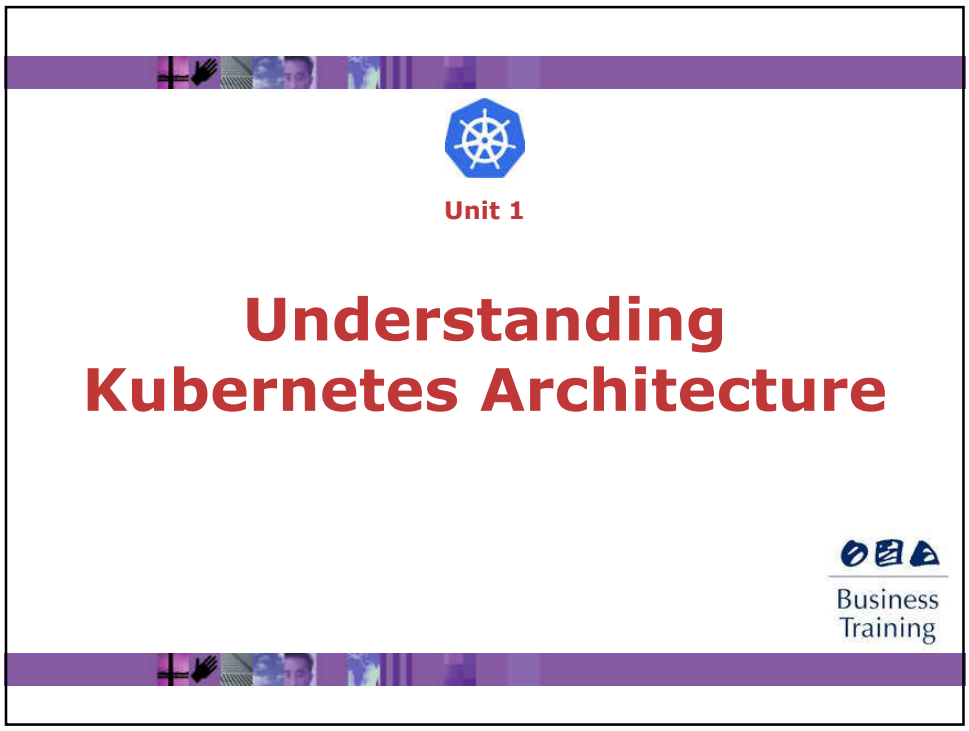


1

Outline

- Introduction to Kubernetes
- Kubernetes Architecture
- Start a Single node cluster locally with Docker Desktop for Windows
- Using the Kubernetes dashboard
- Basic Commands of Kubectl

2

Introduction to Kubernetes

3

What is Kubernetes ?

- **Kubernetes is an open source system for managing containerized applications across multiple hosts.**
  - It provides basic mechanisms for deployment, maintenance, and scaling of applications.

- **Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called Borg, combined with best-of-breed ideas and practices from the community.**
  - Created by three Google employees initially during the summer of 2014; grew exponentially and became the first project to get donated to the Cloud Native Computing Foundation (CNCF).
  - Hit the first production-grade version v1.0.1 in **July 2015**. Has continually released a new minor version every three months since v1.2.0 in March 2016. Lately v1.18.0 was released in March 2020

Greek for "pilot" or "Helmsman of a ship". Kubernetes is also the abbreviated with K8S.

4

Kubernets Key features

- **Kubernetes is an orchestration system. It runs containerized applications (such as microservices) on a cluster.**
  - It's a sort of meta-process that grants the ability to automate the deployment and scaling of several containers at once
  - These containers act as replicas, and serve to load balance incoming requests. A container orchestrator, then, supervises these groups, ensuring that they are operating correctly.
- **Kubernetes features**
  - Scale-in and scale-out workload
  - Self-healing
  - Service discovery/load balancing
  - Automate Rollouts/Rollbacks
  - Secret and Configuration management

5

Basic things we can ask Kubernetes to do

- **Suppose we have a 3-tier e-commerce app:**
  - web frontend
  - Restful API backend
  - database
  - We have built images for our frontend and backend components
- **Let's see how we would deploy our app on Kubernetes !**
  - **Start 5 containers using image myeshop/api:1.2**
  - **Place an internal load balancer in front of these containers**
  - **Start 10 containers using image myeshop/api:1.2**
  - **Place a public load balancer in front of these containers**
  - **It's Black Friday (or Christmas), traffic spikes, grow our cluster and add containers**
  - **New release! Replace my containers with the new image myeshop/webfront:v1.3**
  - **Autoscaling**
  - **Advanced rollout patterns (blue/green deployment, canary deployment)**

