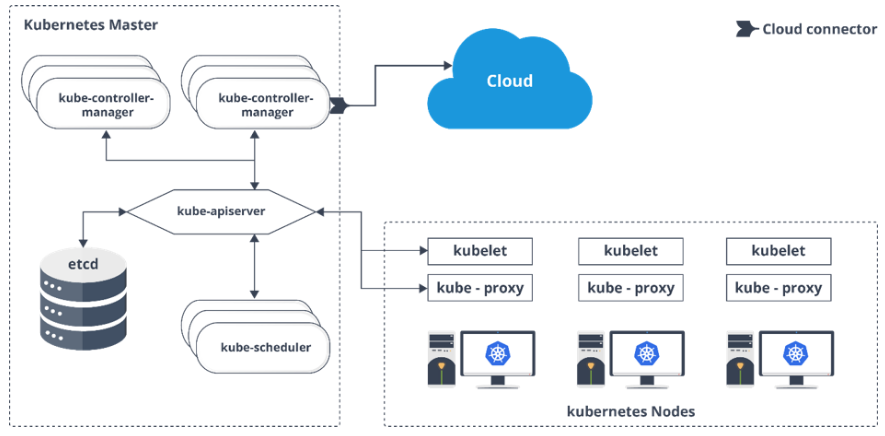
**1. KUBERNNETES**

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery

Arhitecture :

The Kubernetes Architecture has mainly 2 components – the master node and the worker node. As you can see in the below diagram, the master and the worker nodes have many inbuilt components within them. The master node has the kube-controller-manager, kube-apiserver, kube-scheduler, etcd. Whereas the worker node has kubelet and kube-proxy running on each node.



**2. KUBERNETES HEAPSTER**

Heapster is a cluster-wide aggregator of data provided by Kubelet running on each node. This container management tool is supported natively on Kubernetes cluster and runs as a pod, just like any other pod in the cluster. So, it basically discovers all nodes in the cluster and queries usage information from the Kubernetes nodes in the cluster, via on-machine Kubernetes agent

**3. KUBE-PROXY**

Kube-proxy can run on each and every node and can do simple TCP/UDP packet forwarding across backend network service. So basically, it is a network proxy which reflects the services as configured in Kubernetes API on each node. So, the Docker-linkable compatible environment variables provide the cluster IPs and ports which are opened by proxy

**4. KUBE CONTROLLER MANAGER**

In Kubernetes, a controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state. Examples of controllers that ship with Kubernetes today are the replication controller, endpoints controller, namespace controller, and service accounts controller.

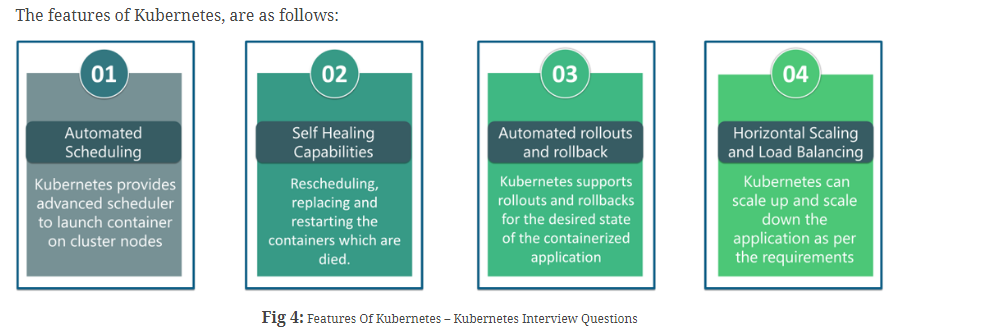
**5. KUBECTL**

kubectl is a command line interface for running commands against Kubernetes clusters. This overview covers kubectl syntax, describes the command operations, and provides common examples. For details about each command, including all the supported flags and subcommands, see the kubectl reference documentation.

