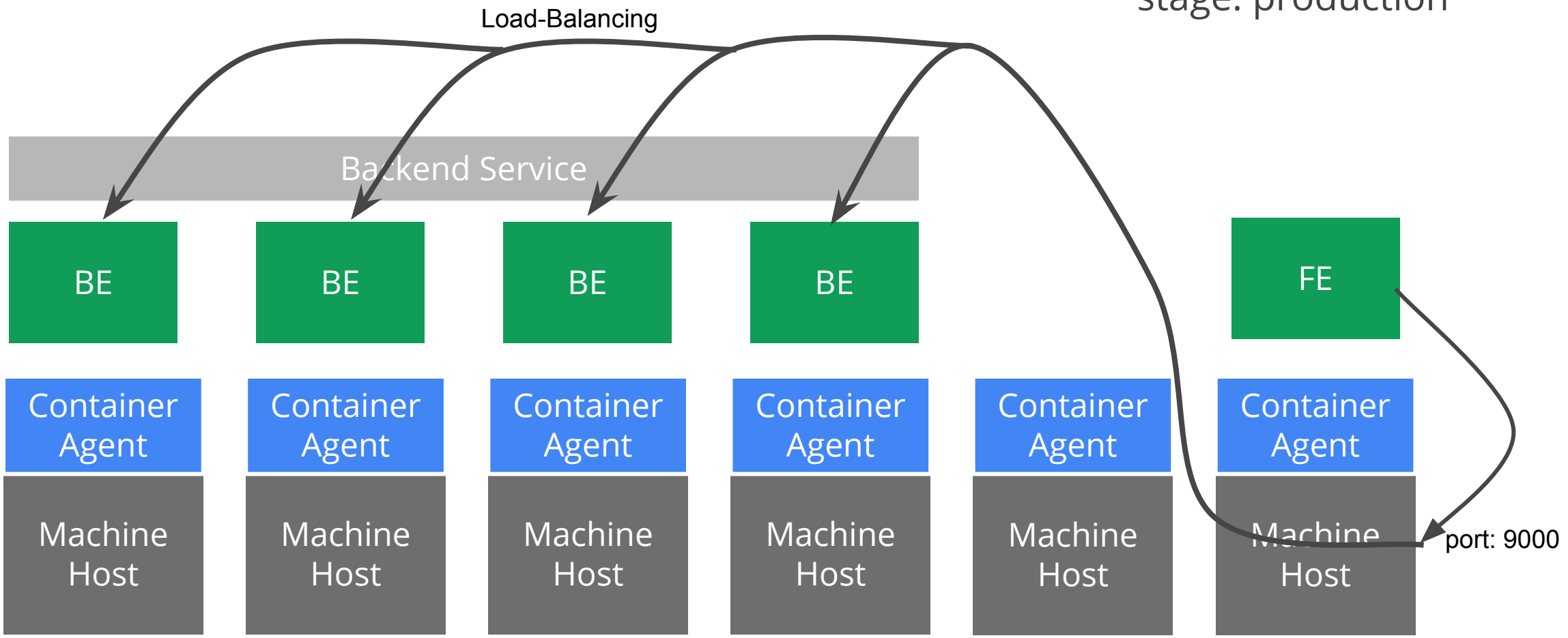


Service

Pods are auto-placed: COOL!

but: How to direct FE->BE traffic ?

id: backend-service
port: 9000
labels:
  role: backend
  stage: production



Declarative Over Imperative

Imperative:

"for pod in pod{001..100} ; start $pod

Declarative:

"Run 100 copies of this pod with a target of <= 2 tasks down at any time"

Pros:

