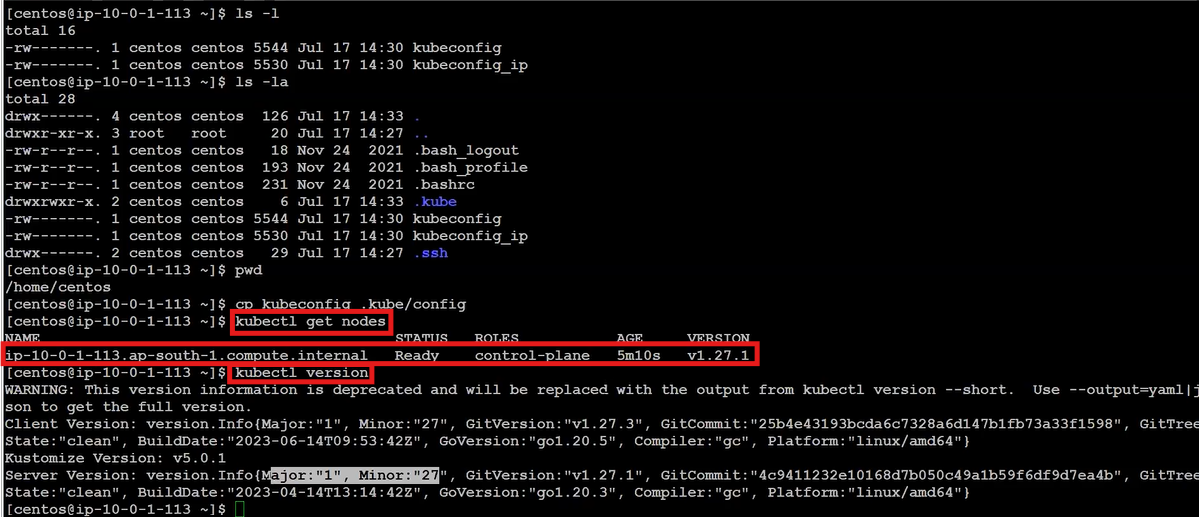
**KUBERNETES: -** It provides multiple services which docker doesn’t provide like auto scaling, auto healing, storage, enterprise level behaviour

**MINIKUBE KUBERNETES CLUSTER: -**

* Create a hosted zone in AWS
* Single node cluster 🡪 Master and the node components are in single server
* For minikube kubernetes we need to create a **VPC and Hosted DNS zone configured in Route53**

[**scholzj/terraform-aws-minikube: Terraform module for single node Kubernetes instance bootstrapped using kubeadm (github.com)**](https://github.com/scholzj/terraform-aws-minikube?tab=readme-ov-file) **🡪 For reference**

* Minikube uses centos7 and the username is centos
* Authentication config file = kubeconfig
* Default kubeconfig is in ~/.kube/config folder **(cp kubeconfig .kube/config)**

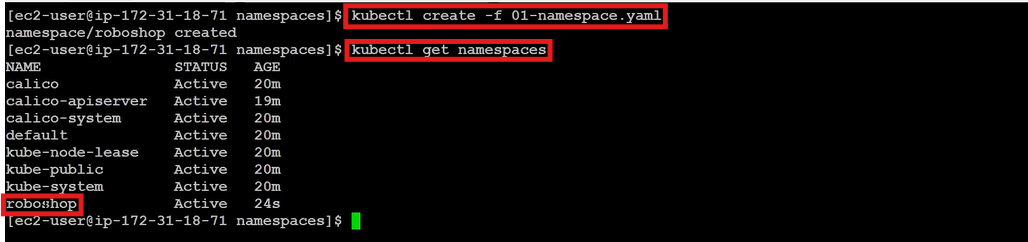
****

* Kubectl is a command line for Kubernetes cluster
* For giving the executive permissions and **connect using docker** run the following commands

1. Download and install **kubectl in workstation** using command from internet
2. Chmod +x ./kubectl
3. Sudo mv kubectl /usr/local/bin
4. Kubectl
5. Cat kubeconfig 🡪 **Copy the configuration file from minikube cluster (k8 ec2)**
6. mkdir .kube
7. cd .kube/
8. vim config 🡪 **paste the copied configuration file in workstation (docker ec2)**
9. kubectl get nodes 🡪 **we need to get the nodes present inside the cluster**

**Kubernetes Resources: -**

1. Namespace **(kubectl get namespaces): -** In Kubernetes namespace is a logical isolation of resources, network, policies, rbac and everything EX: - If 2 projects using same k8 cluster, one can use ns1 and other can use ns2 without any overlap

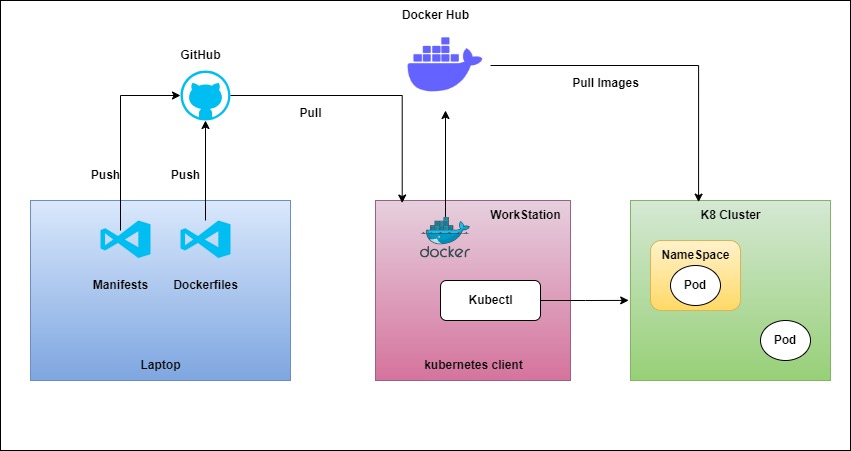
****

* **Kubectl create -f <file-name.yaml> 🡪** To create the resources
* **Kubectl apply -f <file-name.yaml> 🡪** if it’s not created, it will create. If created before no error
* **Kubectl delete -f <file-name.yaml> 🡪** To delete the resources

1. **Pod (kubectl get pods): -** It is the smallest deployable unit in Kubernetes.

