Cognixia

A Collabera LEARNING SOLUTIONS COMPANY

Docker Container Clustering using Kubernetes

www.cognixia.com

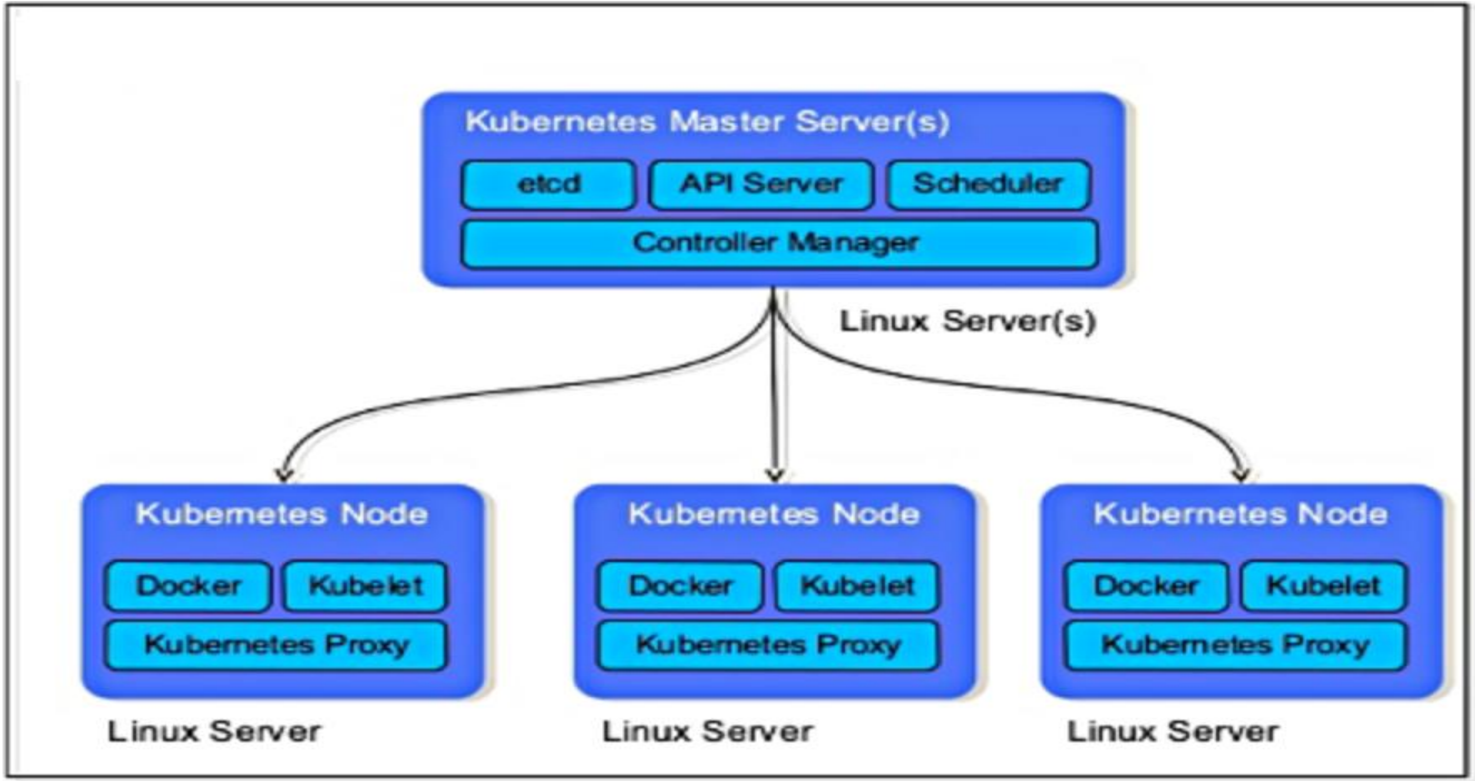
What is Kubernetes

- Kubernetes is a container management technology developed in Google lab
- Kubernetes in an open source container management tool hosted by Cloud Native Computing Foundation (CNCF)
- Kubernetes also called "K8s" is an open-source system for automating deployment, management and scaling of containerized applications
- It supports a range of container tools, including Docker
- Kubernetes v1.0 was released on July 21, 2015

