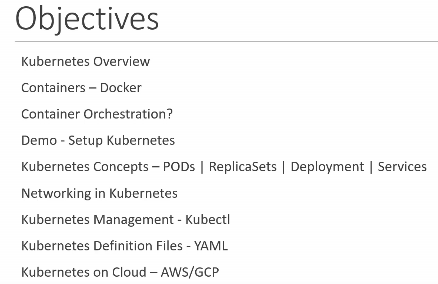
**KUBERNETES:**



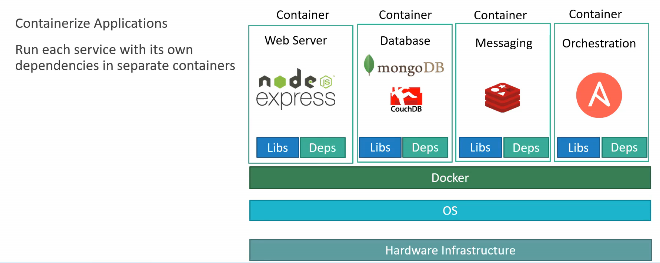
[**https://www.cncf.io/certification/ckad/**](https://www.cncf.io/certification/ckad/)

The online, proctored, performance-based test consists of a set of performance-based items (problems) to be solved in a command line and is expected to take approximately two (2) hours to complete.

This exam curriculum includes these general domains and their weights on the exam:

* 13% – Core Concepts
* 18% – Configuration
* 10% – Multi-Container Pods
* 18% – Observability
* 20% – Pod Design
* 13% – Services & Networking
* 8% – State Persistence