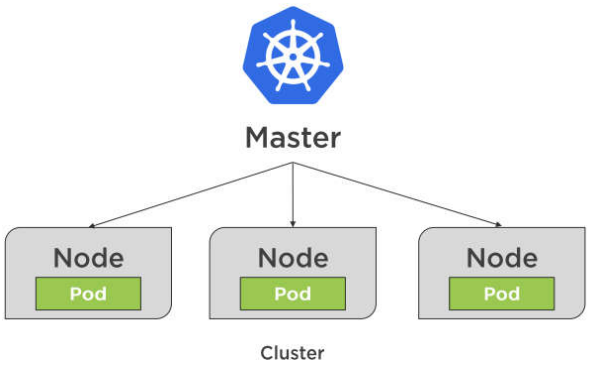
6

Kubernetes Architecture

7

Kubernetes Architecture

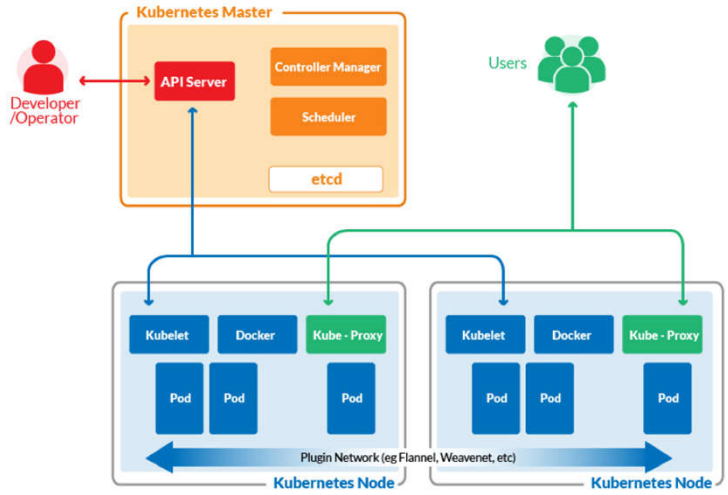
- **The Master node (Control Plane) is responsible for managing whole cluster.**
- **The worker nodes expose the underlying compute, storage and networking to the applications.**



8

Understanding Kubernetes architecture

Kubernetes Architecture



9

Understanding Kubernetes architecture

The kubernetes Master Compoents

- **The API Server**
  - The apiserver provides a facing REST interface into the kubernetes control plane and datastore. All clients, including nodes, users and other applications interact with kubernetes strictly through the API Server.
- **Etcd**
  - Etcd acts as the cluster datastore; providing a strong, consistent and highly available key-value store used for persisting cluster state.
- **The Controller Manager**
  - Controller managers are in charge of regulating the state of the system. They implement a set of the control loops that allows for Kubernetes to enforce the target desired state.
- **The Scheduler**
  - The Kubernetes scheduler is in charge of scheduling pods onto nodes. It needs to take into account individual and collective resource requirements, quality of service requirements, …

10

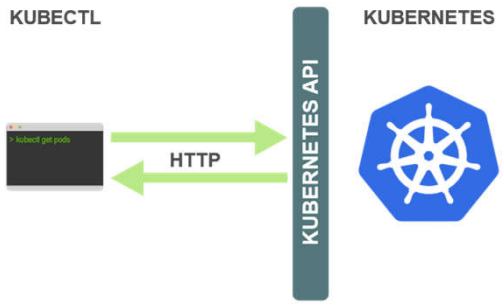
Node Components

- **Kubelet**
  - Acts as the node agent responsible for managing pod lifecycle on its host. Kubelet understands YAML container manifests that it can read from several sources:
    - File path
    - HTTP Endpoint
    - HTTP Server mode accepting container manifests over a simple API.
- **Kube-proxy**
  - Manages the network rules on each node and performs connection forwarding or load balancing for Kubernetes cluster services.
    - Creates the rules on the host to map and expose services
      - Uses a combination of iptables to manage networking/loadbalancing
- **Container Runtime**
  - With respect to Kubernetes, a container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
    - Containerd (docker)/Cri-o/rkt/Kata (formerly clear and hyper)