Features of Kubernetes

- Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters
- Following are some of the important features of Kubernetes
  - Continues development, integration and deployment
  - Containerized infrastructure
  - Application-centric management
  - Auto-scalable infrastructure
  - Environment consistency across development testing and production
  - Loosely coupled infrastructure, where each component can act as a separate unit
  - Higher density of resource utilization
  - Predictable infrastructure which is going to be created

Kubernetes Architecture



Kubernetes – Master Machine Components

- Following are the components of Kubernetes Master Machine
  - etcd
    - Stores the configuration information which can be used by each of the nodes in the cluster
    - It is a high availability key value store that can be distributed among multiple nodes
    - It is accessible only by Kubernetes API server as it may have some sensitive information
    - It is a distributed key value Store which is accessible to all
  - API Server
    - Kubernetes is an API server which provides all the operation on cluster using the API
    - API server implements an interface, which means different tools and libraries can readily communicate with it
    - **Kubeconfig** is a package along with the server side tools that can be used for communication
    - It exposes Kubernetes API

Kubernetes – Master Machine Components Contd.

- Controller Manager
  - This component is responsible for most of the collectors that regulates the state of cluster and performs a task
  - It can be considered as a daemon which runs in nonterminating loop and is responsible for collecting and sending information to API server
  - It works toward getting the shared state of cluster and then make changes to bring the current status of the server to the desired state.
  - The key controllers are replication controller, endpoint controller, namespace controller, and service account controller.
  - The controller manager runs different kind of controllers to handle nodes, endpoints, etc.
- Scheduler
  - This is one of the key components of Kubernetes master
  - It is a service in master responsible for distributing the workload
  - It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload
  - This is the mechanism responsible for allocating pods to available nodes
  - The scheduler is responsible for workload utilization and allocating pod to new node

Kubernetes – Node Components

- Following are the key components of Node server which are necessary to communicate with Kubernetes master
  - Docker
    - The first requirement of each node is Docker which helps in running the encapsulated application containers in a relatively isolated but lightweight operating environment
  - Kubelet Service
    - This is a small service in each node responsible for relaying information to and from control plane service
    - It interacts with **etcd** store to read configuration details and right values
    - This communicates with the master component to receive commands and work
    - The **kubelet** process then assumes responsibility for maintaining the state of work and the node server
    - It manages network rules, port forwarding, etc
  - Kubernetes Proxy Service
    - This is a proxy service which runs on each node and helps in making services available to the external host
    - It helps in forwarding the request to correct containers and is capable of performing primitive load balancing
    - It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well
    - It manages pods on node, volumes, secrets, creating new containers’ health check up, etc

Kubernetes – Installation

Google Container Services

- The Google Cloud Platform offers a hosted Kubernetes-as-a-Service called Google Container Engine (GKE). To get started with GKE, you need a Google Cloud Platform account with billing enabled and the gcloud tool installed
- Once you have gcloud installed, first set a default zone:

```
gcloud config set compute/zone us-west1-a
```

- Then you can create a cluster:

```
gcloud container clusters create kuar-cluster
```

- When the cluster is ready you can get credentials for the cluster using:

```
gcloud auth application-default login
```

- At this point, you should have a cluster configured and ready to go

Installing Kubernetes with Azure Container Service

- Microsoft Azure offers a hosted Kubernetes-as-a-Service as part of the Azure Container Service
- The easiest way to get started with Azure Container Service is to use the built-in Azure Cloud Shell in the Azure portal
- You can activate the shell by clicking the shell icon
- Once you have the shell up and working, you can run:

```
        az group create --name=kuar --location=westus
```