**Container:**



**Advantage of using contianers:**



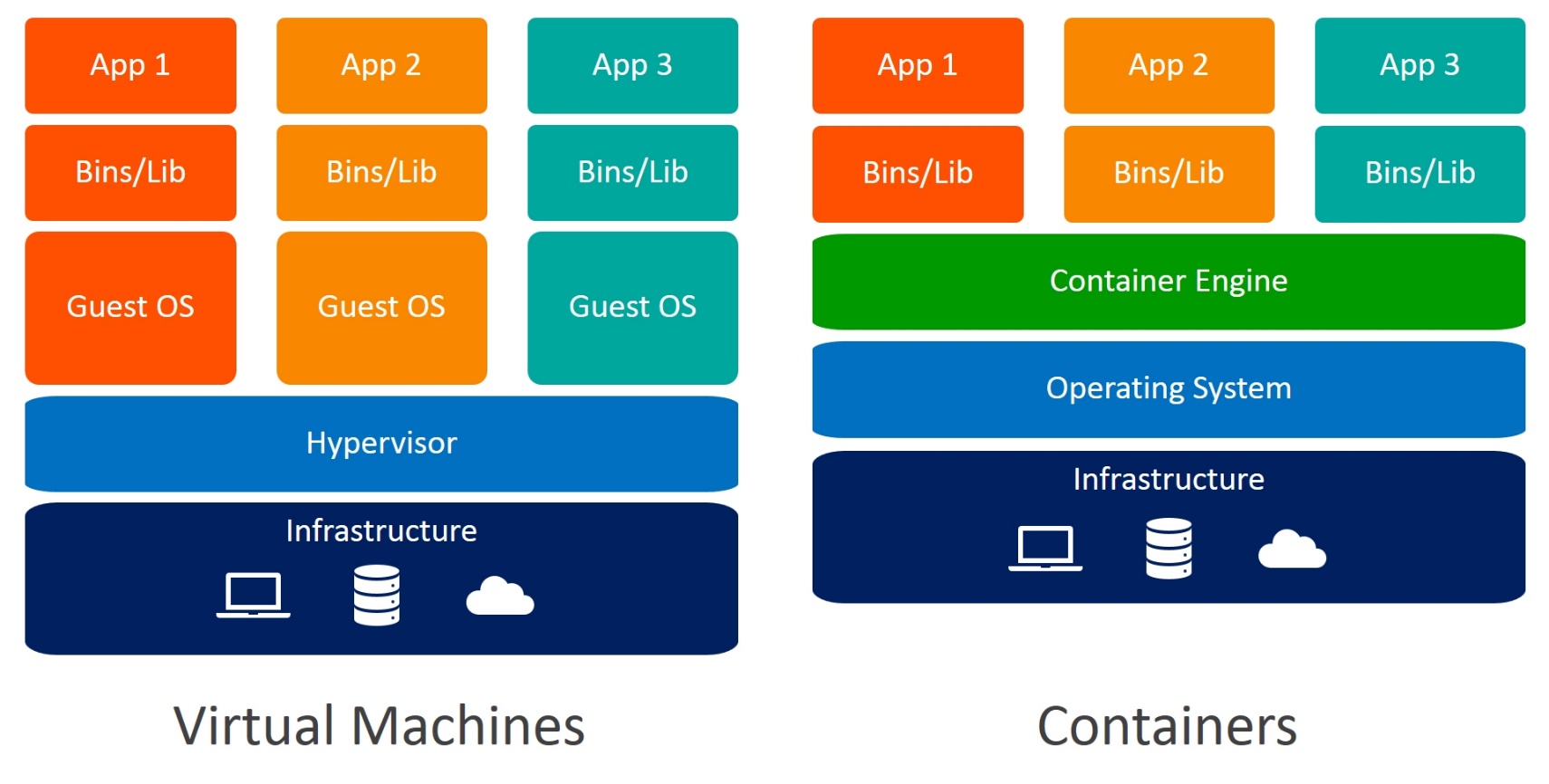
* Docker use LXC container

**LXC** (**Linux Containers**) is an [operating-system-level virtualization](https://en.wikipedia.org/wiki/Operating-system-level_virtualization) method for running multiple isolated [Linux](https://en.wikipedia.org/wiki/Linux) systems (containers) on a control host using a single Linux kernel.

The [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) provides the [cgroups](https://en.wikipedia.org/wiki/Cgroups) functionality that allows limitation and prioritization of resources (CPU, memory, block I/O, network, etc.) without the need for starting any [virtual machines](https://en.wikipedia.org/wiki/Virtual_machine), and also [namespace isolation](https://en.wikipedia.org/wiki/Linux_namespaces) functionality that allows complete isolation of an application's view of the operating environment, including [process](https://en.wikipedia.org/wiki/Process_(computing)) trees, [networking](https://en.wikipedia.org/wiki/Computer_network), [user IDs](https://en.wikipedia.org/wiki/User_ID) and [mounted](https://en.wikipedia.org/wiki/Mount_(computing)) [file systems](https://en.wikipedia.org/wiki/File_system).[[3]](https://en.wikipedia.org/wiki/LXC#cite_note-3)

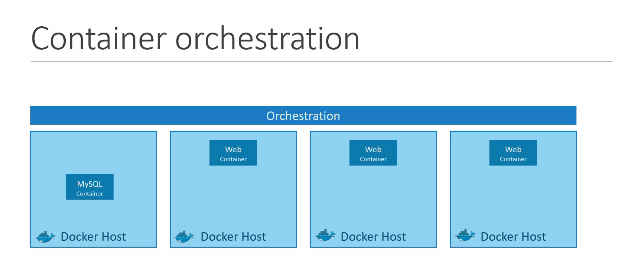
LXC combines the kernel's cgroups and support for isolated [namespaces](https://en.wikipedia.org/wiki/Namespace) to provide an isolated environment for applications. Early versions of [Docker](https://en.wikipedia.org/wiki/Docker_(software)) used LXC as the container execution driver, though LXC was made optional in v0.9 and support was dropped in Docker v1.10.

