Cluster API:
The Future of Kubernetes Installers

Fabrizio Pandini
VMware
Member of Technical Staff
Italy

kubernetes

Agenda

- Introducing Cluster API
- How it works?
- Demo

Trivia Time!
History of kubernetes

1. What was the first Kubernetes installer?

2. Name a Kubernetes installer or distribution?

Cluster API

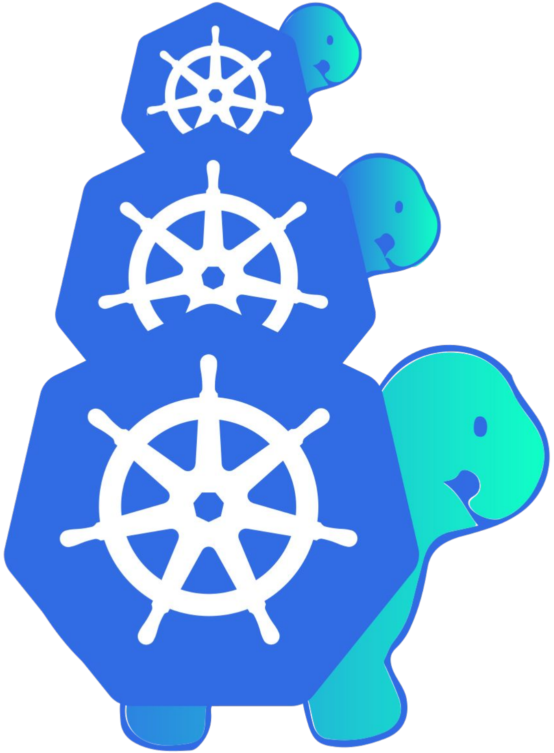
The Cluster API is a Kubernetes project to bring declarative, Kubernetes-style APIs to cluster creation, configuration, and management.

https://cluster-api.sigs.k8s.io/

1. Open Source

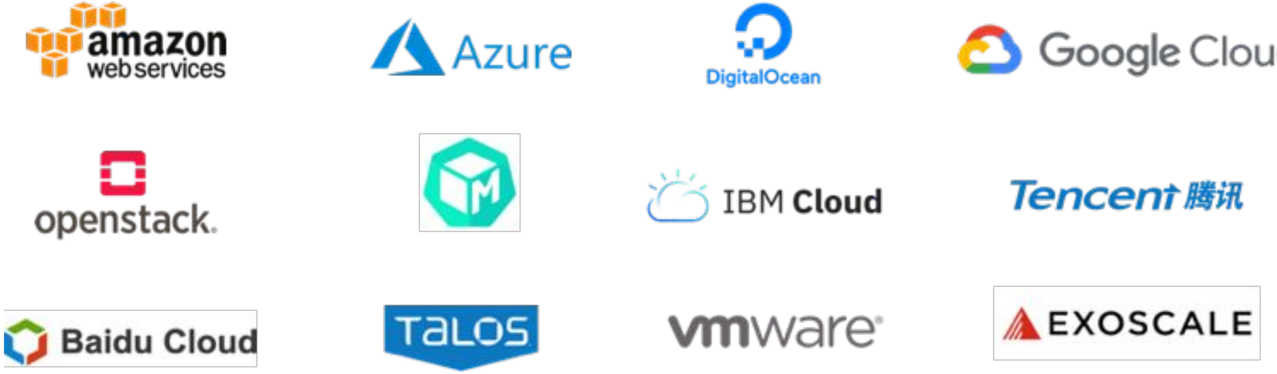
https://github.com/kubernetes-sigs/cluster-api

2. Awesome logo



3. Community traction

sigs.k8s.io/cluster-api/#provider-implementations



4. Declarative

```
kubectl apply -f cluster.yaml
```

kubernetes

5. Same experience on any provider

```
kubectl apply -f cluster-on-aws.yaml

kubectl apply -f cluster-on-vsphere.yaml

kubectl apply -f cluster-on-azure.yaml

kubectl apply -f cluster-on-google.yaml

...
```

Agenda

- Intro
- **How it works?**
- Demo

Guess what...

If you know a little about Pod and Deployments in Kubernetes, then you already know how Cluster API works.

