1. **What is Kubernetes?**

* Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
* It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

1. **Why Kubernetes?**

* Kubernetes provides an easy way to scale your application, compared to virtual machines.
* It keeps code operational and speeds up the delivery process.
* Kubernetes API allows automating a lot of resource management and provisioning tasks.
* Building on top of Kubernetes may also prepare you for cloud migration in the future.

1. **What Kubernetes can do?**

* Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start.

1. **What is the difference between Kubernetes and Docker Swarm?**

* Docker swarm is a default container orchestration tool that comes with Docker.
* Docker swarm can only orchestrate simle Docker containers
* Kubernetes on the other hand helps mange much more complex software application containers.
* Kubernetes offers support for large demand production environment.
* Docker swarm can’t do auto scalling
* Docker Swarm doesn’t have a GUI
* Docker can deploy rolling updates but can’t deploy automatic rollbacks
* Docker requires third-party tools like ELK stack for logging and monitoring while Kubernetes has integrated tools for the same
* Docker Swarm can share storage volumes with any container easily while Kubernetes can only share storage volumes with containers in the same pod.

1. **What is a Heapster?**

* In a Kubernetes cluster, application performance can be examined at many different levels: containers, [pods](https://kubernetes.io/docs/user-guide/pods), [services](https://kubernetes.io/docs/user-guide/services), and whole clusters.
* As part of Kubernetes we want to provide users with detailed resource usage information about their running applications at all these levels.
* This will give users deep insights into how their applications are performing and where possible application bottlenecks may be found.
* In comes [Heapster](https://github.com/kubernetes/heapster), a project meant to provide a base monitoring platform on Kubernetes.
* Heapster is a cluster-wide aggregator of monitoring and event data.
* It currently supports Kubernetes natively and works on all Kubernetes setups.
* Heapster runs as a pod in the cluster, similar to how any Kubernetes application would run.
* The Heapster pod discovers all nodes in the cluster and queries usage information from the nodes’ [Kubelets](https://github.com/kubernetes/kubernetes/blob/master/DESIGN.md" \l "kubelet), the on-machine Kubernetes agent. The Kubelet itself fetches the data from [cAdvisor](https://github.com/google/cadvisor).
* Heapster groups the information by pod along with the relevant labels. This data is then pushed to a configurable backend for storage and visualization.

1. **What is kubelet?**

* The Kublet is a service agent that controls and maintains a set of pods by watching pod specs through the Kubernetes API server.
* It preserves the pod lifecycle by ensuring that a given set of containers are all running as they should.
* The kublet runs on each node and enables the communication between master and slave nodes.
* **Refer the link:** https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

1. **What is kubectl?**

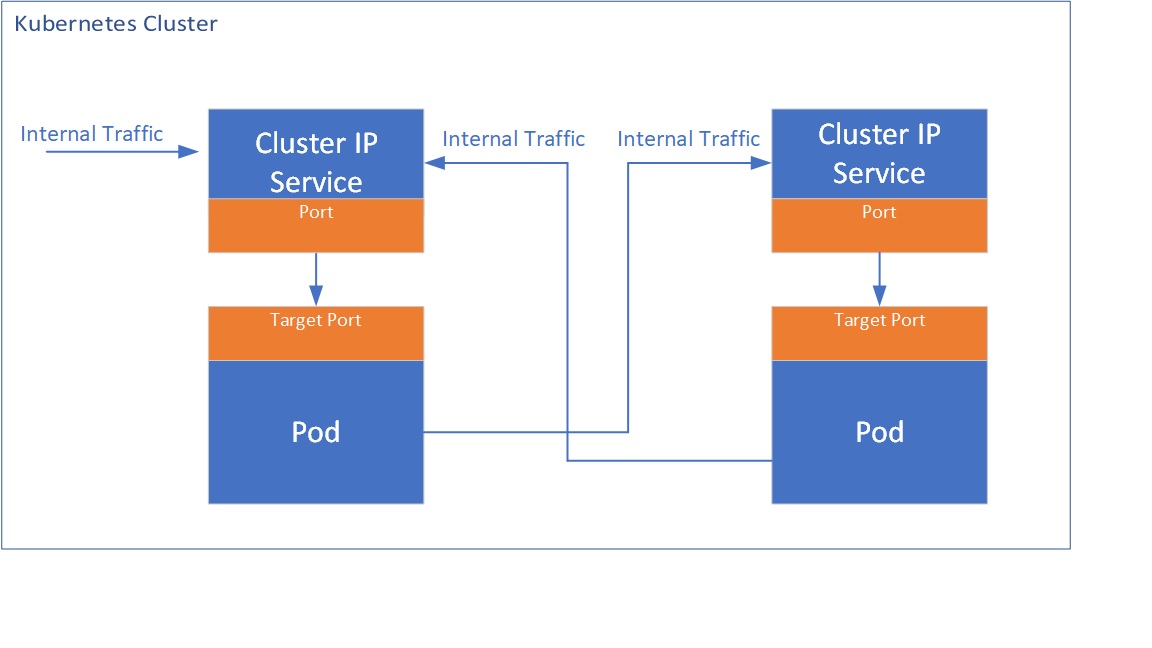
* The Kubernetes command-line tool, [kubectl](https://kubernetes.io/docs/reference/kubectl/kubectl/), allows you to run commands against Kubernetes clusters.
* You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.