**Contianer vs virtualization:**



* **Docker image**
* **Docker file**

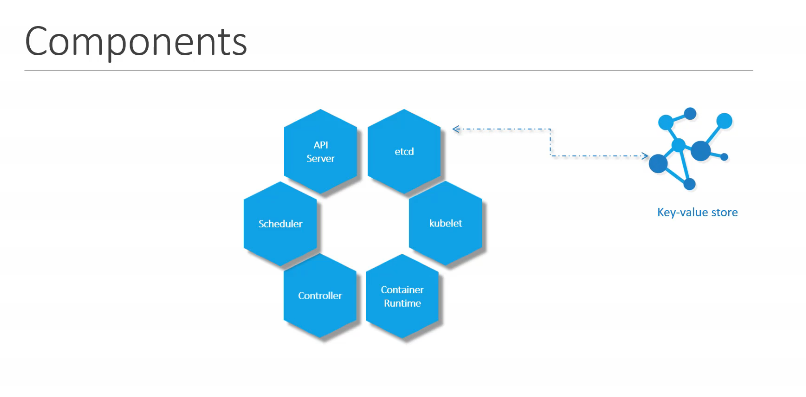
**Contianer orchestration:**

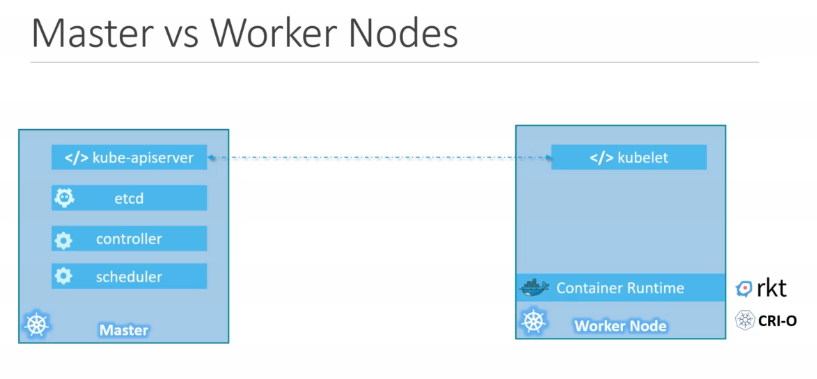


* **Common Platform has to manage the application, its connectivity and scalability is called container orchestration**
* **KUBERNETES is a Container orchestration technology**

1. **It is use Deployment and Management of container in cluster environment.**

**KUBERNETTES Architecture:**



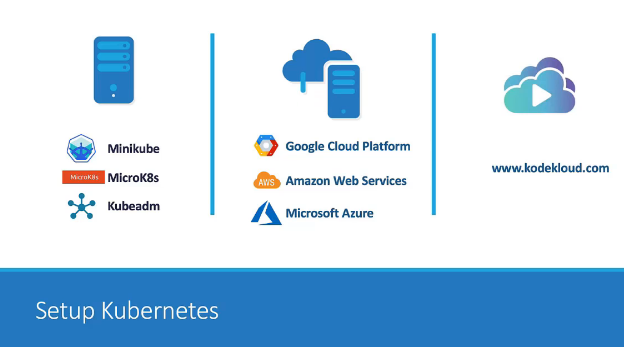


Basic command:

**Kubectl cluster-info**

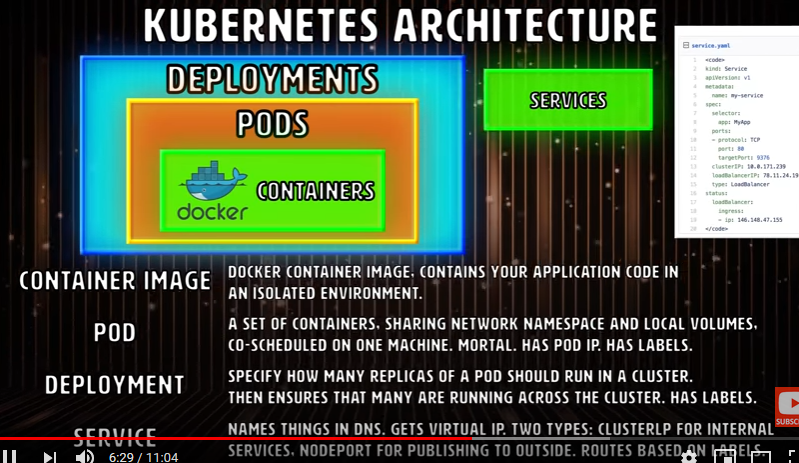
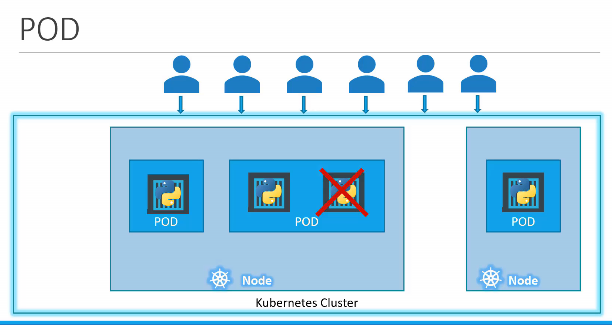
**Kubectl get node**