- Once the resource group is created, you can create a cluster using:

```
az acs create --orchestrator-type=kubernetes --resource-group=kuar --name=kuar-cluster
```

- Once the cluster is created, you can get credentials for the cluster with:

```
        az acs kubernetes get-credentials --resource-group=kuar --name=kuar-cluster
```

- If you don’t already have the kubectl tool installed, you can install it using:

```
        az acs kubernetes install-cli
```

Installing Kubernetes on Amazon AWS

- Amazon Web Services (AWS) recently introduced a managed Kubernetes service called EKS
- It’s still under preview mode
- At the moment Kubernetes can be installed on AWS as explained in the Kubernetes documentation either using conjure-up, Kubernetes Operations (kops), CoreOS Tectonic or kube-aws

Installing Kubernetes Locally Using minikube

- If you need a local development experience, or you don’t want to pay for cloud resources, you can install a simple single-node cluster using minikube
- While minikube is a good simulation of a Kubernetes cluster, it is really intended for local development, learning, and experimentation
- Because it only runs in a VM on a single node, it doesn’t provide the reliability of a distributed Kubernetes cluster
- You can find the minikube tool on GitHub
- There are binaries for Linux, macOS, and Windows that you can download
- Once you have the minikube tool installed you can create a local cluster using:

```
minikube start
```

- This will create a local VM, provision Kubernetes, and create a local kubectl configuration that points to that cluster
- When you are done with your cluster, you can stop the VM with:

```
minikube stop
```

- If you want to remove the cluster, you can run:

```
minikube delete
```

Installing Kubernetes on Bare Metal

- We will use Amazon AWS cloud for native installation and setup of Kubernetes
- Exercise 1: Install Kubernetes on Bare Metal

Kubernetes - Kubectl

- Kubectl is the command line utility to interact with Kubernetes API
- **Kubectl** controls the Kubernetes Cluster
- It is one of the key components of Kubernetes which runs on the workstation on any machine when the setup is done
- It has the capability to manage the nodes in the cluster.
- **Kubectl** commands are used to interact and manage Kubernetes objects and the cluster

Kubernetes Images

- Kubernetes (Docker) images are the key building blocks of Containerized Infrastructure
- Each container in a pod has its Docker image running inside it
- When we are configuring a pod, the image property in the configuration file has the same syntax as the Docker command does
- The configuration file has a field to define the image name, which we are planning to pull from the registry

Kubernetes Images

- In order to pull the image and create a container, we will run the following command

```
kubectl create –f Testing_for_Image_pull
```

- Once we fetch the log, we will get the output as successful

```
kubectl log Testing_for_Image_pull
```

- The above command will produce an output of success or we will get an output as failure

Kubernetes Jobs

- The main function of a job is to create one or more pod and tracks about the success of pods
- Jobs ensure that the specified number of pods are completed successfully
- When a specified number of successful run of pods is completed, then the job is considered complete

Creating a Job

- We will create the job using the following command with yaml which is saved with the name **py.yaml**

```
kubectl create –f py.yaml
```

- The above command will create a job
- If you want to check the status of a job, use the following command

```
kubectl describe jobs/py
```

Scheduled Jobs

- Scheduled job in Kubernetes uses **Cronetes**, which takes Kubernetes job and launches them in Kubernetes cluster
- Scheduling a job will run a pod at a specified point of time
- A parodic job is created for it which invokes itself automatically
- We will use the same yaml which we used to create the job and make it a scheduled job
- This scheduled job concept is useful when we are trying to build and run a set of tasks at a specified point of time and then complete the process

Kubernetes – Labels & Selectors

- **Labels**
  - Labels are key-value pairs which are attached to pods, replication controller and services
  - They are used as identifying attributes for objects such as pods and replication controller.
  - They can be added to an object at creation time and can be added or modified at the run time
- **Selectors**
  - Labels do not provide uniqueness
  - In general, we can say many objects can carry the same labels
  - Labels selector are core grouping primitive in Kubernetes
  - They are used by the users to select a set of objects.
  - Kubernetes API currently supports two type of selectors –
    - Equality-based selectors
    - Set-based selectors

