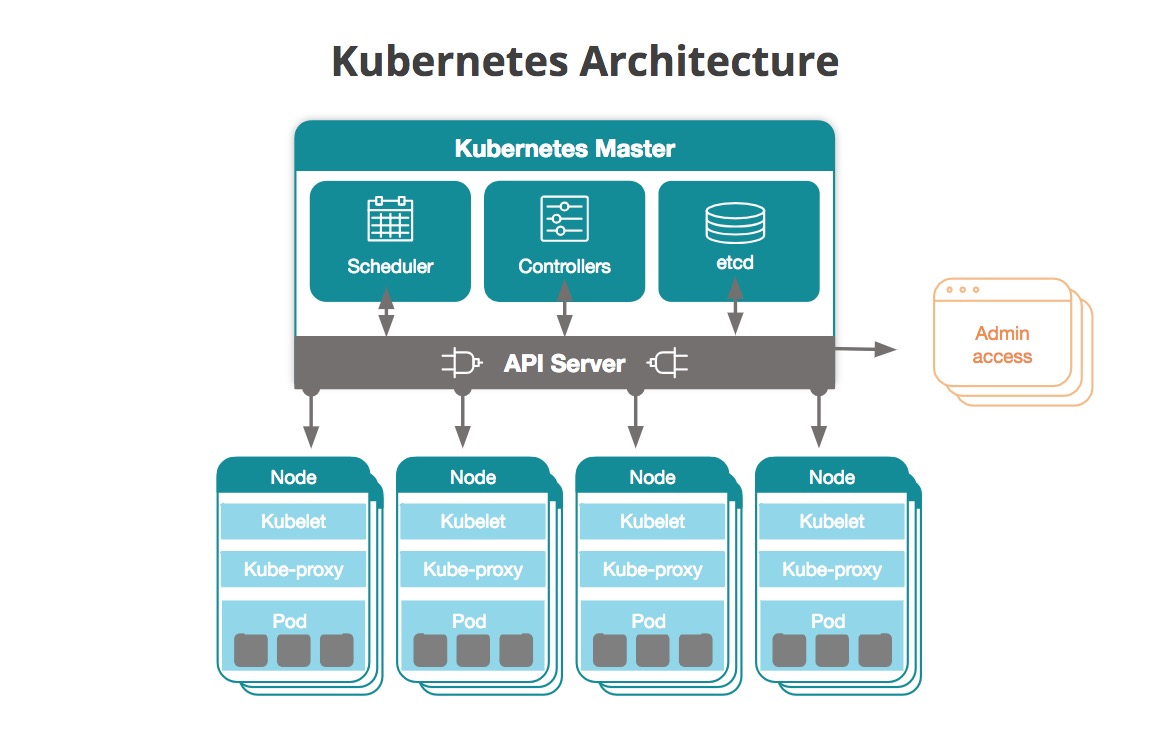
**Kubernetes**

**Introduction to Kubernates:**

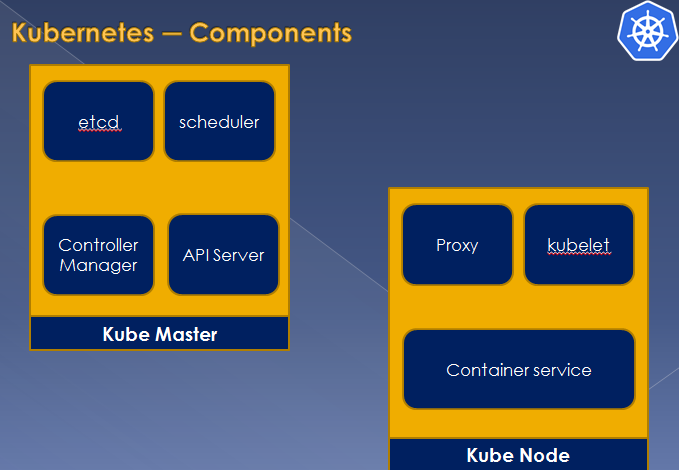
* Kubernetes in an open source container management tool hosted by Cloud Native Computing Foundation (CNCF).
* Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters.
* It is capable of creating container centric Infrastructure
* One of the key components of Kubernetes is, it can run application on clusters of physical and virtual machine infrastructure.
* It also has the capability to run applications on cloud.

**Kubernetes Architecture:**



**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



**Key Components:**

**etcd**

It 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.

**Installation on Ubuntu**

**Install Docker on all three nodes:**

1. Update apt-get repository for Docker installation

ubuntu@ip-172-31-90-147:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

ubuntu@ip-172-31-90-147:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

ubuntu@ip-172-31-90-147:~$ sudo apt-get update

ubuntu@ip-172-31-90-147:~$ sudo apt-cache policy docker-ce

ubuntu@ip-172-31-90-147:~$ sudo apt-get install -y docker-ce=18.06.1~ce~3-0~ubuntu

1. Verify that Docker is up and running with:

ubuntu@ip-172-31-90-147:~$ sudo systemctl status docker

ubuntu@ip-172-31-90-147:~$ docker version

Make sure the Docker service status is active (running)!