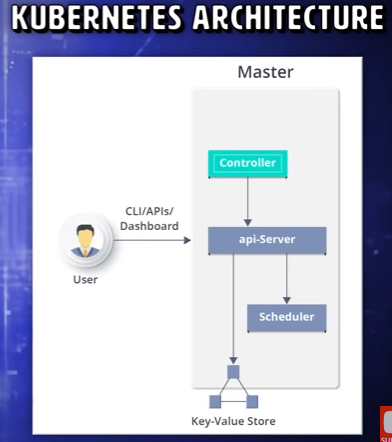
**Kubectl run <application>**

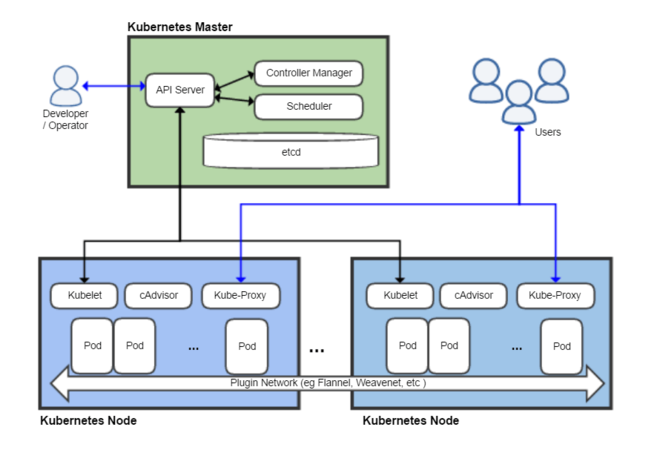


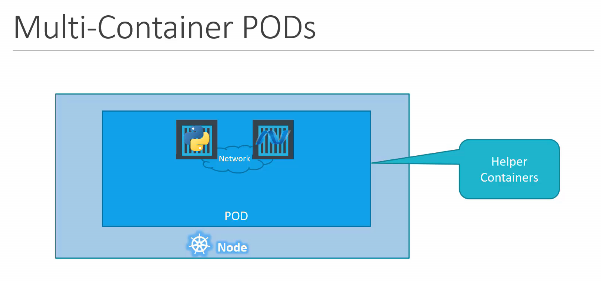
[**https://www.kodekloud.com/**](https://www.kodekloud.com/)

[**https://kodekloud.com/p/public-labs?scenario=kubernetes-for-beginners-basiccommands-test**](https://kodekloud.com/p/public-labs?scenario=kubernetes-for-beginners-basiccommands-test)

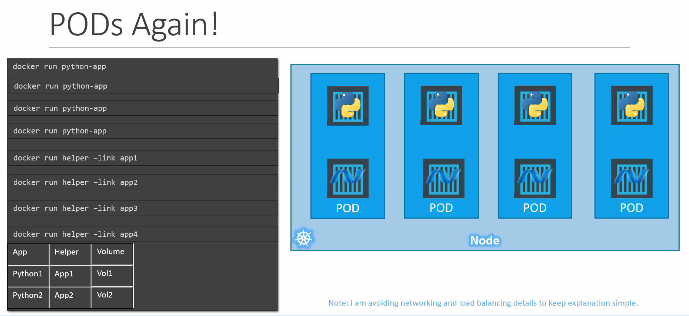




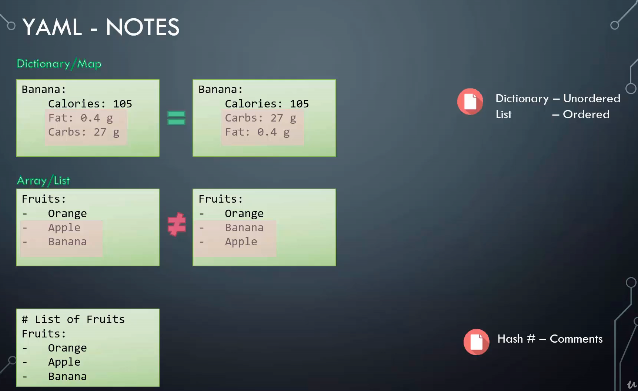




1. Worker machine in Kubernetes known as **NODE or minions.**
2. Multiple nodes are together to form a **cluster.**
3. **Etcd** is distributed reliable key-value store used by Kubernetes to store all data used to manage the cluster.
4. **Schedular** services is responsible for distributing work or containers across multiple nodes.
5. **Container runtime** is the underlying framework that is responsible for running application in containers like Docker.
6. **Kubectl** is the command line utility used to manage a Kubernetes cluster.
7. The smallest unit you can create in Kubernetes object model is **POD.**
8. **Minikube etc** is used to run Kubernetes in local environment.
9. **Google container engine** is an instant way of setting up a permanent kubernetes cluster on the cloud
10. **Kubeadm** is the solution is used for setting up a multi-node Kubernetes cluster in a local environment