API objects in Kubernetes

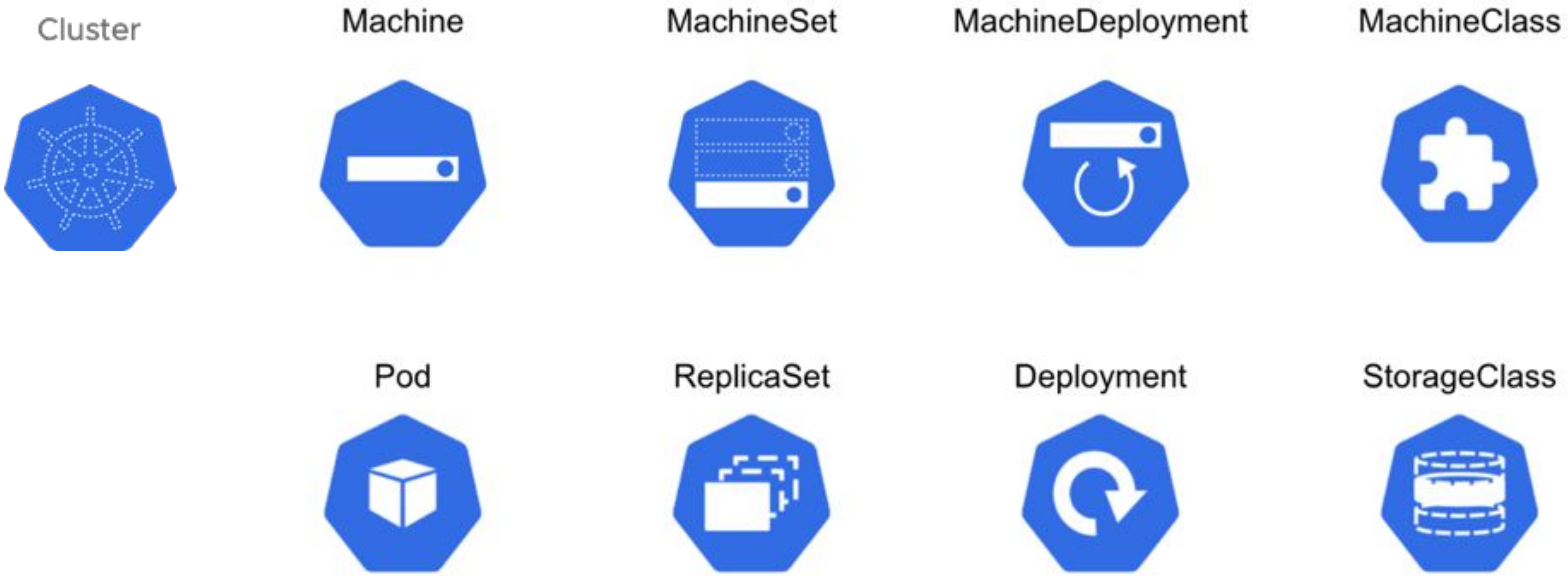
In Kubernetes API objects are **abstractions** of real objects e.g.

A Pod object is an abstraction of a workload running in a container

API objects in Cluster API

Then, why not to define
a **Machine** object
as an abstraction of any
infrastructure running in a
Kubernetes Node?

API objects in Cluster API



Cluster

Machine

MachineSet

MachineDeployment

MachineClass

Pod

ReplicaSet

Deployment

StorageClass

In practice: Cluster CRDs

my-cluster spec

```
apiVersion: cluster.x-k8s.io/v1alpha2
kind: Cluster
metadata:
  name: my-cluster
spec:
  clusterNetwork:
    pods:
      cidrBlocks: ["192.168.0.0/16"]

  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1alpha2
    kind: AWSCluster
    name: my-cluster-aws
```

my-cluster spec for AWS

```
apiVersion: infrastructure.cluster.x-k8s.io/v1alpha2
kind: AWSCluster
metadata:
  name: my-cluster-aws
spec:
  region: us-east-1
  sshKeyName: default
```