- Repeatable
- "Set it and forget it"
- Eventually consistent
- Easily updatable

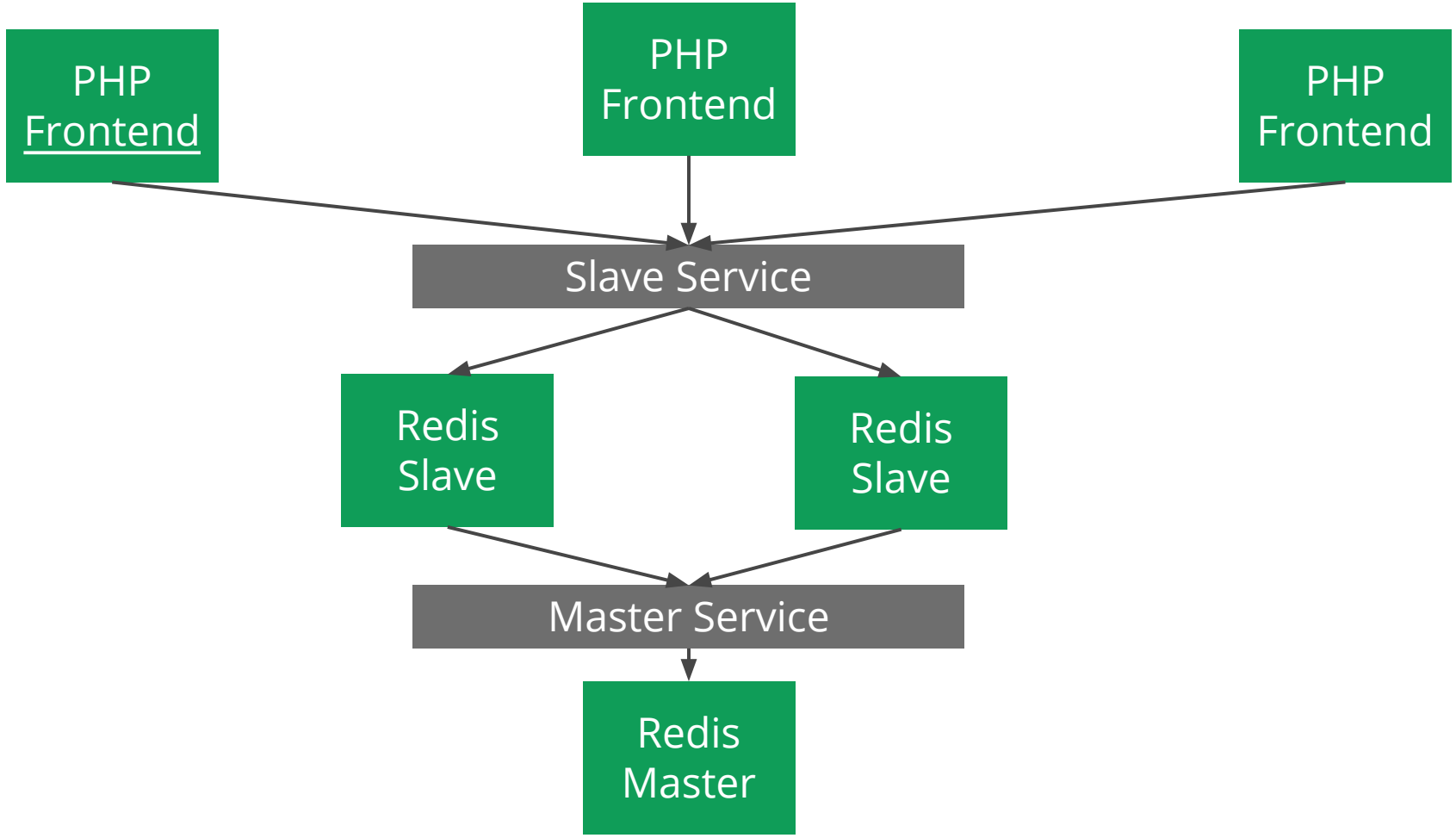
Con:

- Tracing action/reaction can be difficult. "I made a change, is it done?"