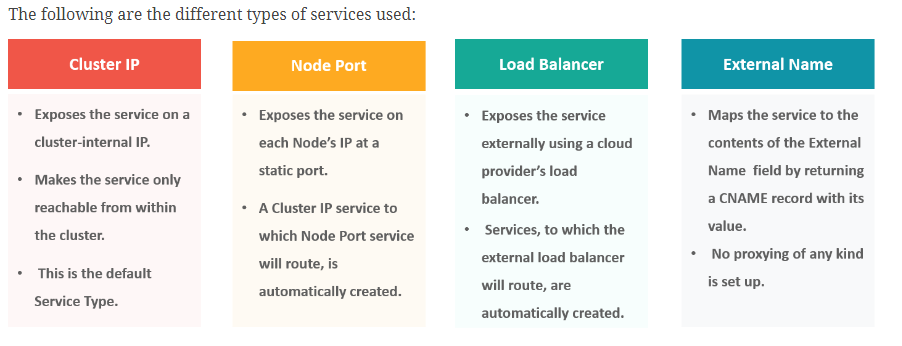
**6. NAMESPACE**

Namespaces are intended for use in environments with many users spread across multiple teams, or projects. Namespaces are a way to divide cluster resources between multiple uses (via resource quota). In future versions of Kubernetes, objects in the same namespace will have the same access control policies by default

**7. FEATURES OF KUBERNETES:**



**8. TYPES OF KUBERNETES SERVICES**

****

**9. KUBELET**

This is an agent service which runs on each node and enables the slave to communicate with the master. So, Kubelet works on the description of containers provided to it in the PodSpec and makes sure that the containers described in the PodSpec are healthy and running

**10. KUBECTL**

Kubectl is the platform using which you can pass commands to the cluster. So, it basically provides the CLI to run commands against the Kubernetes cluster with various ways to create and manage the Kubernetes component.

**11. HEADLESS SERVICE:**

Headless Service is similar to that of a ‘Normal’ services 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

**12. API SERVER AND KUBE SCHEDULER**

The kube – apiserver follows the scale-out architecture and, is the front-end of the master node control panel. This exposes all the APIs of the Kubernetes Master node components and is responsible for establishing communication between Kubernetes Node and the Kubernetes master components.

The kube-scheduler is responsible for distribution and management of workload on the worker nodes. So, it selects the most suitable node to run the unscheduled pod based on resource requirement and keeps a track of resource utilization. It makes sure that the workload is not scheduled on nodes which are already full.

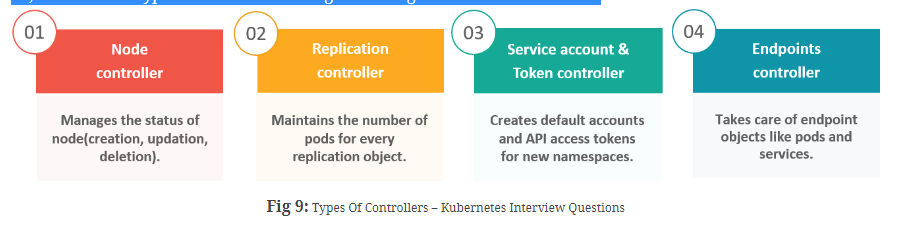
**13. ETCD**

Etcd is written in Go programming language and is a distributed key-value store used for coordinating between distributed work. So, Etcd stores the configuration data of the Kubernetes cluster, representing the state of the cluster at any given point in time.

**14. KUBERNETES CONTROLLER MANAGER**

Multiple controller processes run on the master node but are compiled together to run as a single process which is the Kubernetes Controller Manager. So, Controller Manager is a daemon that embeds controllers and does namespace creation and garbage collection. It owns the responsibility and communicates with the API server to manage the end-points.