kubernetes

In practice: Machine CRDs

my-machine spec

```
apiVersion: cluster.x-k8s.io/v1alpha2
kind: Machine
metadata:
  name: my-machine
spec:
  version: v1.15.3

  infrastructureRef:
    apiVersion: infrastructure.cluster.x-k8s.io/v1alpha2
    kind: AWSMachine
    name: my-machine-aws
```

my-machine spec for AWS

```
apiVersion: infrastructure.cluster.x-k8s.io/v1alpha2
kind: AWSMachine
metadata:
  name: my-machine-aws
spec:
  instanceType: t3.large
  sshKeyName: ...
  ...
```

kubernetes

In practice: Machine CRDs

my-machine spec

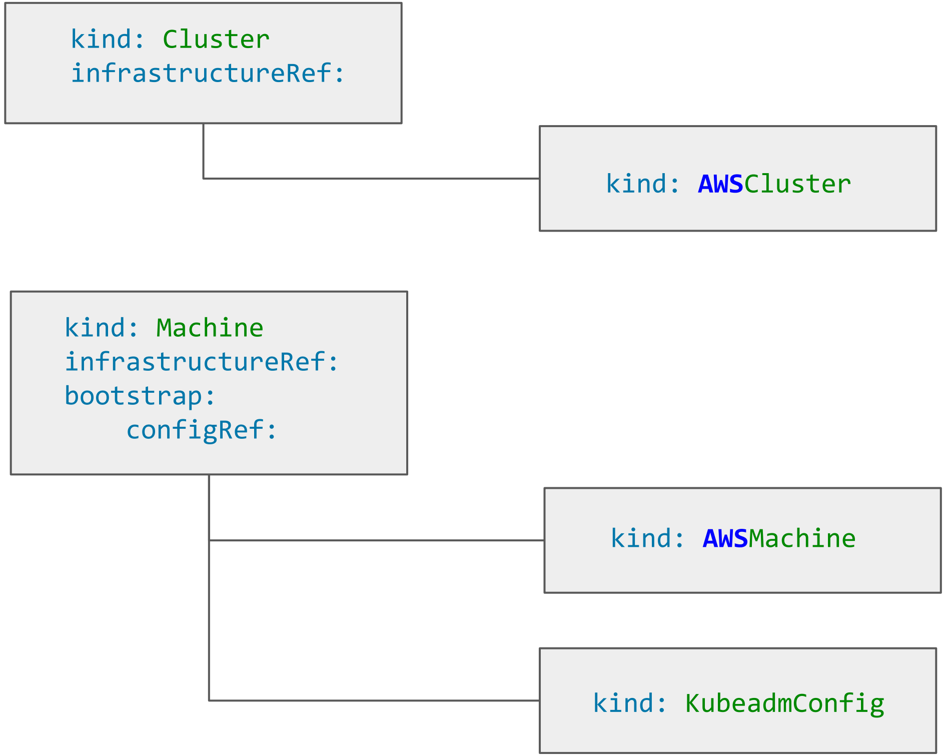
```
apiVersion: cluster.x-k8s.io/v1alpha2
kind: Machine
metadata:
  name: my-machine
spec:
  version: v1.15.3

  infrastructureRef:
    ...

  bootstrap:
    configRef:
      apiVersion: bootstrap.cluster.x-k8s.io/v1alpha2
      kind: KubeadmConfig
      name: my-machine-bootstrapper
```

node bootstrapper for my-machine

```
apiVersion:
bootstrap.cluster.x-k8s.io/v1alpha2
kind: KubeadmConfig
metadata:
  name: my-machine-bootstrapper
spec:
    initConfiguration: …
    clusterConfiguration: …
```

kubernetes

Recap



```
kind: Cluster
infrastructureRef:
```

```
kind: AWSCluster
```

```
kind: Machine
infrastructureRef:
bootstrap:
    configRef:
```

```
kind: AWSMachine
```

```
kind: KubeadmConfig
```

Trivia Time!
Kubernetes Architecture

1. Name three Kubernetes controllers?

Controllers in Kubernetes

In Kubernetes **operational best practices** for managing API objects are encoded into the controllers e.g.