**YAML:**

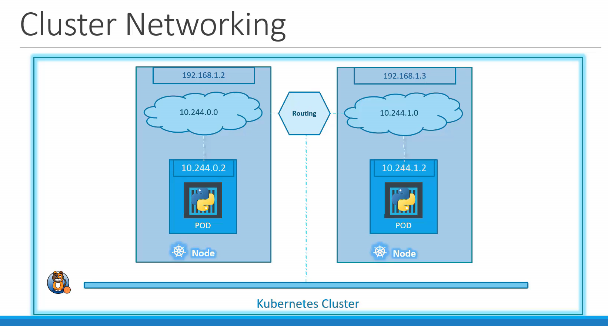


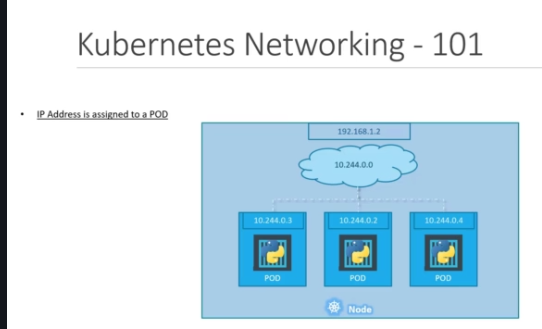
**List of Kubernetes model:**

* **Network**
* **POD**
* **Service**
* **Deployment**
* **Replica set**
* **Ingress**

**Network:**

* **Ip address is assigned to pod**
* **Kubernetes intial assign a private network**
* **Cluster networking**



****

[**https://kubernetes.io/docs/tutorials/hello-minikube/**](https://kubernetes.io/docs/tutorials/hello-minikube/)

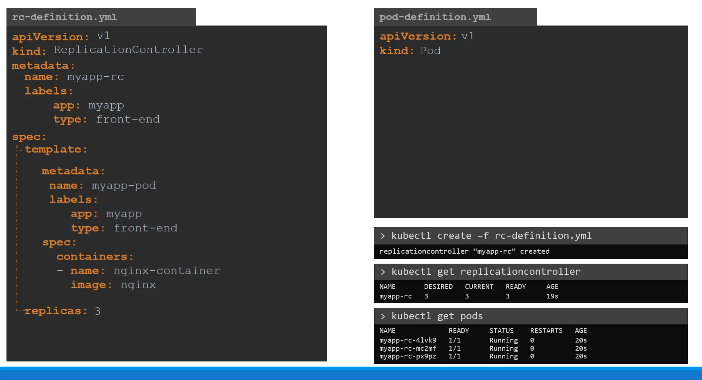
[**https://minikube.sigs.k8s.io/docs/start/**](https://minikube.sigs.k8s.io/docs/start/)

**Kubectl run redis –image=redis123 –dry-run=client –o yaml > pod.yaml**

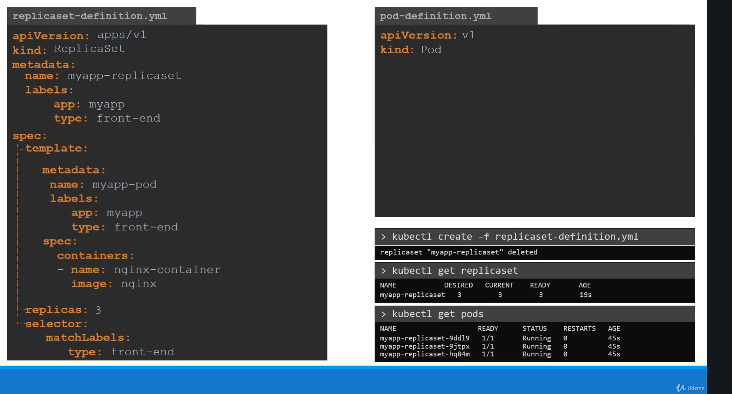
**Kubectl edit pods <podname>**

**-----------------------------------------------------------------------------------------------------------------------------------------------------------**