11

Kubectl, the CLI

- **The Kubernetes API is RESTful**
- **kubectl is (almost) the only tool we'll need to talk to Kubernetes**
  - It is a rich CLI tool around the Kubernetes API (Everything you can do with kubectl, you can do directly with the API)
  - On our machines, there is a ~/.kube/config file with: the Kubernetes API address, the path to our TLS certificates used to authenticate

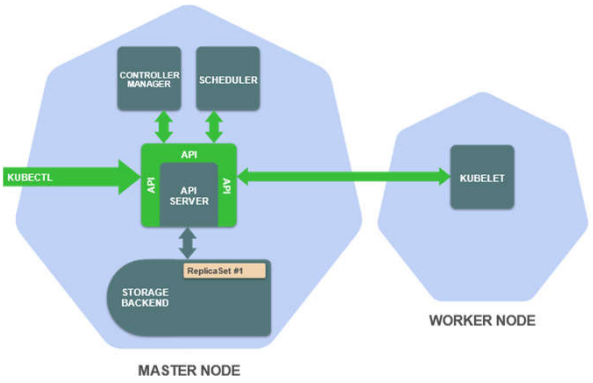


12

Understanding Kubernetes architecture

How Kubernetes works ?

- **Assume, you just executed > kubectl create -f replicaset.yaml upon which kubectl made an HTTP POST request to the create ReplicaSet API endpoint (passing along your ReplicaSet resource definition).**
  - The API server saves your ReplicaSet resource definition in the storage backend.

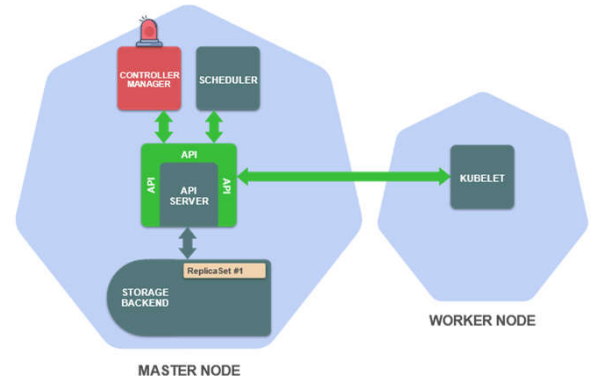


13

Understanding Kubernetes architecture

How Kubernetes works ?

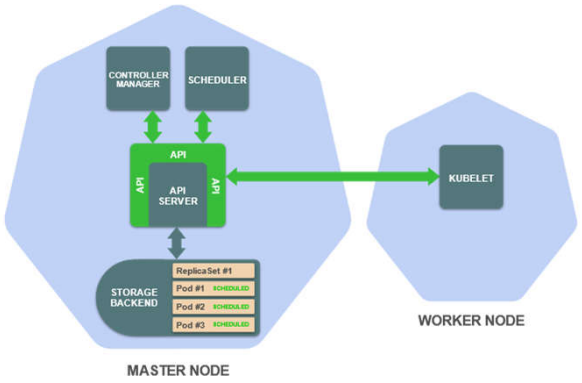
- **This triggers the ReplicaSet controller in the controller manager, who watches for creations, updates, and deletions of ReplicaSet resources.**



14

How Kubernetes works ?

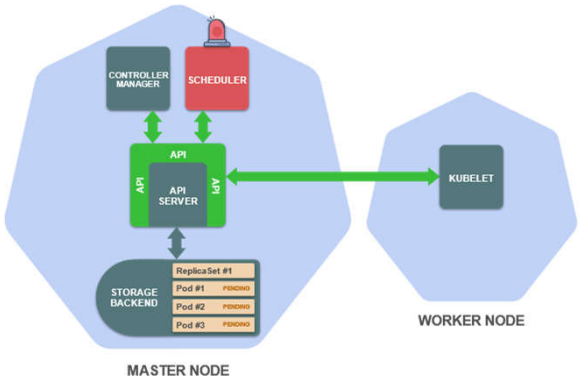
- The ReplicaSet controller creates a Pod definition for each replica of the ReplicaSet (according to the Pod template in the ReplicaSet definition) and saves them in the storage backend.



15

How Kubernetes works ?