The deployment controller encodes best practices for Pod rollout strategies

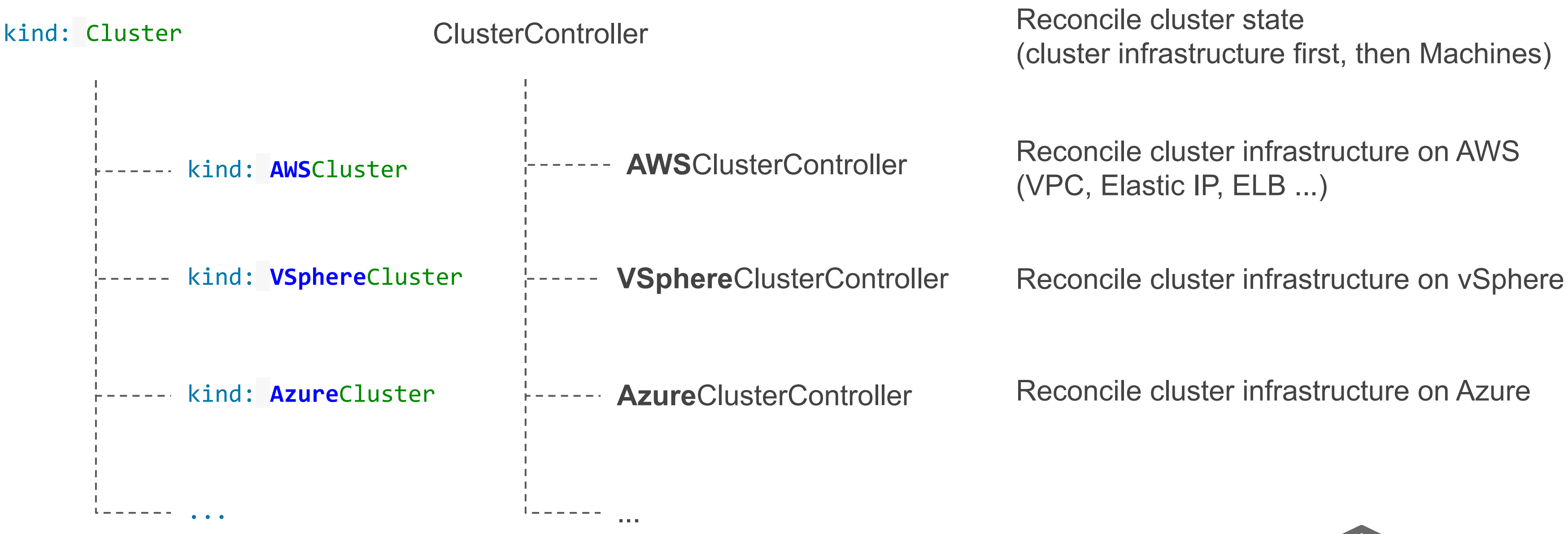
Controllers in Cluster API

In cluster API **operational best practices** for *managing** a cluster are encoded into a set of controllers.