1. **What are the different services within Kubernetes?**

**Different types of Kubernetes services are**

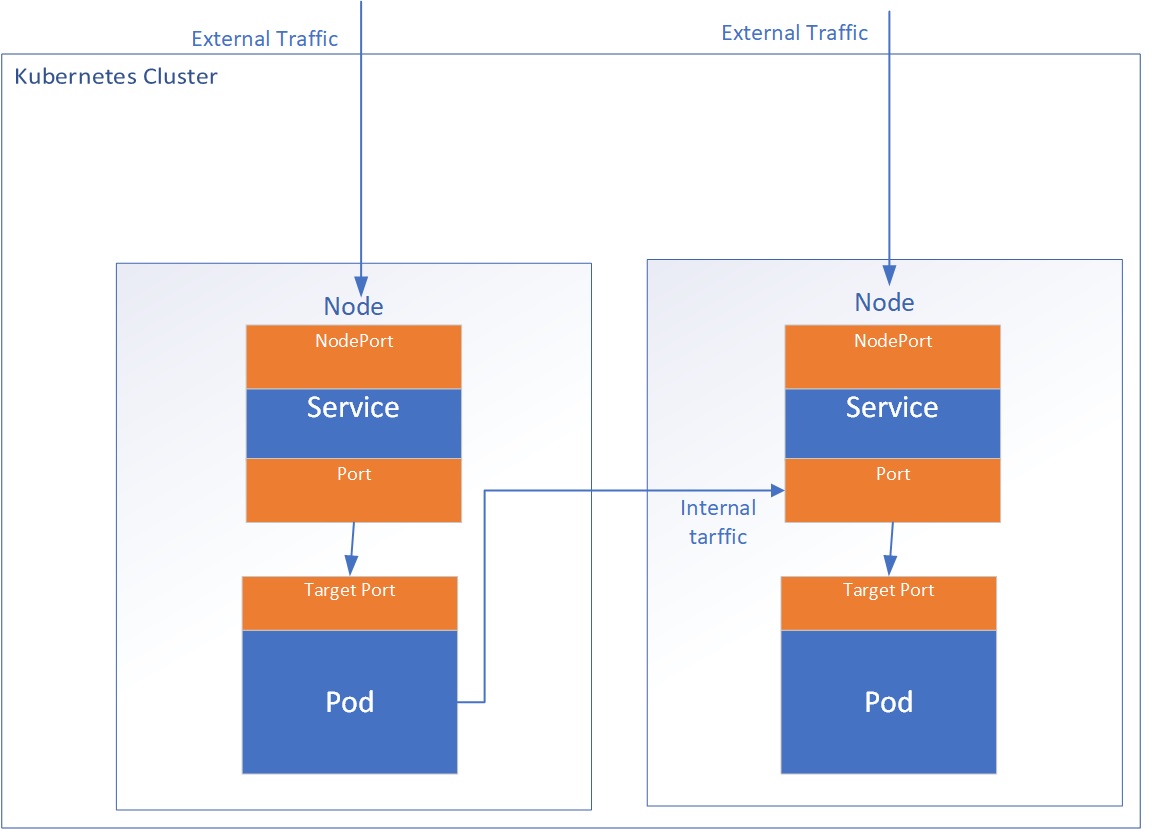
**Cluster IP:**

* In ClusterIP, the services are not available for external access of the cluster and used for internal communications between different Pods or microservices in the cluster.