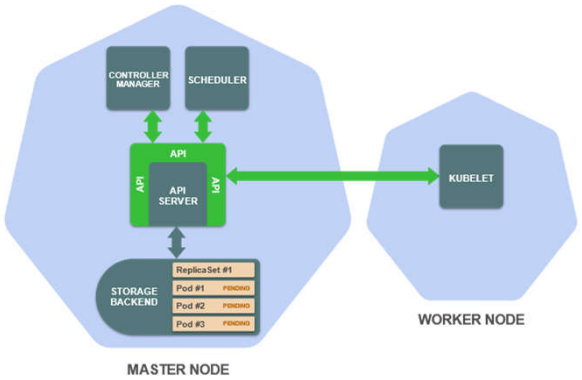
- This triggers the scheduler who watches for Pods that have not yet been assigned to a worker node.



16

How Kubernetes works ?

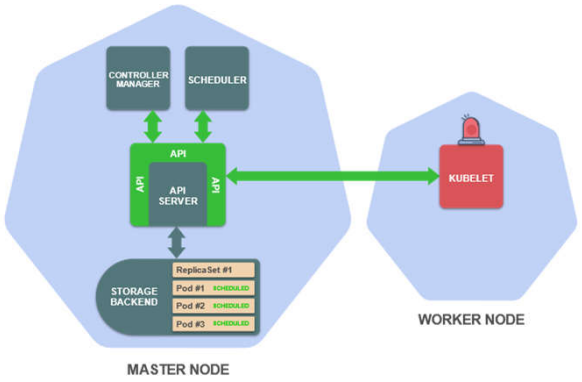
- The scheduler chooses a suitable worker node for each Pod and adds this information to the Pod definitions in the storage backend.



17

How Kubernetes works ?

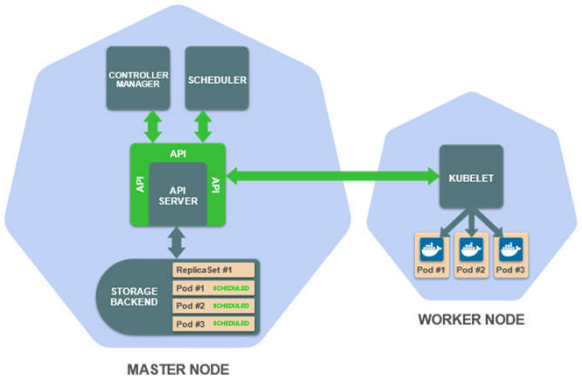
- This triggers the kubelet on the worker node that the Pods have been scheduled to, who watches for Pods that have been scheduled to its worker node.



18

How Kubernetes works ?

- The kubelet reads the Pod definitions from the storage backend and instructs the container runtime (Docker, for example) to run the containers on the worker node.



19

Kubernetes Concepts

20

Core Concepts

- **Namespace**
  - A logical cluster or environment. Primary method of dividing a cluster or scoping access.
- **Label**
  - Key-value pairs that are used to identify, describe and group together related sets of objects. Labels have a strict syntax and available character set.
- **Selector**
  - Selectors use labels to filter or select objects.

21

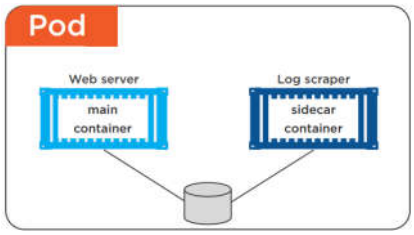
Workload Concepts

- **Pod**
  - A pod is the **smallest unit of work** or management resource within Kubernetes. It is comprised of one or more containers that share their storage, network, and context (namespace, cgroups, etc).
- **Replica Set**
  - Method of managing pod replicas and their lifecycle. Their scheduling, scaling, and deletion. Next Generation Replication Controller.
- **Deployment**
  - A declarative method of managing stateless Pods and Replica Sets. Provides rollback functionality in addition to more granular update control mechanisms.
- **Service**
  - Services provide a method of exposing and consuming Pod network accessible resources.

22

What is a Pod ?

- **A Pod is the basic building block of Kubernetes–the smallest and simplest unit in the Kubernetes object model that you create or deploy.**
  - A Kubernetes pod is a group of containers that are deployed together on the same host and share storage and networking resources. it's very common to have a group of containers work together to produce an artifact or process a set of work.