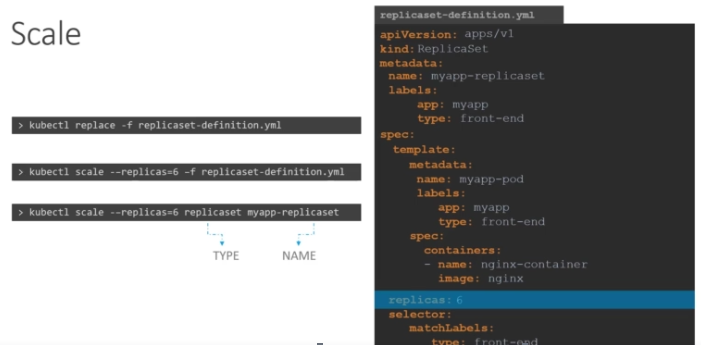
* **Replication Controller:**

****

* **Replica set:**

****

**Scale**

****

****

**Deployments:**

**Deployed**

**Upgrade**

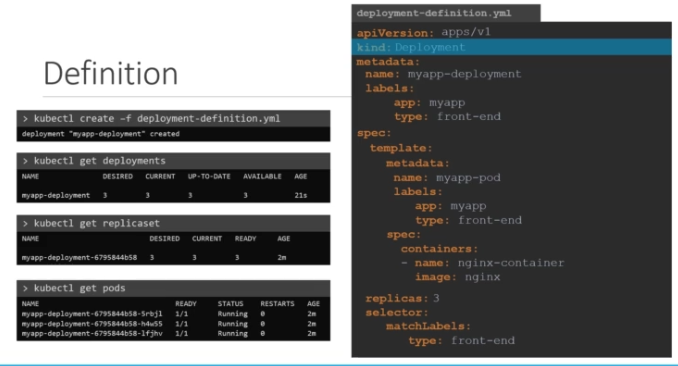
**Rolling update**

**Rollback**

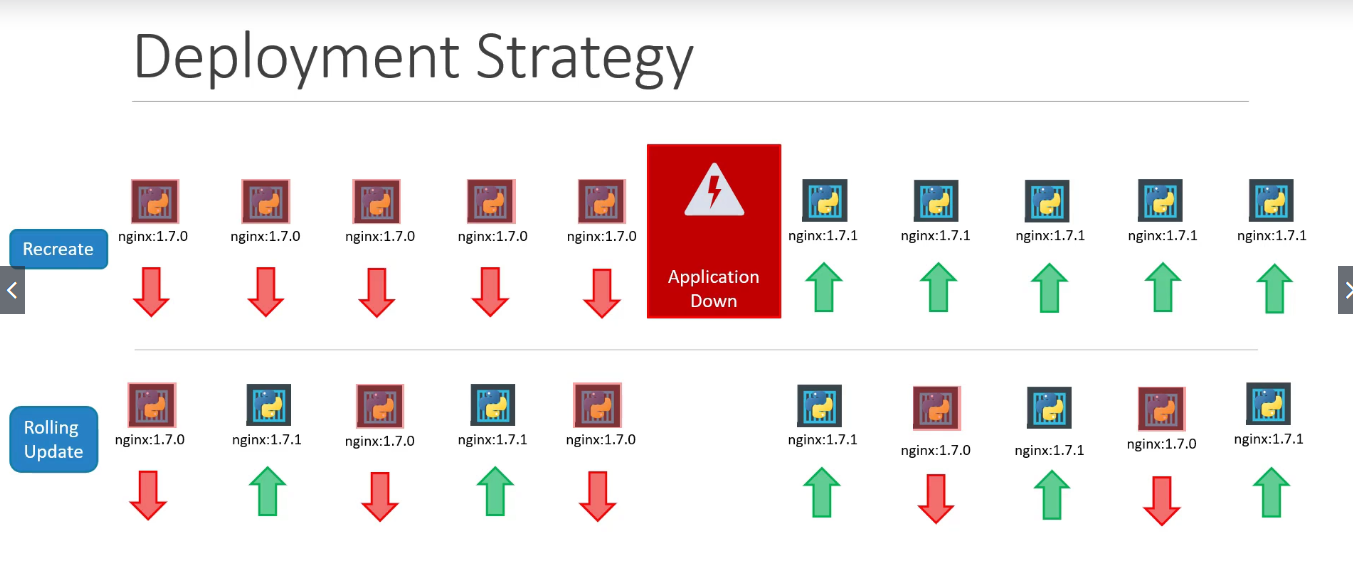
**Pause the application**

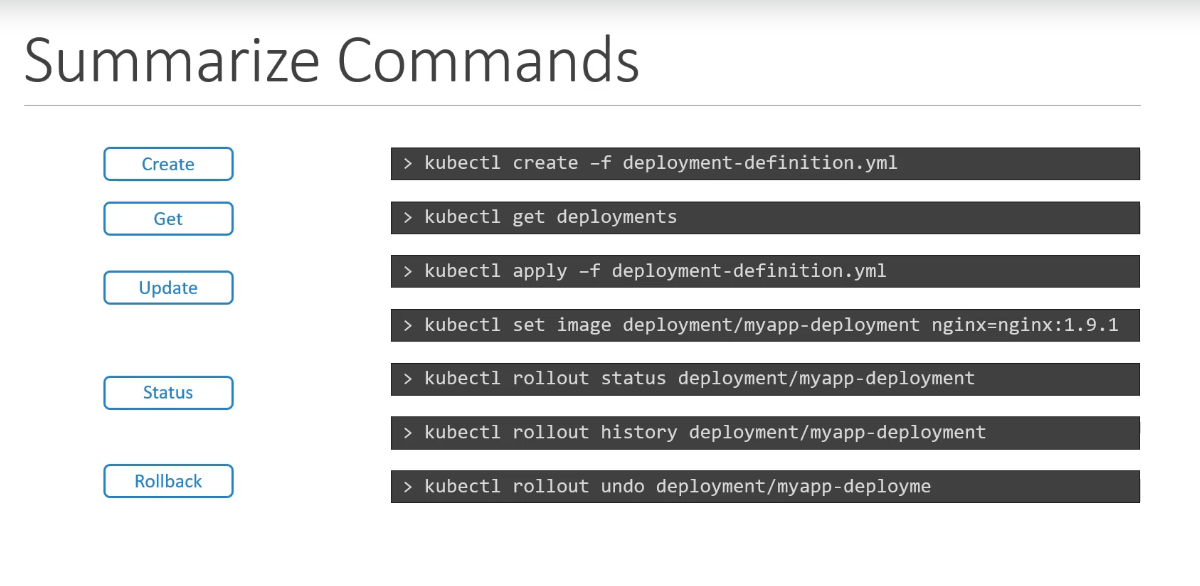
