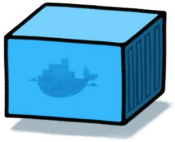
Kubernetes 101

@pionerasdev
Pioneras Developers

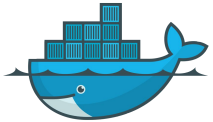
@UserTwoGG
Catherin Cruz

Background



Containers

Allows to specify exactly which parts of your environment you need and want to run. Are easy to package and move, and often utilize compute resources more efficiently. Decouple an application and its dependencies from OS.



Docker

Performs OS-level virtualization also known as containerization, is primarily developed for Linux, where it uses the resource isolation features of the Linux kernel to allow independent "containers" to run within a single Linux instance, avoiding the overhead of startings virtual machines (VMs).

Where it begins...

It was developed by **Google**, and announced in mid-2014. Many of the key developers were previously working on Google's in-house orchestration.

With Kubernetes v1.0 in the summer of 2015, Google open-sourced it and partnered with the Linux Foundation in a joint effort to advance the technology.

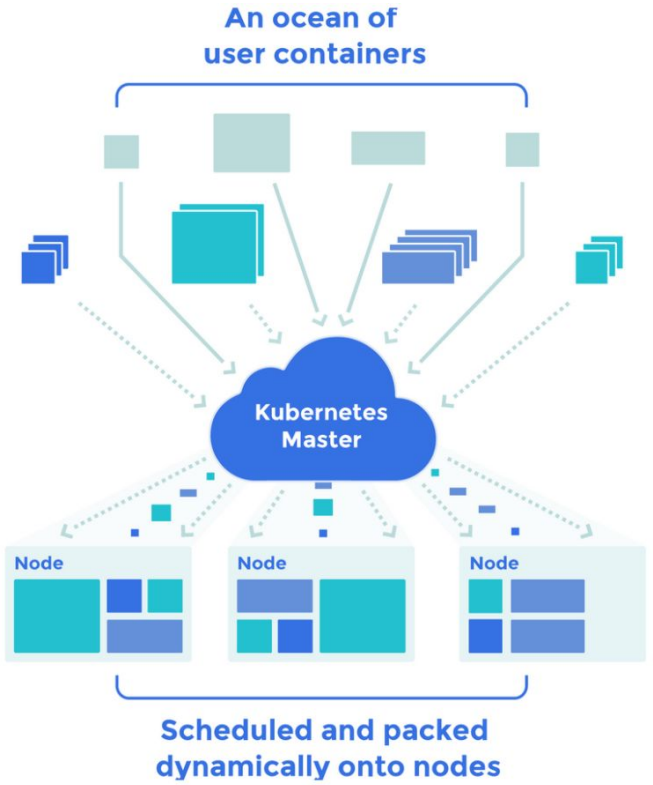
Currently, it is hosted by Cloud Native Computing Foundation(CNCF).

K8s handles all the crucial deployment, scaling and management steps so you don't have to. It's essentially a tool for managing your containers.

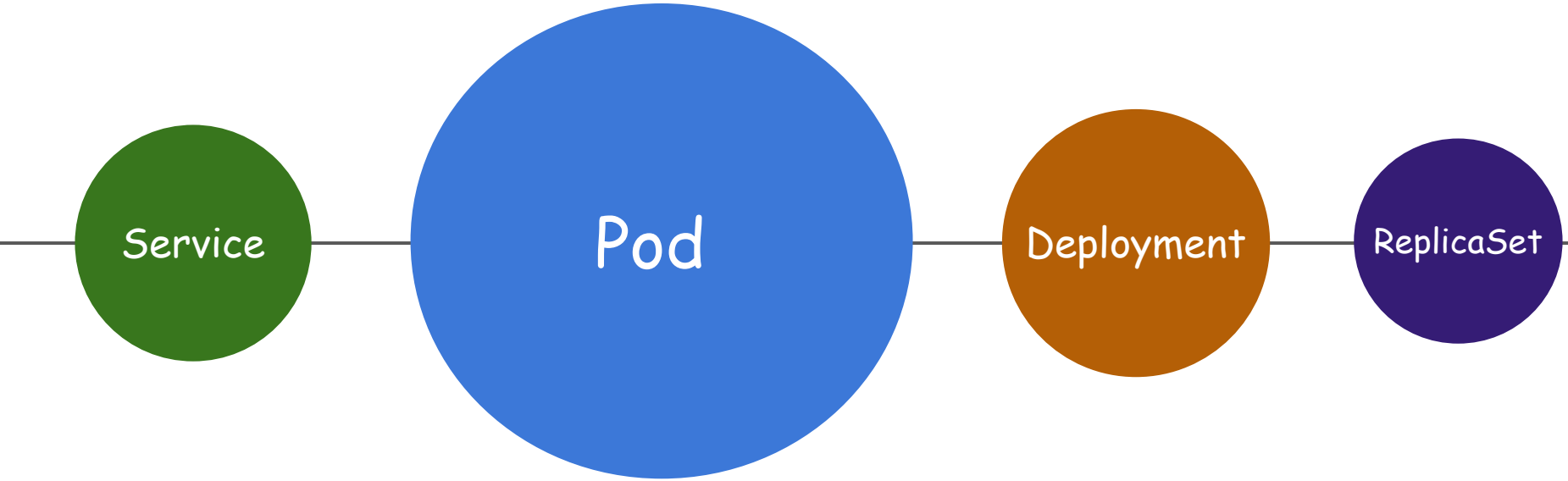
A platform to set how to serve your containers and then manage them.