*both provisioning and day 2 operations

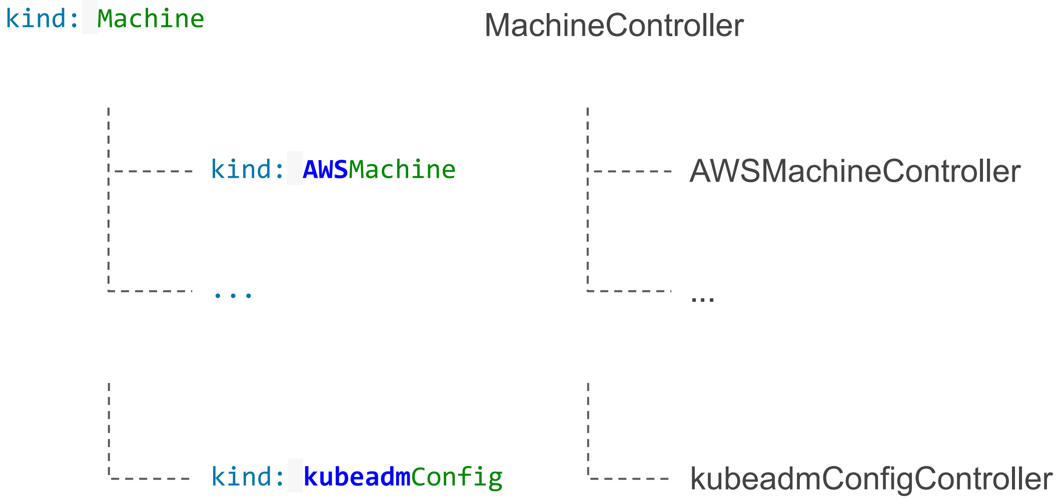
kubernetes

In practice: Cluster controllers



`kind: Cluster`	ClusterController	Reconcile cluster state (cluster infrastructure first, then Machines)
`kind: AWSCluster`	**AWS**ClusterController	Reconcile cluster infrastructure on AWS (VPC, Elastic IP, ELB ...)
`kind: VSphereCluster`	**VSphere**ClusterController	Reconcile cluster infrastructure on vSphere
`kind: AzureCluster`	**Azure**ClusterController	Reconcile cluster infrastructure on Azure
`...`	...	

In practice: Machine Controllers



kind: Machine

MachineController

Reconcile machine state
(bootstrap first, then infrastructure)

kind: AWSMachine

AWSMachineController

Reconcile machine infrastructures on AWS (EC2)

...

...

kind: kubeadmConfig

kubeadmConfigController

Generates the cloud-init scripts for bootstrapping the node with kubeadm

Recap

One controller for each “Cluster” kind

One controller for each “Machine” kind

One controller for each “Config” kind

Too many moving parts?

Up to a certain extent, yes, but

- as a user, you usually not care about controllers
- as an administrator
  - you can install only the providers you care about
  - controllers are packed into binaries/containers
- from an architectural PoV
  - each controller does one task (linux philosophy)
  - the overall system can scale
  - the overall system is more resilient

kubernetes

One last thing to know about how Cluster API works

Mutable infrastructure

In a traditional infrastructure, servers are continually updated and modified in place.

Administrators SSH into their servers and apply changes on a server-by-server basis

Immutable infrastructure

Immutable infrastructure is an approach to managing services and software deployments on IT resources wherein components are replaced rather than changed

What about Cluster API approach?

Also in this case, Cluster API mirrors what Kubernetes does for Pods/Containers.

Machines (and the related K8s) are assumed immutable and replaced in case of necessity.

TL;DR;

Clusters, Machines, are only the beginning of the story ...

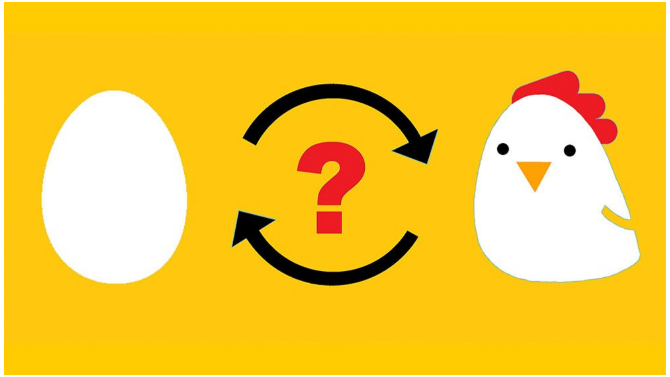
A new wave of tools and higher level abstractions is going to be created on top of that. *Stay tuned!*

Agenda

- Intro
- How it works
- Demo

https://cluster-api.sigs.k8s.io/user/quick-start.html

The chicken and egg problem



wait, wait,
I need a cluster for creating a new cluster...

1. Create a cluster

```
# create a kind config file with a mount required by the docker infrastructure provider
cat > kind-cluster-with-extramounts.yaml <<EOF
kind: Cluster
apiVersion: kind.sigs.k8s.io/v1alpha3
nodes:
- role: control-plane
  extraMounts:
    - hostPath: /var/run/docker.sock
      containerPath: /var/run/docker.sock
EOF

# create a Kubernetes cluster
kind create cluster --config ./kind-cluster-with-extramounts.yaml

# makes the cluster accessible to kubectl
export KUBECONFIG="$(kind get kubeconfig-path)"
```