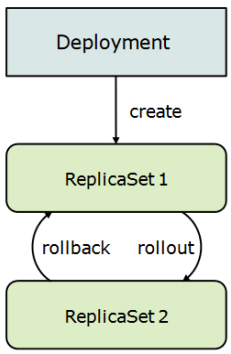


- **Containers within a Pod share an IP address and port space, and can find each other via localhost.**
- **Pods aren't intended to be treated as durable entities. They are ephemeral.**
- **Pods serve as unit of deployment, horizontal scaling, and replication.**

23

What is a deployment ?

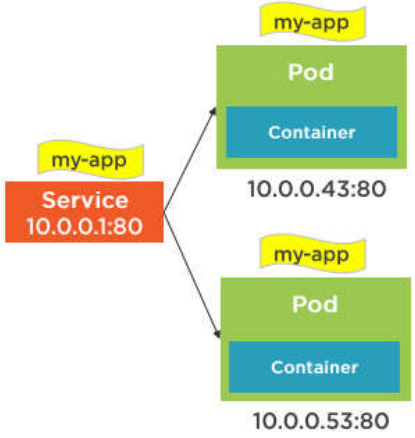
- **ReplicaSets**
  - A ReplicaSet is a set of Pod templates that describes a set of Pod replicas. It uses a template that describes what each Pod must contain. The ReplicaSet ensures that a specified number of Pod replicas are running at any time. As such, it is often used to guarantee the availability of a specified number of identical Pods.
- **Deployment**
  - A Deployment provides declarative updates for Pods and ReplicaSets.
  - You describe a desired state in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.



24

What is a Service ?

- **A service is an abstract way to expose an application running on a set of Pods as a network service.**
  - It acts as the unified method of accessing replicated pods.
- **Services provide important features that are standardized across the cluster:**
  - **Services abstract Pod IP Addresses from consumers**
  - **Load-balancing between Pods**
  - **Rely on labels to associate service with a Pod**
  - **Node's kube-proxy creates virtual IP for services**
  - **Services are not ephemeral**



25

Labels and Selectors

Deployment configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Deployment

ReplicaSet

Pod

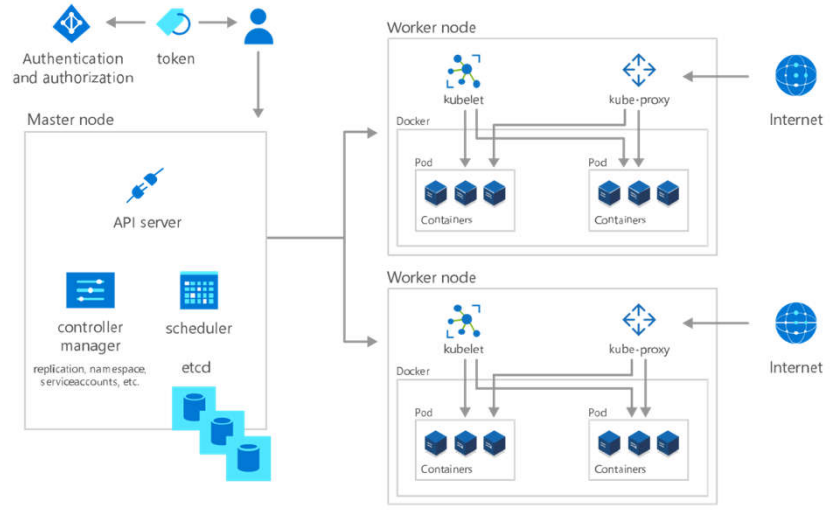
Service configuration:

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```


26

Understanding Kubernetes architecture

Kubernetes Architecture, Recap



M.Romdhani, 2020

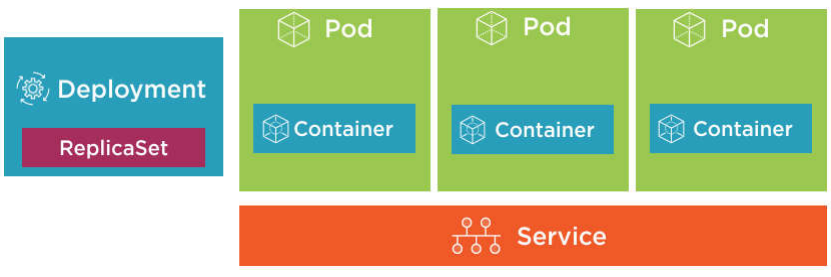
27

27

Understanding Kubernetes architecture

Workload Concepts, Recap

- **ReplicaSet ensures that a specified number of Pod replicas are running at any given time.**
- **Deployment is used to change the current state to the desired state.**