Platform features

- ❑ Container grouping using pod
- ❑ Self-healing
- ❑ Auto-scalablility
- ❑ DNS management
- ❑ Load balancing
- ❑ Rolling update or rollback
- ❑ Resource monitoring and logging

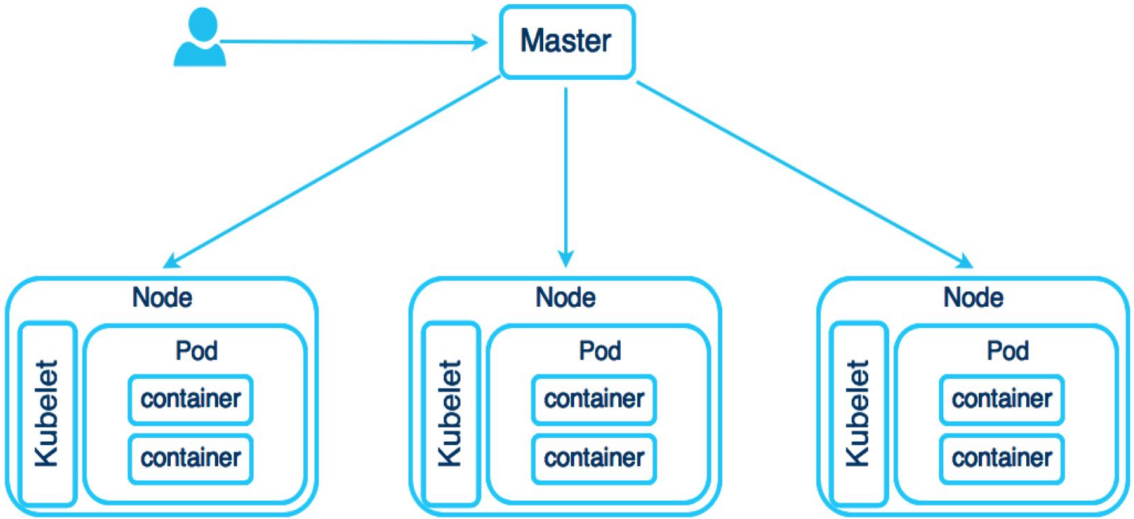


Kubernetes Objects



The Deployment instructs K8s how to create and update instances of your application. Once created, the K8s master schedules the application instances into individual Nodes in the cluster.

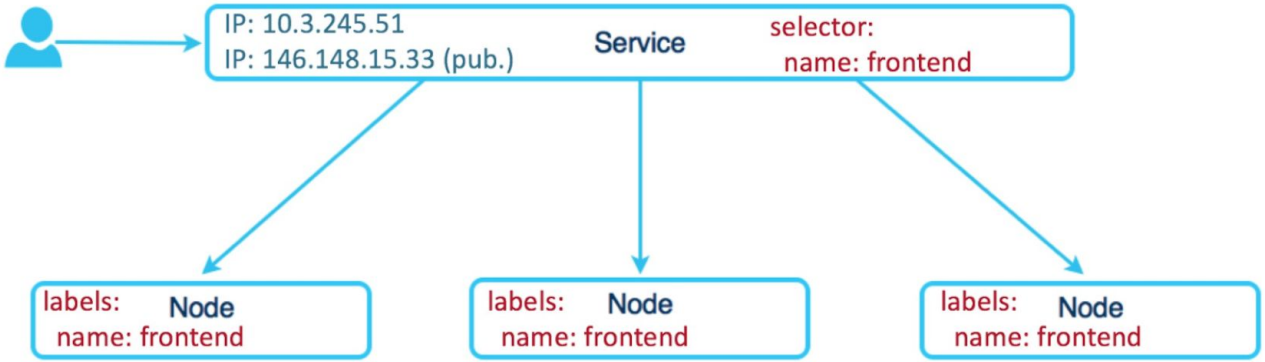
Pod



A group of one or more containers (such as Docker containers), with shared storage/network, and a specification for how to run the containers. Even if the pod has several containers, they will all be reachable in the network through a single IP address.