1. Add your username to the docker group to avoid typing sudo whenever you run the docker command

ubuntu@ip-172-31-90-147:~$ sudo usermod -aG docker ${USER}

**Install Kubeadm, Kubelet, and Kubectl on all three nodes.**

1. Install the Kubernetes components by running below commands on all three nodes:

ubuntu@ip-172-31-90-147:~$ sudo apt-get install apt-transport-https curl -y

ubuntu@ip-172-31-90-147:~$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

ubuntu@ip-172-31-90-147:~$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

ubuntu@ip-172-31-90-147:~$ sudo apt-get update

ubuntu@ip-172-31-90-147:~$ sudo swapoff -a

ubuntu@ip-172-31-90-147:~$ sudo apt-get install kubeadm -y

ubuntu@ip-172-31-90-147:~$ kubeadm version

ubuntu@ip-172-31-90-147:~$ sudo apt-get install -y kubelet kubectl kubernetes-cni

ubuntu@ip-172-31-90-147:~$

**Bootstrap the cluster on the Kube master node.**

1. **On the Kube master node, do this**: Kubernetes requires a Pod Network for the pods to communicate. For this guide we will use Flannel although there are several other Pod Networks available

sudo kubeadm init --pod-network-cidr=10.240.0.0/16

~~cidr = Make sure you should use your network cidr~~

Note: That command may take a few minutes to complete.

**Output looks like below**:

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.10.1.40:6443 --token rpqm5t.dc2ewkidfvzg61ly \

--discovery-token-ca-cert-hash sha256:71a16fca377b474a136b9b6bf0d28945c62e837c422196e0b8ecfe8075501275

1. When it is done, set up the local kubeconfig:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Take note that the kubeadm init command printed a long kubeadm join command to the screen. You will need that kubeadm join command in the next step!

1. Run the following commmand on the Kube master node to verify it is up and running:

kubectl version

This command should return both a Client Version and a Server Version.

**Join the two Kube worker nodes to the cluster.**

1. Copy the kubeadm join command that was printed by the kubeadm init command earlier, with the token and hash. Run this command on both worker nodes, but make sure you add sudo in front of it:

sudo kubeadm join $some\_ip:6443 --token $some\_token --discovery-token-ca-cert-hash $some\_hash

1. Now, on the Kube master node, make sure your nodes joined the cluster successfully:

kubectl get nodes

Verify that all three of your nodes are listed. It will look something like this:

NAME STATUS ROLES AGE VERSION

ip-10-0-1-101 NotReady master 30s v1.12.2

ip-10-0-1-102 NotReady <none> 8s v1.12.2

ip-10-0-1-103 NotReady <none> 5s v1.12.2

Note that the nodes are expected to be in the NotReady state for now.

**Set up cluster networking with flannel**

Once Kubernetes has been initialized we then install the Flannel Pod Network by running.

Let’s deploy the pod network, Pod network is the network through which our cluster nodes will communicate with each other. We will deploy Flannel as our pod network, Flannel will provide the overlay network between cluster nodes.

Turn on iptables bridge calls on all **three** nodes: here command is **sysctl** only not **systemctl**

sudo sysctl net.bridge.bridge-nf-call-iptables=1

1. Next, run this only on the **Kube master** node:

sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Now flannel is installed! Make sure it is working by checking the node status again:

kubectl get nodes

After a short time, all three nodes should be in the Ready state. If they are not all Ready the first time you run kubectl get nodes, wait a few moments and try again. It should look something like this:

NAME STATUS ROLES AGE VERSION

ip-10-0-1-101 Ready master 85s v1.12.2

ip-10-0-1-102 Ready <none> 63s v1.12.2

ip-10-0-1-103 Ready <none> 60s v1.12.2

**Deploying services on Kubernetes:**

* To get list of services

kubectl get services

kubectl describe service <servicename>

* get service ip

kubectl get svc <servicename>

* expose service for deployment

kubectl expose deployment nginx –port 80 –type NodePort

1 clear

2 kubectl create namespace sock-shop

3 git clone https://github.com/linuxacademy/microservices-demo.git

11 kubectl -n sock-shop create -f microservices-demo/deploy/kubernetes/complete-demo.yaml 13 kubectl get pods -n sock-shop

14 kubectl get deployments -n sock-shop

15 kubectl get deployments -n sock-shop -w

17 kubectl get services -n sock-shop

18 kubectl describe services -n sock-shop front-end

kubectl scale deployment/user --replicas=2 -n sock-shop

kubectl autoscale deployments catalogue -n sock-shop --min 2 --max 6 --cpu-percent 65

kubectl get hpa -n sock-shop

kubectl delete hpa -n sock-shop catalogue

Vertical Scalling: Increase the resource limits on an individual pod

HorizatalAutoScalling: Increase the number of pods across a cluster

**What is POD:**

A Kubernetes pod is a group of containers that are deployed together on the same host. If you frequently deploy single containers, you can generally replace the word "**pod**" with "**container**" and accurately understand the concept.