M.Romdhani, 2020

28

28

Start a Single node cluster locally with Docker Desktop for Windows

29

Kubernetes installation Options

- **There are many installation options**
  - Local installation
  - Managed installation
    - Amazon Elastic Container Service for Kubernetes
    - Azure Kubernetes Service
    - Google Kubernetes Engine
- **Local Installation Options on Windows**
  - Minikube
    - HyperV not required, unable to support Windows Containers
  - Docker Desktop for Windows
    - HyperV Required, supports Windows containers
- **Docker Desktop for Windows Community Edition**
  - Docker Desktop is the fastest and simplest way to get a Kubernetes cluster running on your desktop machine
  - It still give the freedom to choose Docker Swarm if you prefer.

30

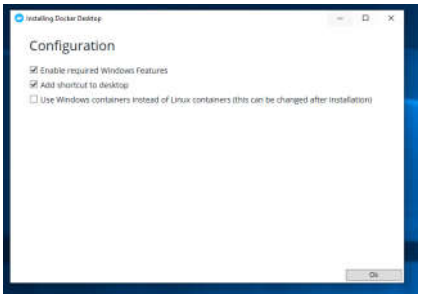
Installing Docker Desktop/Kubernetes

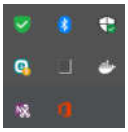
- **You can install Docker for Windows with:**
  - Chocolarey Installer

    ```
    > choco install docker-desktop -y
    ```

  - Docker Desktop for Windows installer
    - Available here : https://docs.docker.com/docker-for-windows/install

- **Brining Docker to Life**
  - Run the installer
  - Docker will ask you if you wish to enable Hyper-V
  - When the installer finishes, Select close and restart, this will restart your machine and start docker when windows starts.
  - Once Windows has restarted you will see a little animated docker icon in your taskbar which shows that docker is starting, this can take anything from 20 seconds to a couple of minutes.

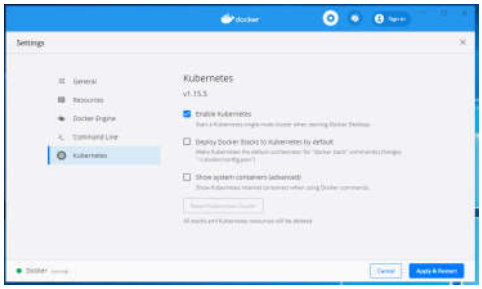




31

Installing Docker Desktop/Kubernetes

- **Enabling Kubernetes:**
  - Kubernetes can **then be enabled by simply clicking "Enable Kubernetes**" and then "**Apply & Restart**".
  - This will then download and initialise all the k8s containers required to get a local one node k8s cluster running on your local machine. Go and make a coffee this can take a long time(15–20min)



- **You can test that k8s is all running by using the kubectl command below.**

```
> kubectl get nodes
NAME             STATUS   ROLES    AGE    VERSION
docker-desktop   Ready    master   167m   v1.15.5
```

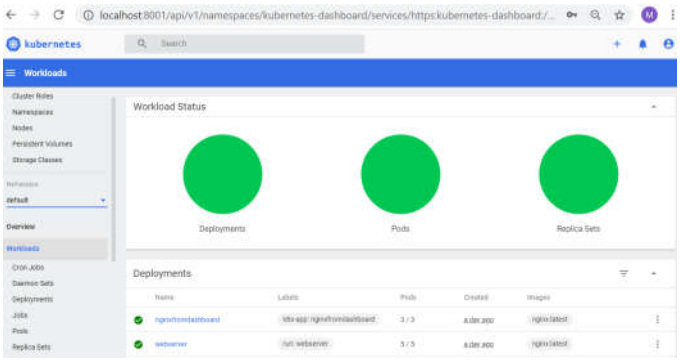

32

Using the Kubernetes dashboard

33

Kubernetes Dashboard

- **Dashboard is a web-based Kubernetes user interface.**
  - You can use Dashboard to deploy containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster resources.
  - You can use Dashboard to get an overview of applications running on your cluster, as well as for creating or modifying individual Kubernetes resources (such as Deployments, Services, etc).