**Resume the application**

****

****

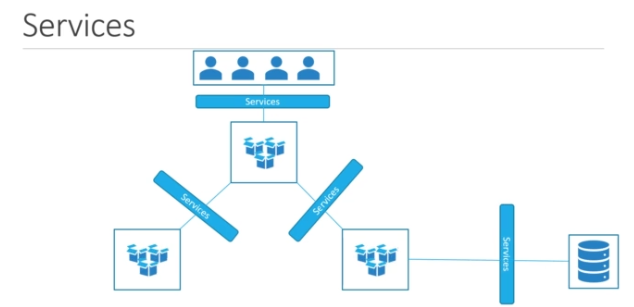
**Deployment update and rollout:**

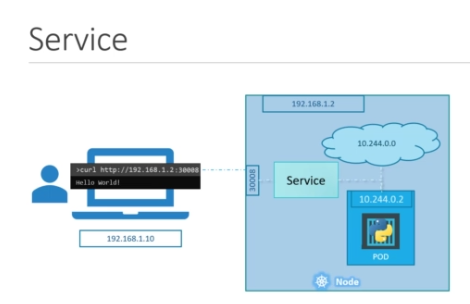
****

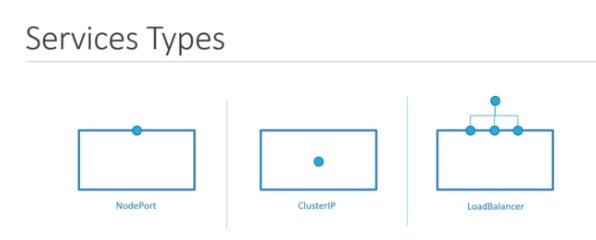
****

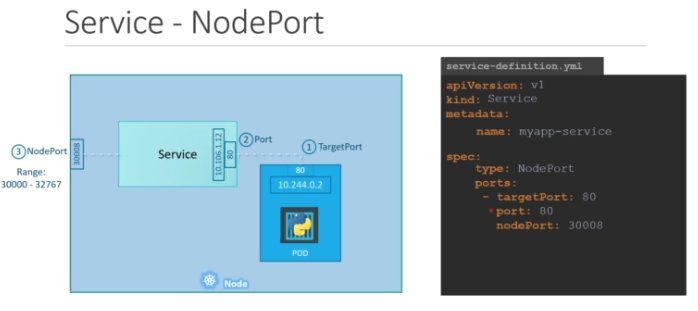
**Kubectl set image deployment <deployment-name> imagename=image –record**