So, the different types of controller manager running on the master node are

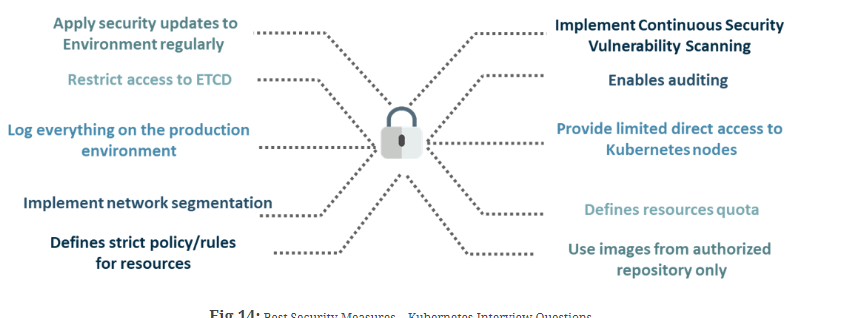
****

**15. DIFFERENCE BETWEEN REPLICA SET AND CONTROLLER**

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.

* **Equity-Based Selectors:**This type of selector allows filtering by label key and values. So, in layman terms, the equity-based selector will only look for the pods which will have the exact same phrase as that of the label.  
  **Example**: Suppose your label key says app=nginx, then, with this selector, you can only look for those pods with label app equal to nginx.
* **Selector-Based Selectors:**This type of selector allows filtering keys according to a set of values. So, in other words, the selector based selector will look for pods whose label has been mentioned in the set.  
  **Example:** Say your label key says app in (nginx, NPS, Apache). Then, with this selector, if your app is equal to any of nginx, NPS, or Apache, then the selector will take it as a true result.

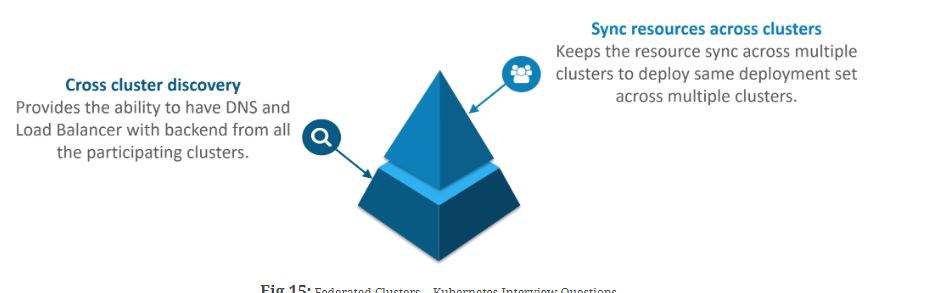
**16.SECURITY MEASURES IN KUBERNETES**

****

**18. FEDERATED CLUSTER:**

Multiple Kubernetes clusters can be managed as a single cluster with the help of federated clusters. So, you can create multiple Kubernetes clusters within a data center/cloud and use federation to control/manage them all at one place.

The federated clusters can achieve this by doing the following two things. Refer to the below diagram.

****

**19. INGRESS IN KUBERNETES**

Ingress network is a collection of rules that acts as an entry point to the Kubernetes cluster. This allows inbound connections, which can be configured to give services externally through reachable URLs, load balance traffic, or by offering name-based virtual hosting. So, Ingress is an API object that manages external access to the services in a cluster, usually by HTTP and is the most powerful way of exposing service

Few of INGRESS controller types

NGINX, HAPROXY, TRAEFIK & ISTIO

Ingress can provide load balancing, SSL termination and name-based virtual hosting

**20. HELM**

**Kuberneted package manager**

Helm helps you manage Kubernetes applications — Helm Charts helps you define, install, and upgrade even the most complex Kubernetes application.

Charts are easy to create, version, share, and publish — so start using Helm and stop the copy-and-paste.

The latest version of Helm is maintained by the CNCF - in collaboration with Microsoft, Google, Bitnami and the Helm contributor community.

**Kubeapps** is a web-based UI for deploying and managing applications in Kubernetes clusters