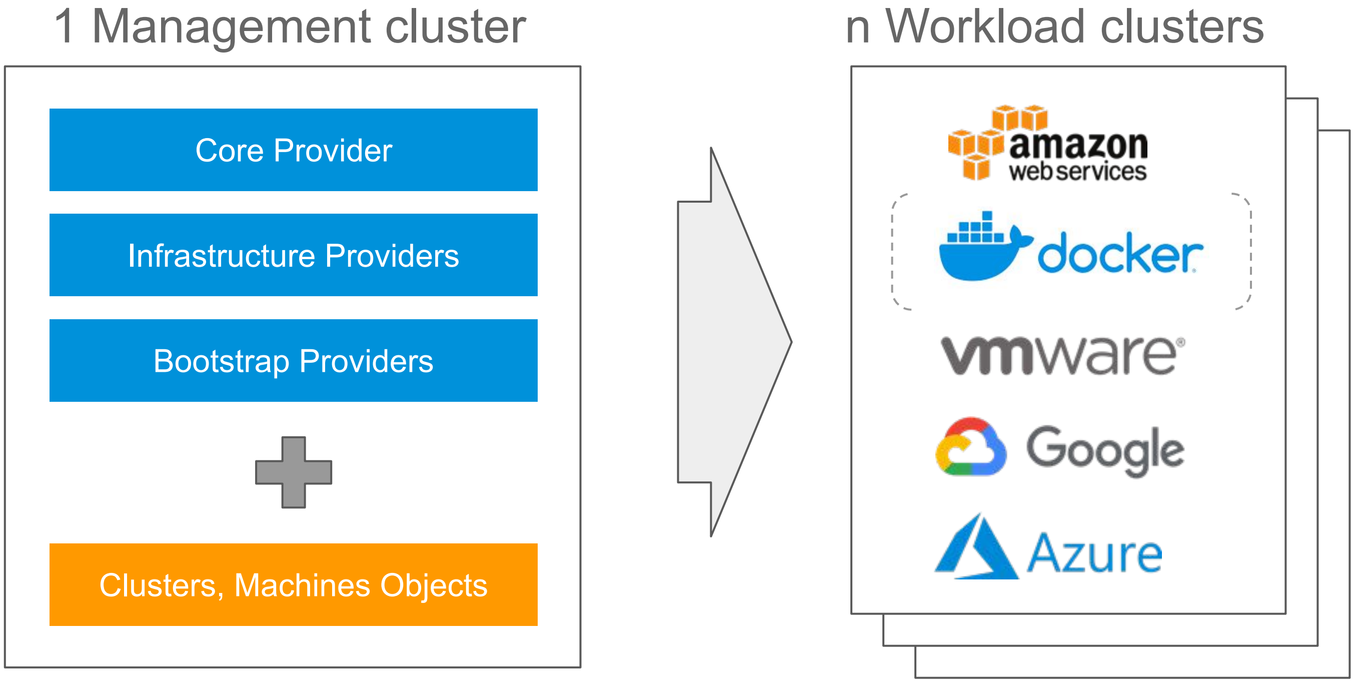
2. Transform the cluster into a management cluster.

```
# install cluster API provider
kubectl create -f
https://github.com/kubernetes-sigs/cluster-api/releases/download/v0.2.6/cluster-api-components.yaml

# install the kubeadm bootstrap provider
...

# install the docker infrastructure provider (for local testing)
...

# install the AWS infrastructure provider
...
```

or ... in the next release of cluster API

```
clusterctl init --infrastructure aws,docker --bootstrap kubeadm
```

Wrapping up, so far … and moving forward



1 Management cluster

Core Provider

Infrastructure Providers

Bootstrap Providers

Clusters, Machines Objects

n Workload clusters

Let's create our first workload cluster in docker !

```
# define the specs for the cluster
cat > cluster.yaml <<EOF
...
EOF
cat > controlplane.yaml <<EOF
...
EOF
cat > workers.yaml <<EOF
...
EOF

# create the cluster
kubectl apply -f cluster.yaml
kubectl apply -f controlpane.yaml
kubectl apply -f workers.yaml
```

or ... in the next release of cluster API

```
clusterctl config cluster my-cluster1 --infrastructure docker | kubectl apply -f -
```

Let's create our first workload cluster in AWS !

```
# Guess what… you already know how to do it !
```

Thank you!

Next up…
implementing a Kubernetes controller