**Node Port:**

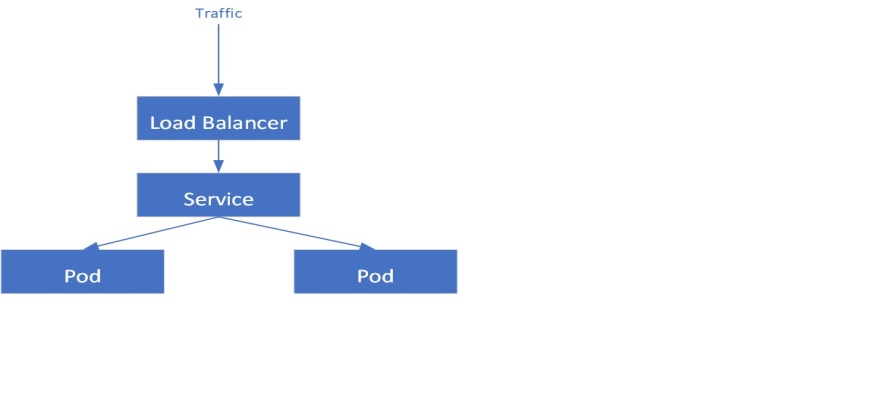
* This service exposes outside and allows the outside traffic to connect to Kubernetes Pods through the node port which is the port opened at Node end.
* The Pods can be accessed from external using <NodeIp>:<Nodeport>
* If there are multiple nodes, multiple IP addresses with the same port can be exposed.



* Each node in your cluster has an open port called a NodePort. Even if your app runs on a different node, Kubernetes straightforwardly routes traffic from the NodePort to the service.
* Every Kubernetes cluster accepts NodePort, but you have to modify your firewalls if you’re using a cloud service provider like Google Cloud.

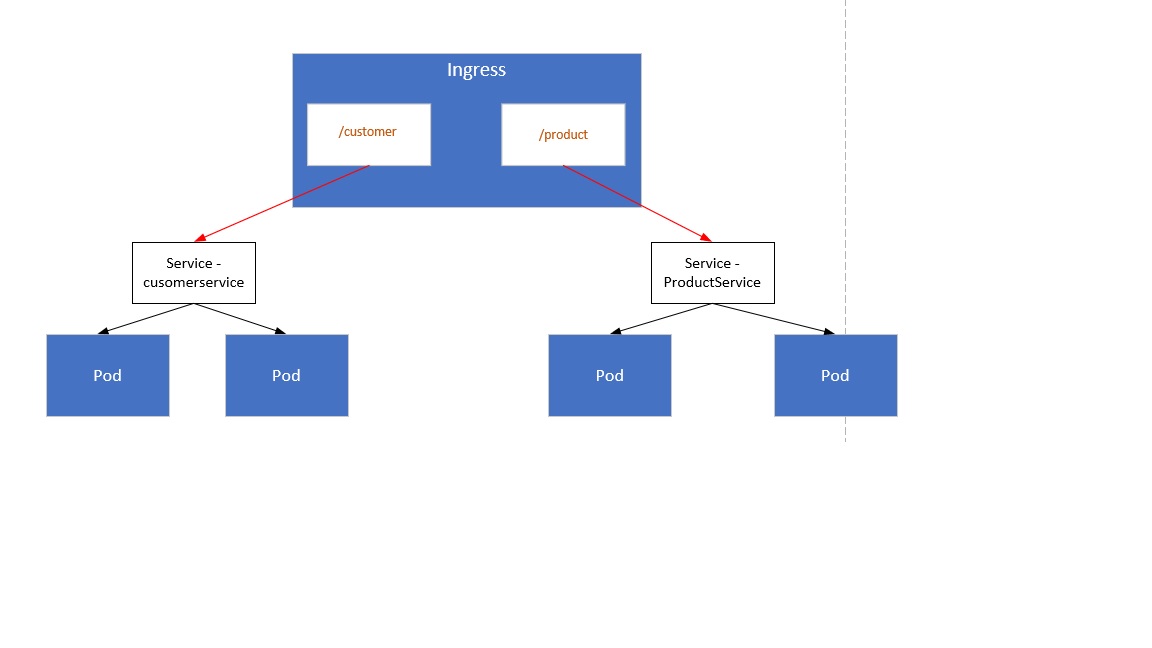
**Load Balancer:**

* This service will use or dynamically create an external load balancer like a cloud load balancer when running in the cloud.
* This uses Network load balancer (Layer 4 load balancer).  This generates additional costs for additional load balancer components.
* The advantage of this service is external load balancer features can be leveraged.