Service

```
ports:
  -port: 80
   targetPort: 8181
```



An abstraction which defines a logical set of Pods and a policy by which to access them. Pods have a life cycle. They get created and die. We need a way to make them accessible on a regular basis, even if they are re-created. By giving Pods a certain label we use a Service to route traffic to all Pods with that particular label. Voila! Reliable access to Pods even if they are re-created.

Deployment

Manifest File(s)

- Optional only in *trivial* cases.
  - (trivial = CLI only possible)
- YAML (or JSON) format.

Labels

- Key/Value "tags" – placed on any deployable object.
- Selectable – by actions and other declarations.
  - Configuration Flexibility

Descriptor Types (partial list)

- Replication Controller
- Deployment
- Pod
- Job
- Service

PodSpec clause – within most descriptors

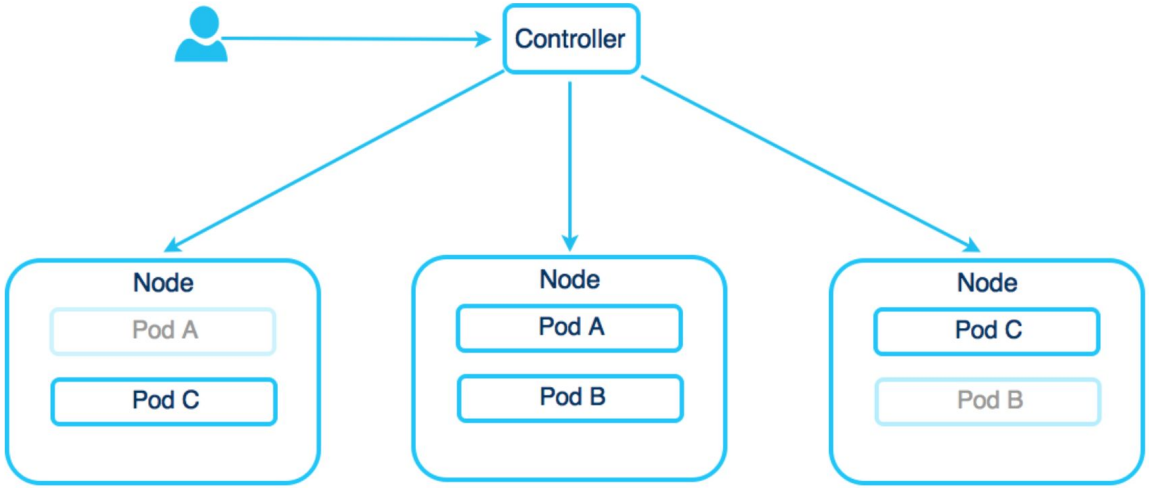
- Labeled
  - allows versioning
  - other constraint application
- Container(s)
  - very Dockerfile / docker-compose like.
  - Image location, (including image version)
  - Volume requirements
  - Ports exposed

Replication Controller descriptor

- "template/spec" clause declares PodSpec configuration.
- "replica" clause declares sizing of the service.
- Rolling-updates & canary deploys are a supported

Describes the desired state and makes sure to change the actual state to the desired state if needed. A deployment manages Pods and ReplicaSets so you don't have to. Just like magic!