Image: "Space Needle under construction, 1961" seattlemunicipalarchives CC-BY-2.0 http://www.flickr.com/photos/seattlemunicipalarchives/6847114249

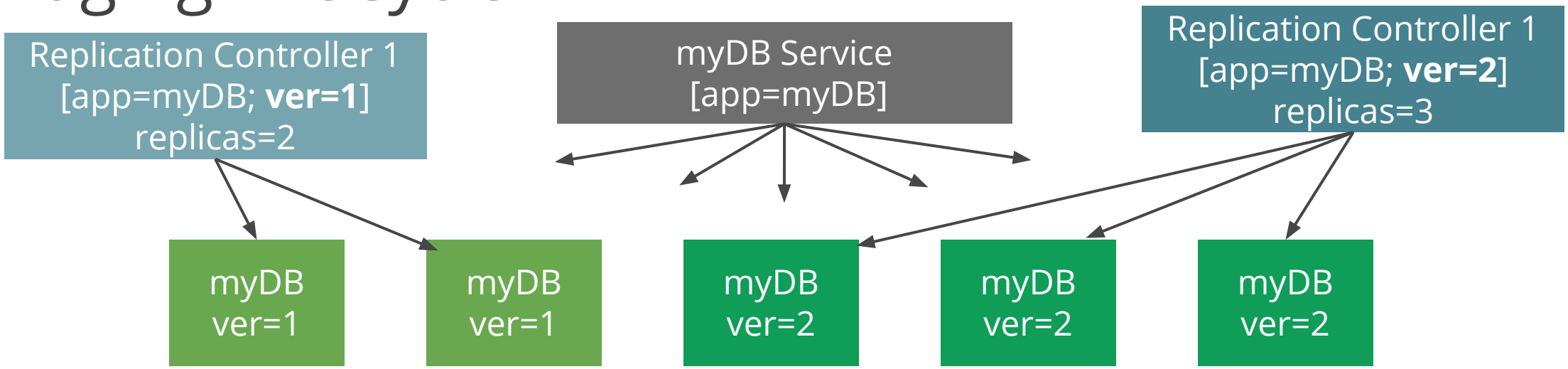
Demo (Guestbook)

Finish



Cat in a Container

Managing LifeCycle



Kubernetes - Master/Scheduler

Container Agent	Container Agent	Container Agent	Container Agent	Container Agent	Container Agent	Container Agent
Machine Host	Machine Host	Machine Host	Machine Host	Machine Host	Machine Host	Machine Host

Summing Up

1. We're taking lessons we've learned and open sourcing them
2. Kubernetes is evolving
3. We're eager to hear from you!

We are just getting started...

Clone Kubernetes at:
github.com/GoogleCloudPlatform/kubernetes

IRC: **#google-containers** on Freenode

Google group: **google-containers**

Reach out:

jdk@google.com
ricc@google.com

WANT TO TRY?

https://cloud.google.com/developers/starterpack/

Backup slides

Step 1: Create a redis master

```
jdk$ ls
README.md              index.php              redis-slave
frontend-controller-jdk.json php-redis             redis-slave-controller.json
frontend-controller.json redis-master-service.json redis-slave-service.json

jdk$ ../../cluster/kubecfg.sh -c redis-master.json create pods
ID                  Image(s)            Host                Labels              Status
----------          ----------          ----------          ----------          ----------
redis-master-2      dockerfile/redis    /                   name=redis-master   Waiting

jdk-macbookpro2:guestbook jdk$ ../../cluster/kubecfg.sh list pods
ID                  Image(s)            Host                                                Labels              Status
----------          ----------          ----------                                          ----------          ----------
redis-master-2      dockerfile/redis    kubernetes-minion-2.c.pivotal-gearing-505.internal/   name=redis-master   Waiting

jdk-macbookpro2:guestbook jdk$ ../../cluster/kubecfg.sh list pods
ID                  Image(s)            Host                                                Labels              Status
----------          ----------          ----------                                          ----------          ----------
redis-master-2      dockerfile/redis    kubernetes-minion-2.c.pivotal-gearing-505.internal/   name=redis-master   Running
```

Step 1: master.json

```
{
  "id": "redis-master-2",
  "kind": "Pod",
  "apiVersion": "v1beta1",
  "desiredState": {
    "manifest": {
      "version": "v1beta1",
      "id": "redis-master-2",
      "containers": [{
        "name": "master",
        "image": "dockerfile/redis",
        "ports": [{
          "containerPort": 6379,
          "hostPort": 6379
        }]
      }]
    }
  },
  "labels": {
    "name": "redis-master"
  }
```