**Services:**

****

****

****

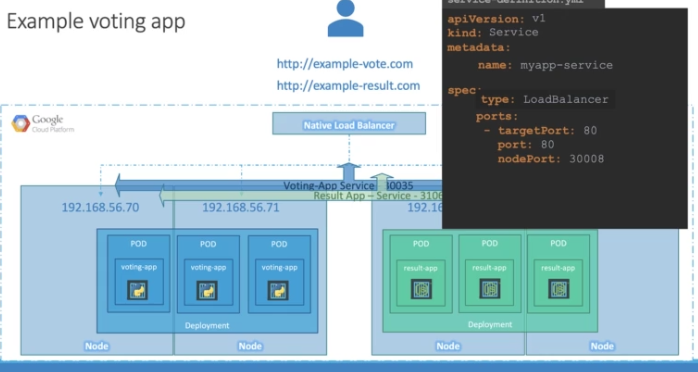
****

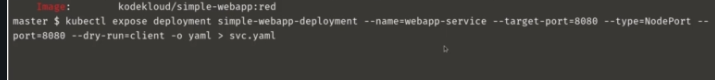
**To get the ipaddress of the node:**

* **Kubectl get node –o yaml**
* **Minikube ip**
* **minikube service <service-name> -- url**
* **Pod of the single node**
* **Multiple pod in single node**
* **Multiple pod in multiple node**

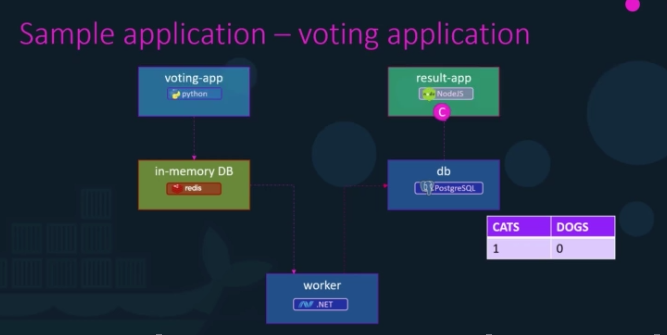
**Service will act upon in the stable manner**

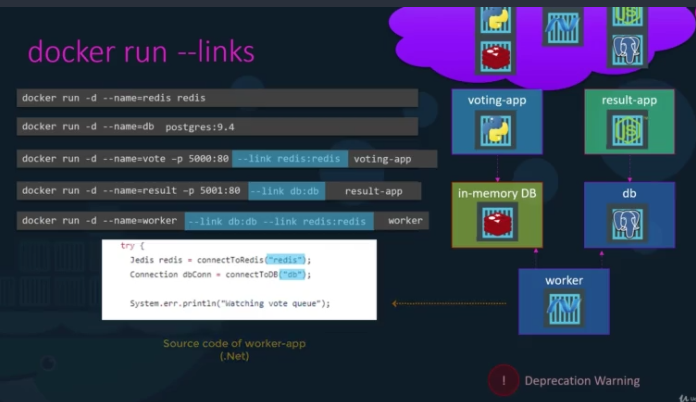
**Load Balancer:**

****

****

**Micro service Architecture:**

****

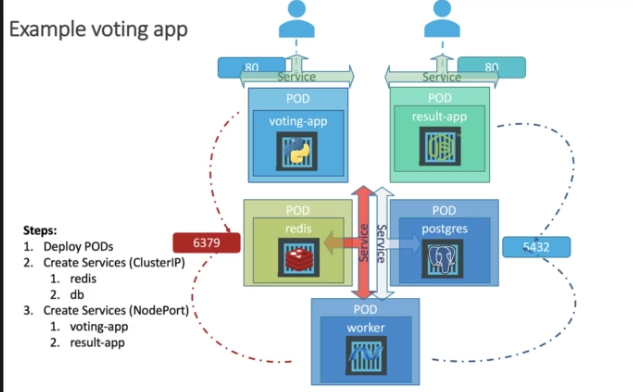
****

**GOAL:**

**1) Deploy application (PODs)**

**2) Enable connectivity**

**3) External Access**

****