**Ingress:**

* This service allows the routing of HTTP(S) traffic according to defined rules like path-based routings.
* This can be associated with one or more service objects where these services are further associated with Pods. The ingress controller creates HTTP(S) load balancer Layer 7 load balancer which are configured automatically using the definition in the Ingress object.



1. **How to set a static IP for Kubernetes load balancer?**

* Kubernetes Master assigns a new IP address.
* We can set a static IP for Kubernetes load balancer by changing the DNS records whenever Kubernetes Master assigns a new IP address.

1. **What is ETCD?**

* etcd is a consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.
* If your Kubernetes cluster uses etcd as its backing store, make sure you have a [back up](https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/" \l "backing-up-an-etcd-cluster) plan for those data.
* Kubernetes uses etcd as a distributed key value store for all of its data, including metadata and configuration data, and allows nodes in Kubernetes clusters to read and write data.
* ETCD represents the state of a cluster at a specific moment in time and is a center for state management and cluster coordination of a Kubernetes cluster.

1. **Can you use many claims out of a persistent volume?**

* The mapping between persistentVolume and persistentVolumeClaim is always one to one.
* Even when you delete the claim, PersistentVolume still remains as we set persistentVolumeReclaimPolicy is set to Retain and It will not be reused by any other claims.

1. **How do you deploy a feature with zero downtime in Kubernetes?**

* In Kubernetes you can define the update strategy in deployments, you should put Rolling Update as a strategy to ensure no down time.
* **Refer the link:** https://kubernetes.io/blog/2018/04/30/zero-downtime-deployment-kubernetes-jenkins/

1. **What is the difference between replication controllers and replica sets?**

* Replica Set and Replication Controller do almost the same thing. Both of them ensure that a specified number of pod replicas are running at any given time.
* The difference comes with the usage of selectors to replicate pods. Replica Set use Set-Based selectors while replication controllers use Equity-Based selectors.

1. **What is Kube-proxy?**

* kube-proxy is a network proxy that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in your cluster, implementing part of the Kubernetes [Service](https://kubernetes.io/docs/concepts/services-networking/service/) concept.
* [kube-proxy](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/) maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.
* kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

1. **What is a Headless Service?**

* Headless service is similar to that of a ‘Normal’ service but does not have a Cluster IP.
* This service enables you to directly reach the pods without the need of accessing it through a proxy.

1. **Explain PVC?**

**PVC (Persistent Volume Claim):**

* It is storage requested by Kubernetes for pods. The user does not know the underlying provisioning.
* The claim should be created in the same namespace where the pod is created.
* **Refer the link:** https://kubernetes.io/docs/concepts/storage/persistent-volumes/

1. **Tell us about the different components of Kubernetes architecture**

* **Refer the Link:** https://kubernetes.io/docs/concepts/overview/components/

1. **If you have to pass sensitive information in you cluster how would you do it?**

* We can pass sensitive information in Kubernetes using secretes. Secrets can be created through yaml and text files.
* Majority of the organization use secret to pass sensitive information like username and password.
* **Refer the link:** https://kubernetes.io/docs/concepts/configuration/secret/

1. **What is sematext Docker agent?**

* Sematext Docker agent is a log collection agent with events and metrics.
* It runs as a small container in each Docker host.
* These agents gather metrics, events, and logs for all cluster nodes and contianers.
* **Refer Links:** https://github.com/sematext/sematext-agent-docker

1. **If you delete a pod(created as part of deployment) what happens to information inside of it?**

* Deployment will make sure that a new pod is created to maintain the number of replicas.
* Now talking about the information, it depends on the type of volume mount used, if you want information to be retained then you need to use persistent volume.

1. **Is there any pattern to pods being assigned to nodes? Can you make sure a Pod gets scheduled to a particular node?**

* Generally when you create a pod spawns automatically on any node(scheduled by Kubernetes internally to mange work-load and resources) but let’s say if you want to spawn a pod on a particular node that can also be done through taints.

1. **Let’s say a Kubernets job should finish in 20 seconds, however sometimes it takes 5 minutes, how I can make sure to stop the application if it exceeds more than 40 seconds.**

* When we create a job spec, we can give - -activeDeadlineSeconds flag to the command, this flag relates to the duration of the job, once the job reaches the threshold specified by the flag the job will be terminated.