34

Installing Kubernetes Dashboard

- **To deploy it, run the following command:**

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml
```

- **To access Dashboard from your local workstation, run the following command:**

```
$ kubectl proxy
```

  - It will proxy server between your machine and Kubernetes API server.

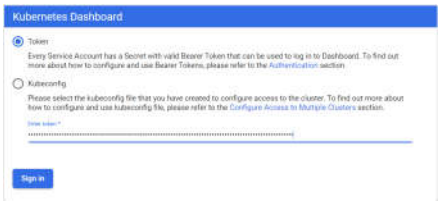
- **To view the dashboard in the browser, navigate to the following address in the browser of your Master VM:**

```
http://localhost:8001/api/v1/namespaces/kubernetes-
dashboard/services/https:kubernetes-dashboard:/proxy/
```

  - You will then be prompted with this page, to enter the credentials

- **Get an Authentication Token**
  - **Get account**

```
› kubectl -n kubernetes-dashboard describe
    secret default
```



35

Basic Commands of Kubectl

36

Kubectl commands format

- **Use the following syntax to run kubectl commands from your terminal window:**

  `kubectl [command] [TYPE] [NAME] [flags]`

  - where command, TYPE, NAME, and flags are:
    - **`command`**: Specifies the operation that you want to perform on one or more resources, for example **create**, **get**, **describe**, **delete**.
    - **`TYPE`**: Specifies the resource type. Resource types are case-insensitive and you can specify the singular, plural, or abbreviated forms

      kubectl get pod pod1

      kubectl get pods pod1

      kubectl get po pod1
    - **`NAME`**: Specifies the name of the resource. Names are case-sensitive. If the name is omitted, details for all resources are displayed, for example kubectl get pods.
    - **`flags`**: Specifies the additional flags, which are either specific for a command or global for kubectl. For example –namespace kube-system

  - Some commands, such as get or create, allow you to specify the output format using the **`-o`** or **`--output`** flag . For example **-o json** to force json output format

37

Kubectl command examples

- **Getting Information about Cluster**

  `kubectl version` Prints the client and server versions.

  `kubectl cluster-info` Prints information about the control plane and add-ons.

  `kubectl config get-contexts` Displays the list of cluster contexts

- **Getting information about resources**

  `kubectl get nodes/pods/deployements/secrets` Prints information about resources

  `kubectl describe nodes/pods/deployements/secrets` Prints detailed information about resources

- **Creating/Updating a Resource from Manifest**

  `kubectl create/apply -f my-nginx-deployment.yaml` Creates/Updates resources described in my-nginx-deployment.yaml

  `kubectl delete -f f my-nginx-deployment.yaml` Deletes resources described in my-nginx-deployment.yaml

- **Editing resources**

  `kubectl edit deployment my-nginx` Opens NotePad (on the editor configured in EDITOR ou KUBE_EDITOR env variable) with the current state of the resource. After editing and saving the resource will be updated.

- **Accessing Pod Container Logs**

  `kubectl logs etcd-docker-desktop –n kube-system` Prints the log of the etcd pod

38

Kubernets manifests examples

- **The required fields in the .yaml file:**
  - **apiVersion** - Which version of the Kubernetes API you're using to create this object
  - **kind** - What kind of object you want to create
  - **metadata** - Data that helps uniquely identify the object, including a name string, UID, and optional namespace
  - **spec** - What state you desire for the object

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```


Installing Kubernetes Dashboard

- **To deploy it, run the following command:**

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml
```

- **To access Dashboard from your local workstation, run the following command:**

```
$ kubectl proxy
```

  - It will proxy server between your machine and Kubernetes API server.

- **To view the dashboard in the browser, navigate to the following address in the browser of your Master VM:**

```
http://localhost:8001/api/v1/namespaces/kubernetes-
dashboard/services/https:kubernetes-dashboard:/proxy/
```

  - You will then be prompted with this page, to enter the credentials

Installing Kubernetes Dashboard

- **Create An Authentication Token**
  - **Create the dashboard service account**

```
› kubectl create serviceaccount admin-user -n kubernetes-dashboard
```

  - **Assign the cluster-admin role to the service account**

```
> kubectl create clusterrolebinding admin-user --clusterrole=cluster-admin --
serviceaccount=kubernetes-dashboard:admin-user
```

  - **List secrets using**

```
> kubectl -n kubernetes-dashboard get secret
```

  - **Use kubectl describe to get the access token:**

```
> kubectl -n kubernetes-dashboard describe  secret  admin-user-token-hpz44
```