Namespace

- Namespace provides an additional qualification to a resource name
- This is helpful when multiple teams are using the same cluster and there is a potential of name collision
- It can be as a virtual wall between multiple clusters
- Following are some of the important functionalities of a Namespace in Kubernetes –
  - Namespaces help pod-to-pod communication using the same namespace
  - Namespaces are virtual clusters that can sit on top of the same physical cluster
  - They provide logical separation between the teams and their environments

Kubernetes - Node

- A node is a working machine in Kubernetes cluster which is also known as a minion
- They are working units which can be physical, VM, or a cloud instance
- Each node has all the required configuration required to run a pod on it such as the proxy service and kubelet service along with the Docker, which is used to run the Docker containers on the pod created on the node
- They are not created by Kubernetes but they are created externally either by the cloud service provider or the Kubernetes cluster manager on physical or VM machines
- The key component of Kubernetes to handle multiple nodes is the controller manager, which runs multiple kind of controllers to manage nodes
- To manage nodes, Kubernetes creates an object of kind node which will validate that the object which is created is a valid node

Node Controller

- They are the collection of services which run in the Kubernetes master and continuously monitor the node in the cluster on the basis of metadata.name
- If all the required services are running, then the node is validated and a newly created pod will be assigned to that node by the controller
- If it is not valid, then the master will not assign any pod to it and will wait until it becomes valid.
- Kubernetes master registers the node automatically, if –register-node flag is true

```
–register-node = true
```

- If the cluster administrator wants to manage it manually then it could be done by turning the flag off –

```
–register-node = false
```

Kubernetes - Service

- A service can be defined as a logical set of pods
- It can be defined as an abstraction on the top of the pod which provides a single IP address and DNS name by which pods can be accessed
- With Service, it is very easy to manage load balancing configuration
- It helps pods to scale very easily.
- A service is a REST object in Kubernetes whose definition can be posted to Kubernetes apiServer on the Kubernetes master to create a new instance

Kubernetes – Replication Controller

- Replication Controller is one of the key features of Kubernetes, which is responsible for managing the pod lifecycle
- It is responsible for making sure that the specified number of pod replicas are running at any point of time
- It is used in time when one wants to make sure that the specified number of pod or at least one pod is running
- It has the capability to bring up or down the specified no of pod
- It is a best practice to use the replication controller to manage the pod life cycle rather than creating a pod again and again

Kubernetes – Replica Sets

- Replica Set ensures how many replica of pod should be running
- It can be considered as a replacement of replication controller
- The key difference between the replica set and the replication controller is, the replication controller only supports equality-based selector whereas the replica set supports set-based selector

Kubernetes – Deployments

- Deployments are upgraded and higher version of replication controller
- They manage the deployment of replica sets which is also an upgraded version of the replication controller.
- They have the capability to update the replica set and are also capable of rolling back to the previous version
- They provide many updated features of **matchLabels** and **selectors**
- We have got a new controller in the Kubernetes master called the deployment controller which makes it happen
- It has the capability to change the deployment midway

Kubernetes – Autoscaling

- **Autoscaling** is one of the key features in Kubernetes cluster
- It is a feature in which the cluster is capable of increasing the number of nodes as the demand for service response increases and decrease the number of nodes as the requirement decreases
- This feature of auto scaling is currently supported in Google Cloud Engine (GCE) and Google Container Engine (GKE) and will start with AWS pretty soon
- In order to set up scalable infrastructure in GCE, we need to first have an active GCE project with features of Google cloud monitoring, google cloud logging, and stackdriver enabled

Cognixia
A Collabera LEARNING SOLUTIONS COMPANY

THANK YOU

www.cognixia.com