Step 2: Create a redis master service

```
jdk$ ls
README.md              index.php              redis-slave
frontend-controller-jdk.json php-redis              redis-slave-controller.json
frontend-controller.json redis-master-service.json redis-slave-service.json

jdk$ ../../cluster/kubecfg.sh -c redis-master-service.json create services
ID                  Labels              Selector            Port
----------          ----------          ----------          ----------
redismaster                             name=redis-master   10000

jdk$ … test redis somehow
```

Step 2: master.service

```
{
  "id": "redismaster",
  "kind": "Service",
  "apiVersion": "v1beta1",
  "port": 10000,
  "containerPort": 6379,
  "selector": {
    "name": "redis-master"
  }
}
```

Step 3: Create redis slaves & service

```
jdk$ ../../cluster/kubecfg.sh -c redis-slave-controller.json create replicationControllers
ID                  Image(s)                  Selector            Replicas
----------          ----------                ----------          ----------
redisSlaveController   brendanburns/redis-slave   name=redisslave     2
```

```
jdk$ ../../cluster/kubecfg.sh -c redis-slave-controller.json list pods
ID                                     Image(s)                Host                                                  Labels                                                        Status
----------                             ----------              ----------                                            ----------                                                    ----------
redis-master-2                         dockerfile/redis        kubernetes-minion-2.c.pivotal-gearing-505.internal/   name=redis-master                                             Running
4df5661d-4e0b-11e4-957e-42010af01e53   brendanburns/redis-slave   kubernetes-minion-2.c.pivotal-gearing-505.internal/   name=redisslave,replicationController=redisSlaveController   Running
4df6be88-4e0b-11e4-957e-42010af01e53   brendanburns/redis-slave   kubernetes-minion-4.c.pivotal-gearing-505.internal/   name=redisslave,replicationController=redisSlaveController   Running
```

```
jdk$ ../../cluster/kubecfg.sh -c redis-slave-service.json create services
ID                  Labels              Selector            Port
----------          ----------          ----------          ----------
redisslave          name=redisslave     name=redisslave     10001
```

```
jdk$ … test redis ...
```

Step 3: redis-slave-controller.service

```
  "id": "redisSlaveController",
  "kind": "ReplicationController",
  "apiVersion": "v1beta1",
  "desiredState": {
    "replicas": 2,
    "replicaSelector": {"name": "redisslave"},
    "podTemplate": {
      "desiredState": {
         "manifest": {
           "version": "v1beta1",
           "id": "redisSlaveController",
           "containers": [{
             "name": "slave",
             "image": "brendanburns/redis-slave",
             "ports": [{"containerPort": 6379, "hostPort": 6380}]
           }]
         }
       },
       "labels": {"name": "redisslave"}
     }},
  "labels": {"name": "redisslave"}
}
```

Step 4: Create frontend servers

```
jdk$ ../../cluster/kubecfg.sh  -c frontend-controller.json create replicationControllers

ID                  Image(s)                Selector            Replicas
----------          ----------              ----------          ----------
frontendController  brendanburns/php-redis  name=frontend       3
```

Step 5: Enjoy :-)


Guestbook
Messages
Submit
Hi Cloud Open 2014!
Welcome to dusseldorf!

Docker demo

$ docker pull dockerfile/nginx
...a7767ce9ec2c: Download complete....
$ docker run -t -i dockerfile/nginx /bin/bash

[root@055dbcc687d8]$ cd /usr/share/nginx/html
[root@055dbcc687d8]$ cp index.html jdk.html

$ docker ps

```
CONTAINER ID        IMAGE                     COMMAND             CREATED             STATUS              PORTS               NAMES
055dbcc687d8        dockerfile/nginx:latest   "/bin/bash"         21 minutes ago      Up 21 minutes       443/tcp, 80/tcp     kickass_wright
```

....
$ docker commit ... jdk1/nginx-modified
$ docker push jdk1/nginx-modified

$ docker pull jdk1/nginx-modified
docker run -p 80:80 jdk1/nginx-modified /usr/sbin/nginx