* Pod can run multiple containers
* Containers inside the pod share the same network and storage

**Kubectl exec -it multipod -c almalinux –bash🡪go into the container**



Kubernetes architecture has two nodes

1. Master node (control plane)

It contains of components like

* **API Server: -** It is useful for exposing the Kubernetes to the external world. It is heart of the kubernetes
* **ETCD: -** It is used as the backup which will store as key pair values.
* **Scheduler: -** It will schedule the pods or the resources in Kubernetes. It will the information from api server to schedule
* **Controller manager: -** Replicaset is used to maintain the pod as the pods are always running. Maintaining these replicasets is by controller manager
* **Cloud Controller Manager: -** If we want to implement the Kubernetes in any cloud provider. The people of the cloud itself contributes to the cluster

1. Worker node (Data Plane)

It contains of components like

* **Kubelet: -** It is responsible running and managing the pod
* **Kubeproxy: -** It is a networking in Kubernetes which allocates the ip to pods and load balancing

**Kubeproxy works by maintaining a set of network rules on each node in the cluster, which are updated dynamically as services are added or removed. When a client sends a request to a service, the request is intercepted by kube-proxy on the node where it was received. Kube-proxy then looks up the destination endpoint for the service and routes the request accordingly.**

**Kube-proxy is an essential component of a Kubernetes cluster, as it ensures that services can communicate with each other**

* **Container runtime: -** It will run the container

**Kubernetes is a cluster**

* The lowest level of deployment in Kubernetes is a **pod**
* **POD: -** It defines how to run a pod
* We can put multiple containers in a single pod with multiple advantages

1. Shared storage
2. Shared networking
3. And they communicate using a local host

**\*\* Kubectl is a command line for Kubernetes which is useful to run the Kubernetes commands**

**KUBERNETES POD:**

**Ex: - kubectl get pods --> It will give the list of pods**

* **To install the kubectl -->** [Install Tools | Kubernetes](https://kubernetes.io/docs/tasks/tools/)
* **Kubectl –version -->** to check the kubectl is installed or not
* **To install minikube -->** [minikube start | minikube (k8s.io)](https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download)
* **Minikube start -->** Kubernetes cluster will start
* If we are using windows or mac then **minikube start** will first create the virtual machine and then creates a single node Kubernetes cluster
* Needs to create a pod.yml file where we have written all the requirements
* Then **kubectl apply -f pod.yml / kubectl cretae -f pod.yml -->** pod is got created
* **Kubectl get pods -->** get the list of pods
* Reference of the yaml file to create pod --> [Pods | Kubernetes](https://kubernetes.io/docs/concepts/workloads/pods/)
* **Kubectl describe pod <podname> -->** To know complete details about the pod
* **minikube ssh -->** To get inside the Kubernetes cluster
* **curl <ip of pod> -->** To get complete details of the cluster
* **Kubectl cmnds reference -->**[kubectl Cheat Sheet - Kubernetes (k8s-docs.netlify.app)](https://k8s-docs.netlify.app/en/docs/reference/kubectl/cheatsheet/)
* To get the properties like auto scaling and auto healing deployment is kind we need to use instead of pod

**KUBERNETES DEPLOYMENT:**

**Container: -** It is used in docker, we use **docker run -d <imageid> -p <port>** to run a container

**POD:** - It is used in kubernetes and manifest a yaml file where we can build multiple containers. Using the yaml file itself we will run the containers. It is running specification of docker container

**Deployment:** - It is responsible for the special features that a Kubernetes have like Auto Healing, Auto Scaling etc which pods doesn’t provide

* The Deployment first create a replicaset which is a **Kubernetes controller.** It is the one which is implementing the autohealing
* Replicasets creates pods
* **Kubectl get deploy -->** It will list all the deployments
* **Kubectl get all -->** It will list all the pods,deployments,resources etc: -
* Reference of The yaml file to create pod --> [Deployments | Kubernetes](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)
* **Kubectl get rs -->** It will list all the replicasets

**KUBERNETES SERVICES:**

* Kuberenets services is created on top of the replicaset. **Service is a load balancer(svc)**
* If there are 3 pods then the traffic will be shared accordingly to the pods
* With the help of the ip addresses the user can access instead of ip addresses service will be as the url to access no matter what the ip addresses are
* Inorder to ignore the ip’s when the time of creating the pod **labels** & **selectors** is added to the pod. EX: - if the application is about the payments, then the label name as a payment. This label should be declared in a metadata of a pod creation
* We can create service as 3 types using yaml manifest

1. Cluster IP

2. nodeport

3. load balancing

* If we create a service using **cluster ip** mode, it will not change anything. It will only give the two benefits which are **discovery and load balancing. It can be accessed only inside the cluster**
* If we create a service using **nodeport** mode then the application is accessed **inside the org**
* If we create a service using **load balancer** mode then the application is accessed by **everyone outside the world.** This will only support on cloud provider

**Advantages of Kubernetes services**

1. Load Balancing
2. Service Discovery
3. Expose to External world