ReplicaSet



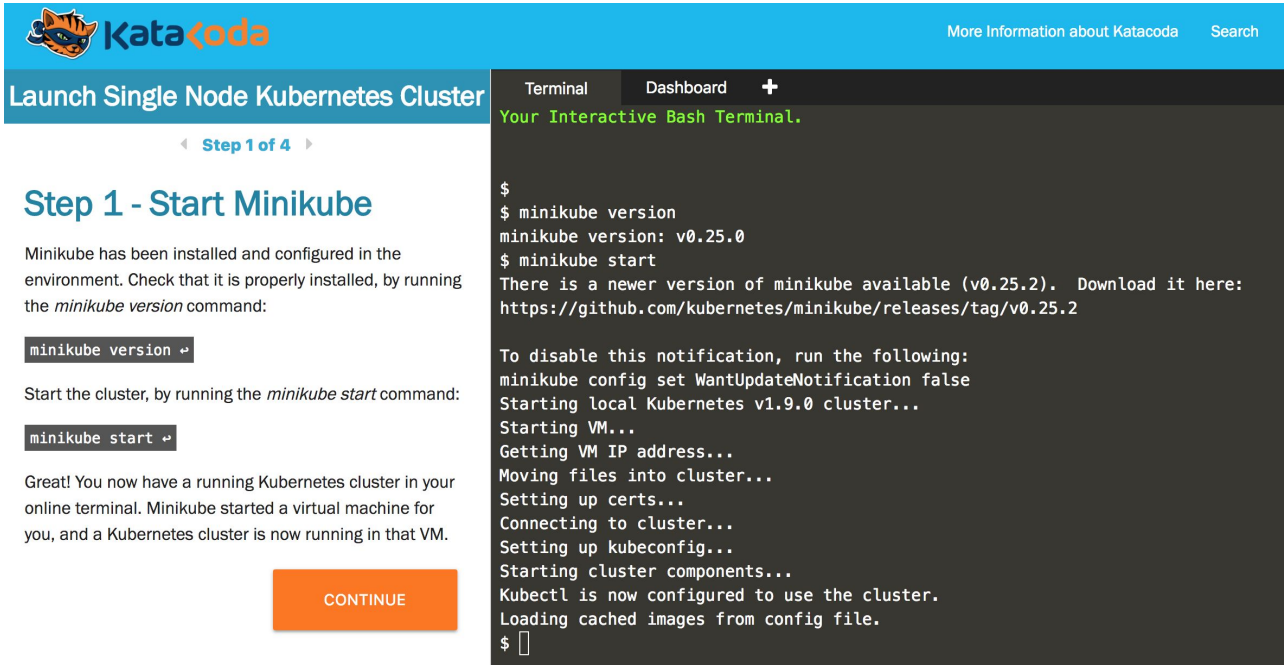
Manages the lifecycle of a Pod. While you are free to deploy a single Pod, you need the RS to ensure that a certain amount of Pods are running. If a Pod dies, the RS creates a new one, if you scale up/down Pods, the RS creates/destroys your Pods.

Now, let's practice in a Cluster!

We are going to run a Minikube single-node cluster, deploy a Go image stored in Hub Docker, then we will interact with some kubectl commands in K8s cluster. Also, we are going to use single node cluster scenario for free in Katacoda online kubernetes playground.

Source: https://github.com/twogg-git/k8s-intro

Katacoda Single-Node Cluster



https://www.katacoda.com/courses/kubernetes/launch-single-node-cluster

Basic Commands

Minikube a tool that makes it easy to run K8s locally, runs a single-node K8s cluster inside a VM.

```
$ minikube start
```

```
$ minikube dashboard
```

Kubectl is a command line interface for running commands against Kubernetes clusters.

```
$ kubectl cluster-info
```

```
$ kubectl get nodes
```

```
$ kubectl get all
```

Deployment

```
$ kubectl run twogg --image=twogghub/k8s-intro:1.4-k8s
```

```
$ kubectl get deployment twogg --output wide
```

```
$ kubectl expose deployment twogg --port=8080
          --external-ip=$(minikube ip) --type=LoadBalancer
```

- In Katacoda click + button then “Select port to view on Host1”, in new tab enter port **8080**

```
$ kubectl describe service twogg
```

```
$ kubectl get pods,services,deployments --output wide
```

Update and Scale

```
$ kubectl set image deployment twogg twogg=twogghub/k8s-intro:1.5-k8s
```

```
$ kubectl scale --replicas=3 deployment twogg
```

- Run --watch flag on a new terminal. In Katacoda click + button then “Open new terminal”.

```
$ kubectl get pods --output wide --watch
```

```
kubectl get pods --output wide
```

```
$ kubectl delete pod <pod-name>
```

Pod Interaction

```
$ kubectl run ghost --image=ghost:0.9
```

```
$ kubectl expose deployment ghost --port=2368 --replicas=3
    --external-ip=$(minikube ip) --type=LoadBalancer
```

```
$ kubectl label pods <pod-namename> owner=pioneras
```

```
$ kubectl logs <pod-namename>
```

```
$ kubectl get pod <pod-namename> --output=json
```

```
$ kubectl exec -ti <pod-namename> /bin/bash
```

Clean up...

```
$ kubectl delete pod <pod-namename>
```

Every pod is generated based on its deployment file, every time you delete a pod, it comes up again because you defined the value 'replicas: X' in the deployment file. To delete a Pod/s permanently, You will have to first delete the deployment of that pod and then delete the pod.

```
$ kubectl delete deployment twogg
```

```
$ kubectl delete services,pods,deployments --all
```

Links & References

- Intro to Kubectl commands
  https://github.com/twogg-git/k8s-intro
- Oficial Kubernetes Documentation
  https://kubernetes.io/docs/home/
- Kubernetes by Example
  http://kubernetesbyexample.com/
- Katacoda training courses
  https://www.katacoda.com/learn
- Kubectl commands documentacion
  https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands