Q1. How is Kubernetes different from Docker Swarm?

Features	Kubernetes	Docker Swarm
Installation & Cluster Config	Setup is very complicated, but once installed cluster is robust.	Installation is very simple, but the cluster is not robust.
GUI	GUI is the Kubernetes Dashboard.	There is no GUI.
Scalability	Highly scalable and scales fast.	Highly scalable and scales 5x faster than Kubernetes.
Auto-scaling	Kubernetes can do auto-scaling.	Docker swarm cannot do auto-scaling.
Load Balancing	Manual intervention needed for load balancing traffic between different containers and pods.	Docker swarm does auto load balancing of traffic between containers in the cluster.
Rolling Updates & Rollbacks	Can deploy rolling updates and does automatic rollbacks.	Can deploy rolling updates, but not automatic rollback.
DATA Volumes	Can share storage volumes only with the other containers in the same pod.	Can share storage volumes with any other container.
Logging & Monitoring	In-built tools for logging and monitoring.	3rd party tools like ELK stack should be used for logging and monitoring.

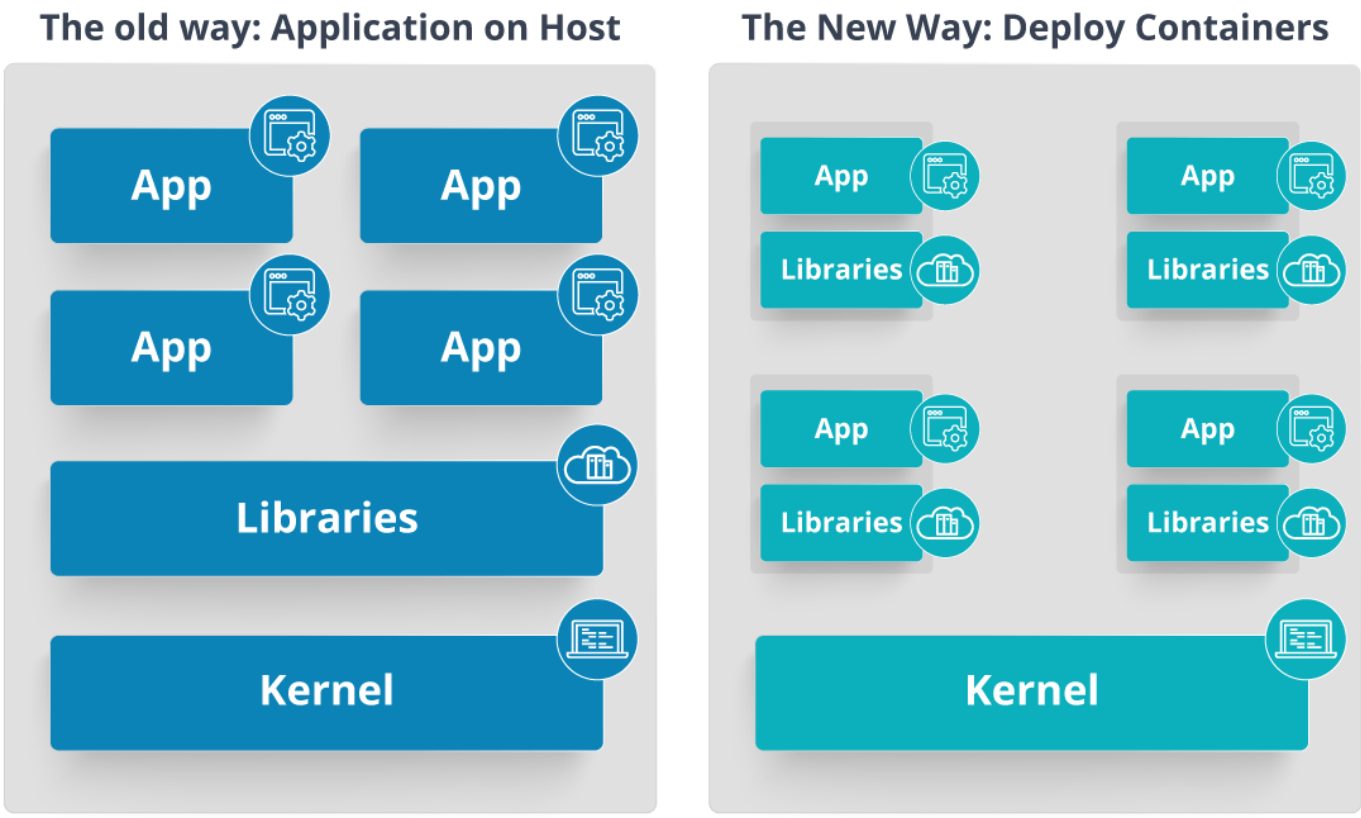
Q2. What is Kubernetes?

Kubernetes is an open-source container management tool which holds the responsibilities of container deployment, scaling & descaling of containers & load balancing. Being the Google's brainchild, it offers excellent community and works brilliantly with all the cloud providers. So, we can say that Kubernetes is not *a containerization platform, but it is a multi-container management solution.*

Q3. How is Kubernetes related to Docker?

It's a known fact that Docker provides the lifecycle management of containers and a Docker image builds the runtime containers. But, since these individual containers have to communicate, Kubernetes is used. So, Docker builds the containers and these containers communicate with each other via Kubernetes. So, containers running on multiple hosts can be manually linked and orchestrated using Kubernetes.

Q4. What is the difference between deploying applications on hosts and containers?



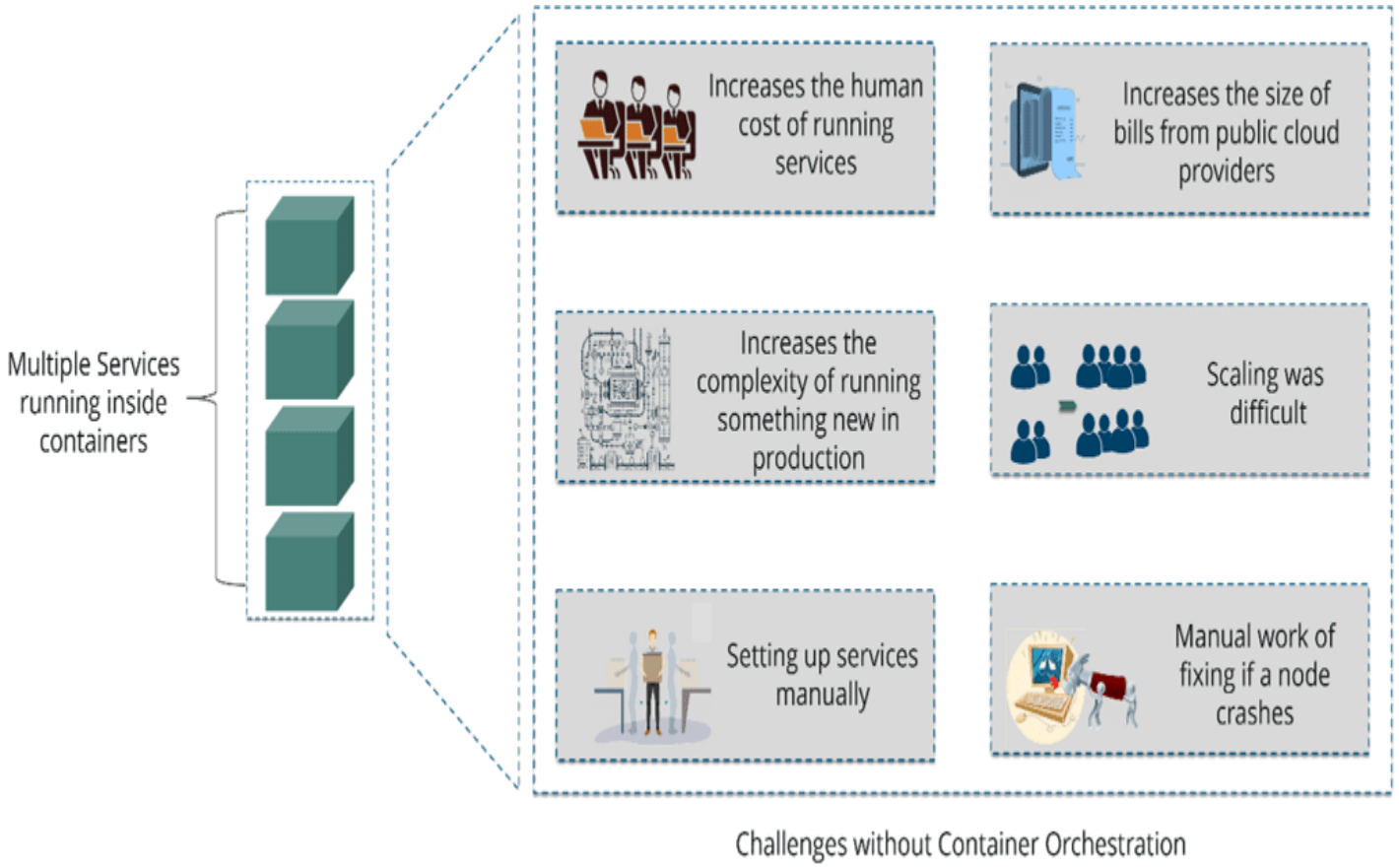
- Refer to the above diagram. The left side architecture represents deploying applications on hosts. So, this kind of architecture will have an operating system and then the operating system will have a kernel which will have various libraries installed on the operating system needed for the application. So, in this kind of framework you can have n number of applications and all the applications will share the libraries present in that operating system whereas while deploying applications in containers the architecture is a little different.
- This kind of architecture will have a kernel and that is the only thing that's going to be the only thing common between all the applications. So, if there's a particular application which needs Java then that particular application we'll get access to Java and if there's another application which needs Python then only that particular application will have access to Python.
- The individual blocks that you can see on the right side of the diagram are basically containerized and these are isolated from other applications. So, the applications have the necessary libraries and binaries isolated from the rest of the system, and cannot be encroached by any other application.

Q5. What is Container Orchestration?

Consider a scenario where you have 5-6 microservices for an application. Now, these microservices are put in individual containers, but won't be able to communicate without container orchestration. So, as orchestration means the amalgamation of all instruments playing together in harmony in music, similarly container orchestration means all the services in individual containers working together to fulfill the needs of a single server.

Q6. What is the need for Container Orchestration?

Consider you have 5-6 microservices for a single application performing various tasks, and all these microservices are put inside containers. Now, to make sure that these containers communicate with each other we need container orchestration.

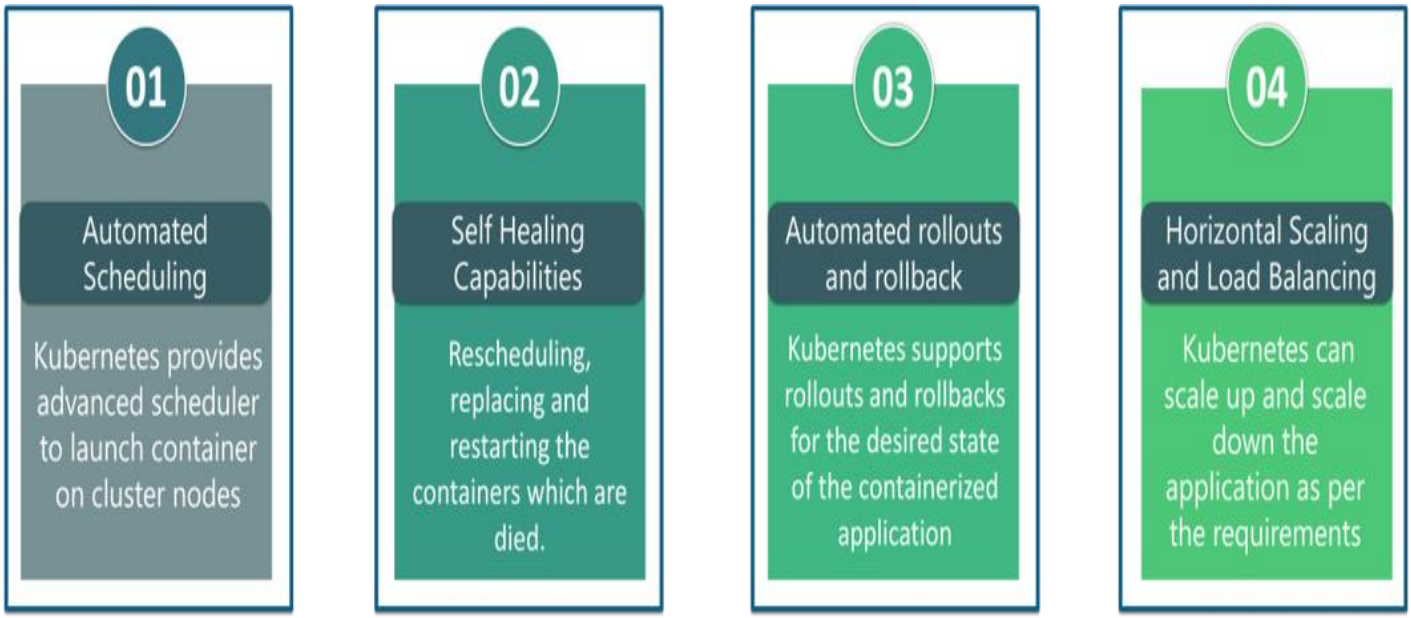


Challenges without Container Orchestration

As you can see in the above diagram, there were also many challenges that came into place without the use of container orchestration. So, to overcome these challenges the container orchestration came into place.

Q7. What are the features of Kubernetes?

The features of Kubernetes, are as follows:

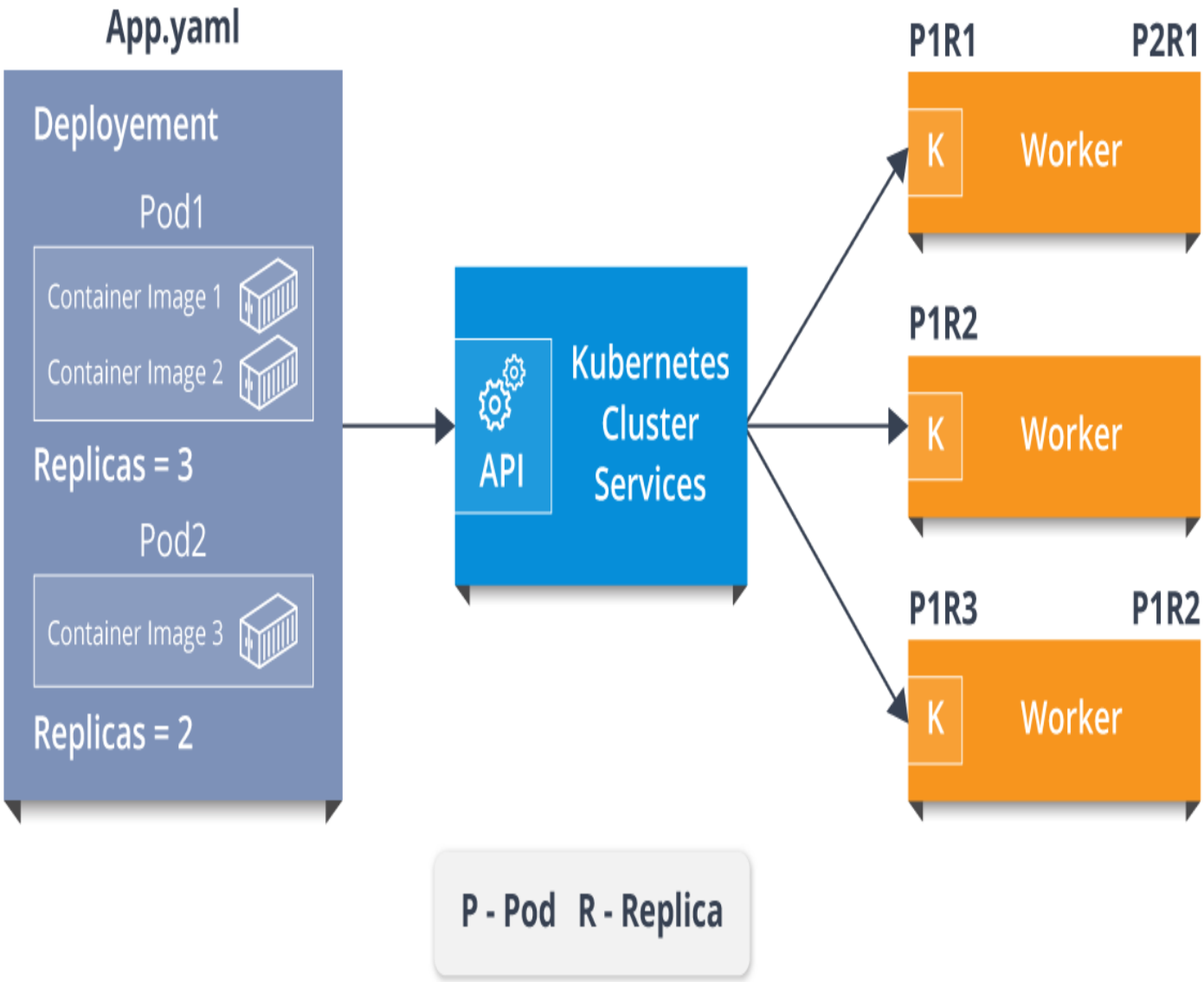


Q8. How does Kubernetes simplify containerized Deployment?

As a typical application would have a cluster of containers running across multiple hosts, all these containers would need to talk to each other. So, to do this you need something big that would load balance, scale & monitor the containers. Since Kubernetes is cloud-agnostic and can run on any public/private providers it must be your choice simplify containerized deployment.

Q9. What do you know about clusters in Kubernetes?

The fundamental behind Kubernetes is that we can enforce the desired state management, by which I mean that we can feed the cluster services of a specific configuration, and it will be up to the cluster services to go out and run that configuration in the infrastructure.



So, as you can see in the above diagram, the deployment file will have all the configurations required to be fed into the cluster services. Now, the deployment file will be fed to the API and then it will be up to the cluster services to figure out how to schedule these pods in the environment and make sure that the right number of pods are running.

So, the API which sits in front of services, the worker nodes & the Kubelet process that the nodes run, all together make up the Kubernetes Cluster.

Q10. What is Google Container Engine?

Google Container Engine (GKE) is an open source management platform for Docker containers and the clusters. This Kubernetes based engine supports only those clusters which run within the Google's public cloud services.

Q11. What is 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.

Q12. What is Minikube?

Minikube is a tool that makes it easy to run Kubernetes locally. This runs a single-node Kubernetes cluster inside a virtual machine.

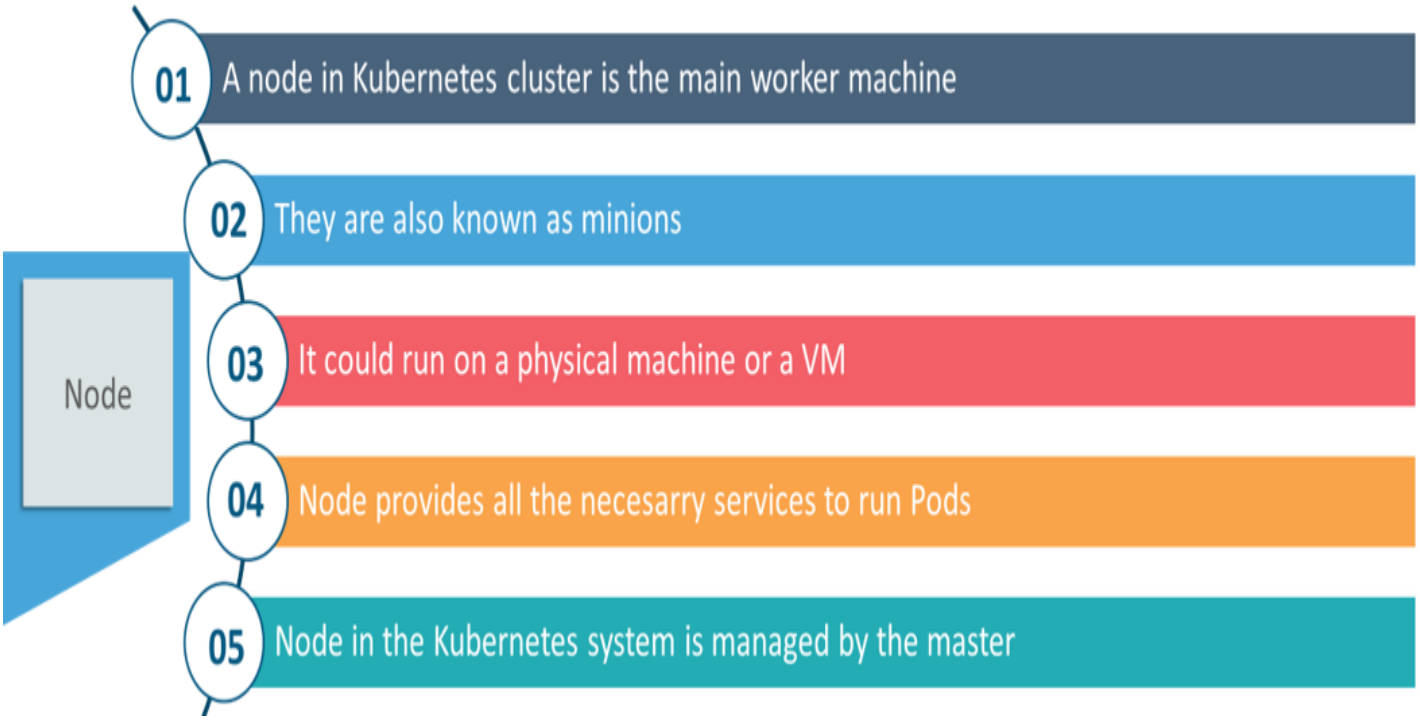
Q13. What is Kubectl?

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.

Q14. What is Kubelet?

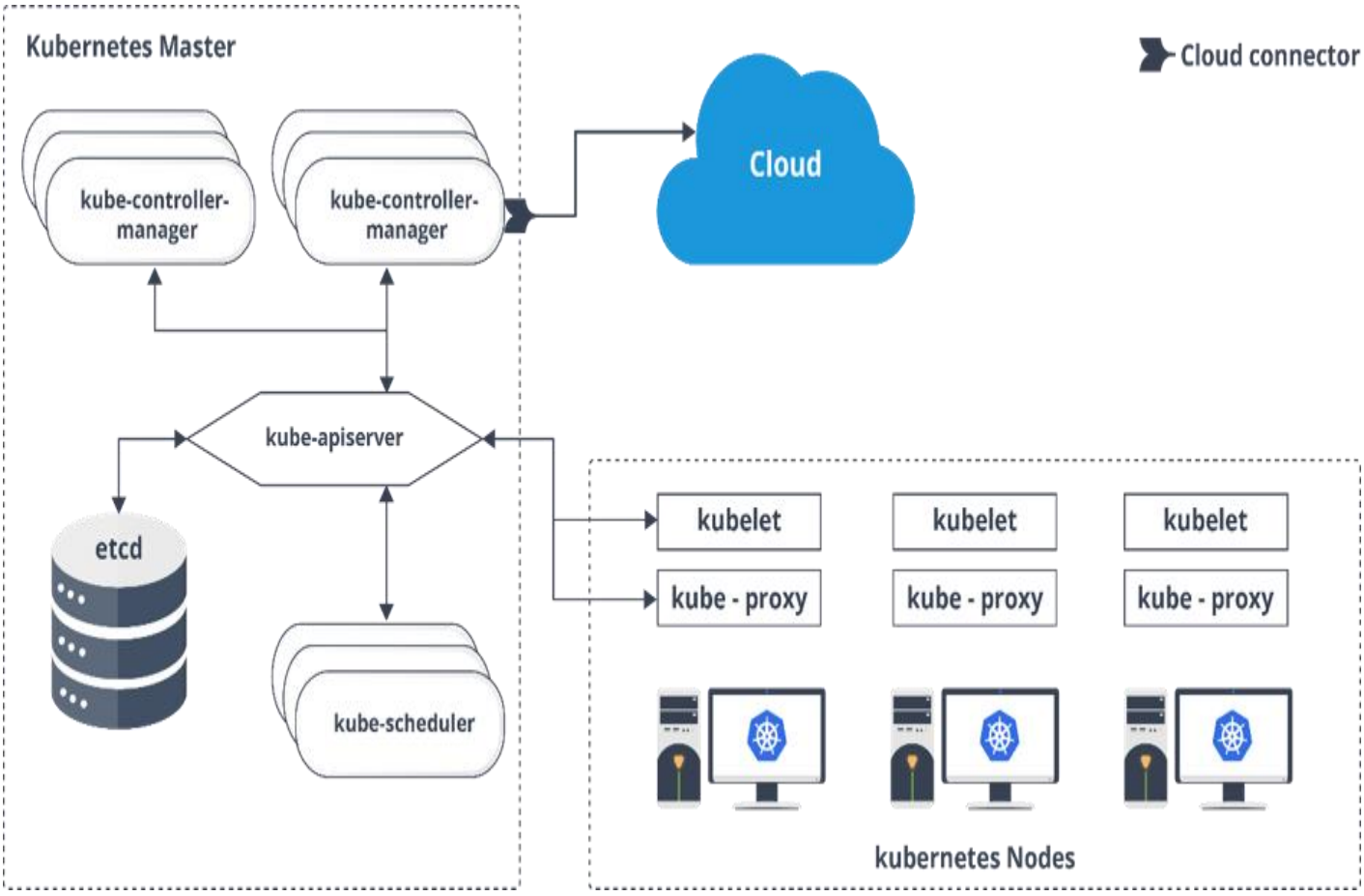
This is an agent service which runs on each node and enables the slave to communicate with the master. So, Kubelet works on the description of containers provided to it in the PodSpec and makes sure that the containers described in the PodSpec are healthy and running.

Q15. What do you understand by a node in Kubernetes?



Q1. What are the different components of Kubernetes Architecture?

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.

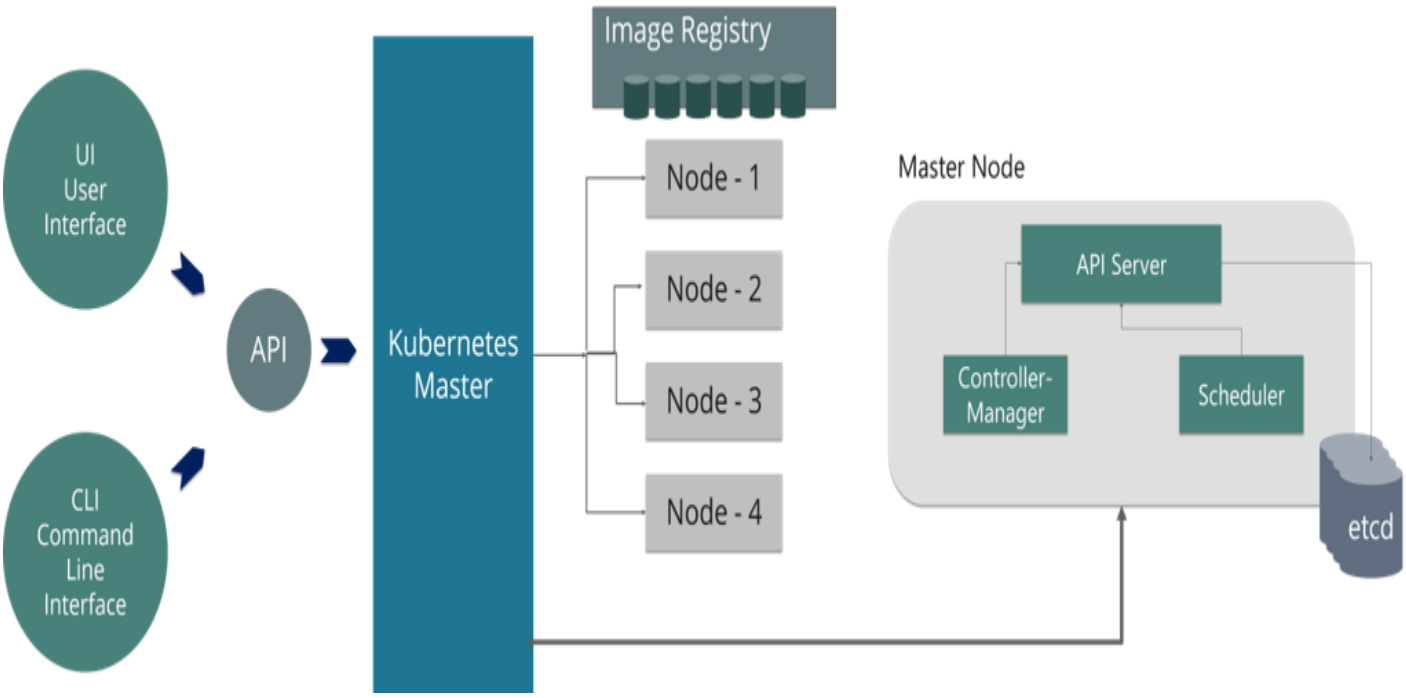


Q2. What do you understand by Kube-proxy?

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.

Q3. Can you brief on the working of the master node in Kubernetes?

Kubernetes master controls the nodes and inside the nodes the containers are present. Now, these individual containers are contained inside pods and inside each pod, you can have a various number of containers based upon the configuration and requirements. So, if the pods have to be deployed, then they can either be deployed using user interface or command line interface. Then, these pods are scheduled on the nodes and based on the resource requirements, the pods are allocated to these nodes. The kube-apiserver makes sure that there is communication established between the Kubernetes node and the master components.



Q4. What is the role of kube-apiserver and kube-scheduler?

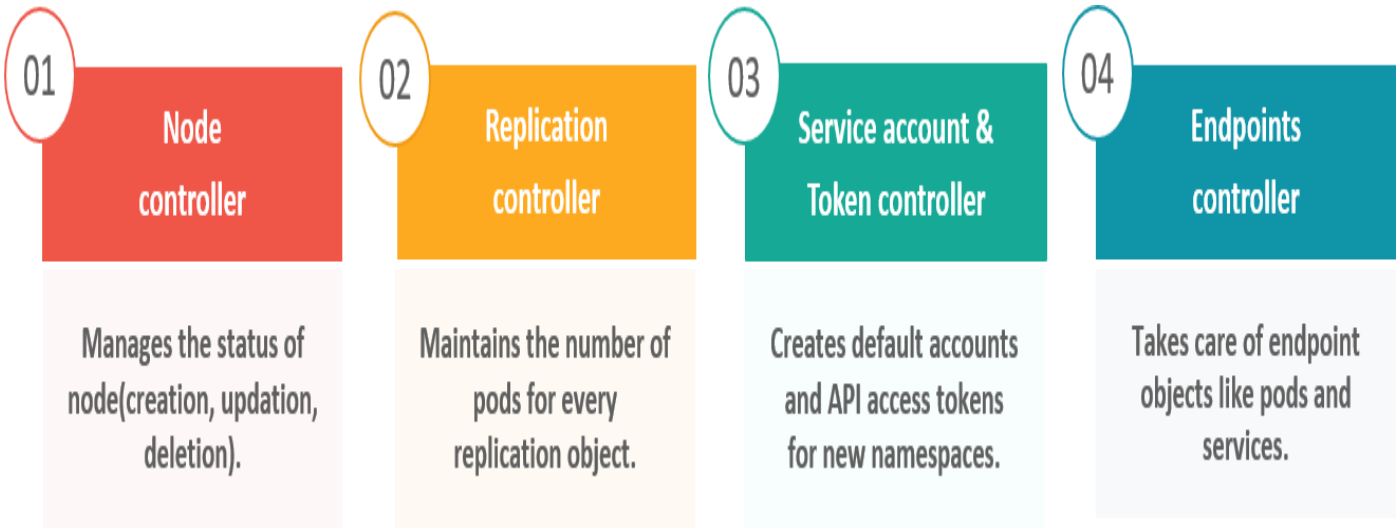
The kube – apiserver follows the scale-out architecture and, is the front-end of the master node control panel. This exposes all the APIs of the Kubernetes Master node components and is responsible for establishing communication between Kubernetes Node and the Kubernetes master components.

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.

Q5. Can you brief about the 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 :

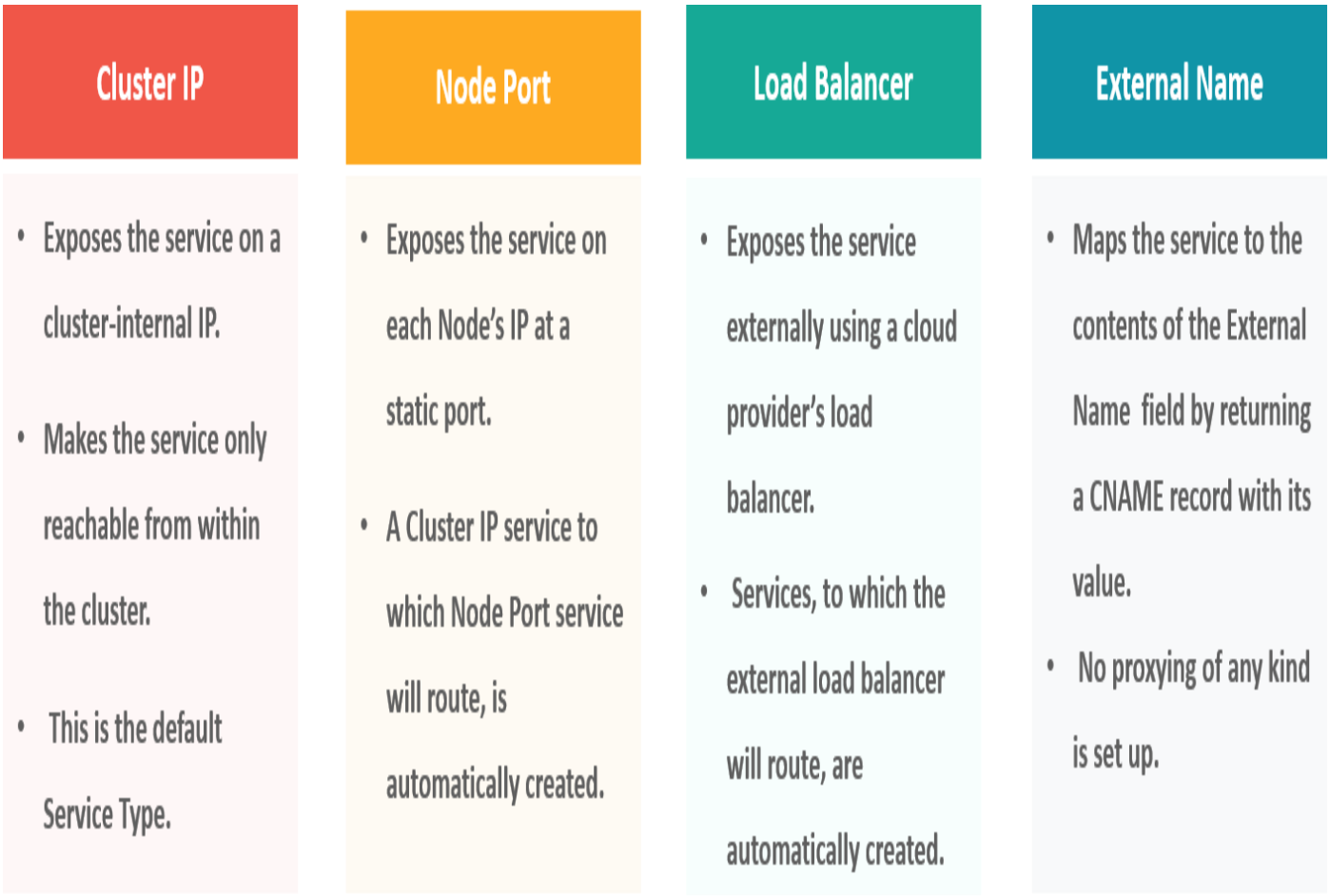


Q6. What is ETCD?

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.

Q7. What are the different types of services in Kubernetes?

The following are the different types of services used:



Q8. What do you understand by load balancer in Kubernetes?

A load balancer is one of the most common and standard ways of exposing service. There are two types of load balancer used based on the working environment i.e. either the Internal Load Balancer or the External Load Balancer. The Internal Load Balancer automatically balances load and allocates the pods with the required configuration whereas the External Load Balancer directs the traffic from the external load to the backend pods.

Q9. What is Ingress network, and how does it work?

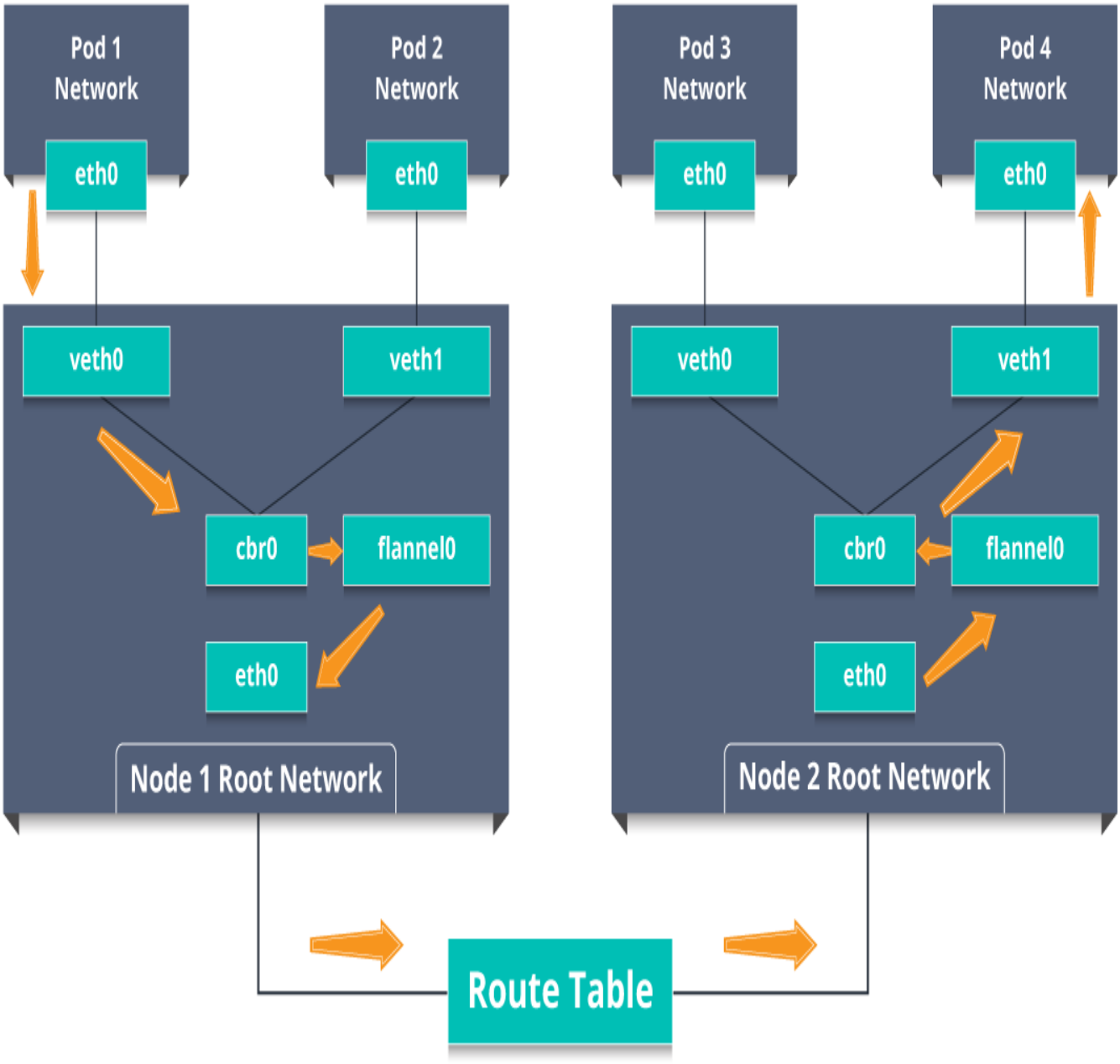
Ingress network is a collection of rules that acts as an entry point to the Kubernetes cluster. This allows inbound connections, which can be configured to give services externally through reachable URLs, load balance traffic, or by offering name-based virtual hosting. So, Ingress is an

API object that manages external access to the services in a cluster, usually by HTTP and is the most powerful way of exposing service.

Now, let me explain to you the working of Ingress network with an example.

There are 2 nodes having the pod and root network namespaces with a Linux bridge. In addition to this, there is also a new virtual ethernet device called flannel0(network plugin) added to the root network.

Now, suppose we want the packet to flow from pod1 to pod 4. Refer to the below diagram.

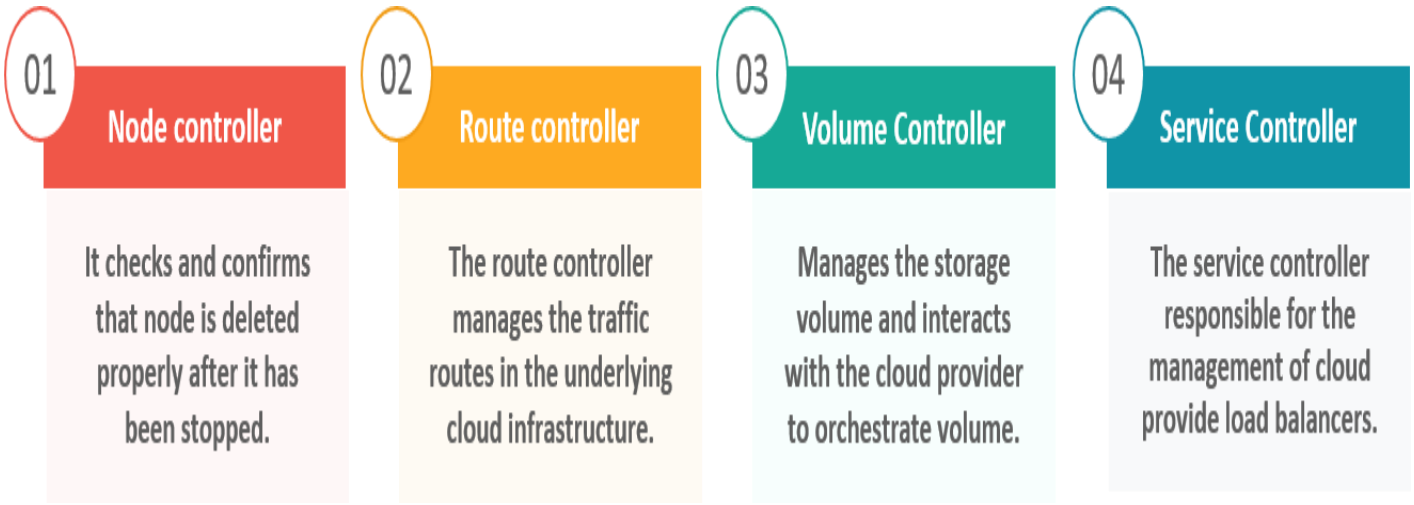


- So, the packet leaves pod1's network at eth0 and enters the root network at veth0.
- Then it is passed on to cbr0, which makes the ARP request to find the destination and it is found out that nobody on this node has the destination IP address.
- So, the bridge sends the packet to flannel0 as the node's route table is configured with flannel0.
- Now, the flannel daemon talks to the API server of Kubernetes to know all the pod IPs and their respective nodes to create mappings for pods IPs to node IPs.
- The network plugin wraps this packet in a UDP packet with extra headers changing the source and destination IP's to their respective nodes and sends this packet out via eth0.
- Now, since the route table already knows how to route traffic between nodes, it sends the packet to the destination node2.
- The packet arrives at eth0 of node2 and goes back to flannel0 to de-capsulate and emits it back in the root network namespace.
- Again, the packet is forwarded to the Linux bridge to make an ARP request to find out the IP that belongs to veth1.
- The packet finally crosses the root network and reaches the destination Pod4.

Q10. What do you understand by Cloud controller manager?

The Cloud Controller Manager is responsible for persistent storage, network routing, abstracting the cloud-specific code from the core Kubernetes specific code, and managing the communication with the underlying cloud services. It might be split out into several different containers depending on which cloud platform you are running on and then it enables the cloud vendors and Kubernetes code to be developed without any inter-dependency. So, the cloud vendor develops their code and connects with the Kubernetes cloud-controller-manager while running the Kubernetes.

The various types of cloud controller manager are as follows:



Q11. What is Container resource monitoring?

As for users, it is really important to understand the performance of the application and resource utilization at all the different abstraction layer, Kubernetes factored the management of the cluster by creating abstraction at different levels like container, pods, services and whole cluster. Now, each level can be monitored and this is nothing but Container resource monitoring.

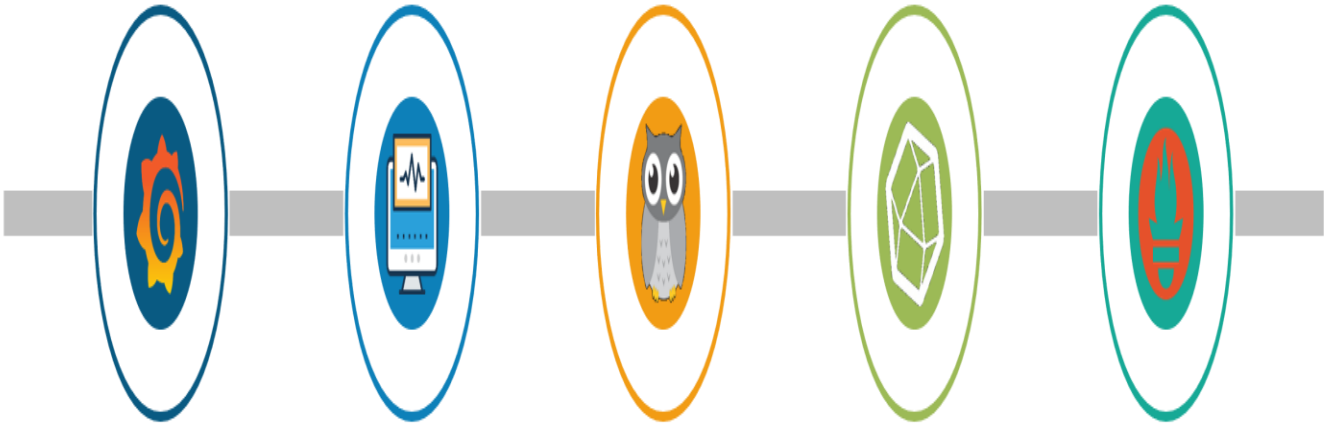
The various container resource monitoring tools are as follows:

Heapster

Gathers data and events from the containers and pods within the cluster.

InfluxDB

Used along with Heapster for visualizing data within the Kubernetes environment.



Grafana

A time-series database to store the data captured by all Heapster pods.

CAdvisor

A built-in tool in a kubelet that automatically discovers all the active containers and monitors them.

Prometheus

A project of CNCF which provides a powerful querying, alerting and visualization capabilities.

Q12. What is the difference between a replica set and replication 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.

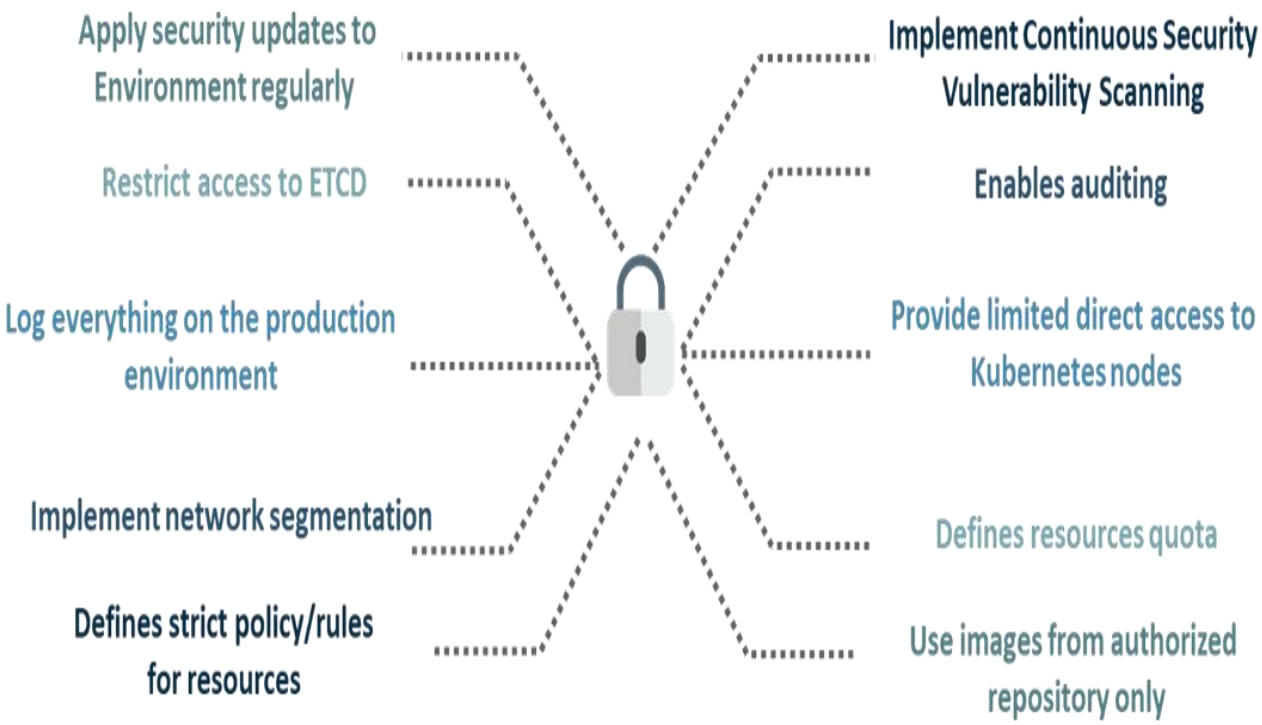
- **Selector-Based Selectors:** This type of selector allows filtering keys according to a set of values. So, in other words, the selector based selector will look for pods whose label has been mentioned in the set.
  Example: Say your label key says app in (nginx, NPS, Apache). Then, with this selector, if your app is equal to any of nginx, NPS, or Apache, then the selector will take it as a true result.

Q13. What is a 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.

Q14. What are the best security measures that you can take while using Kubernetes?

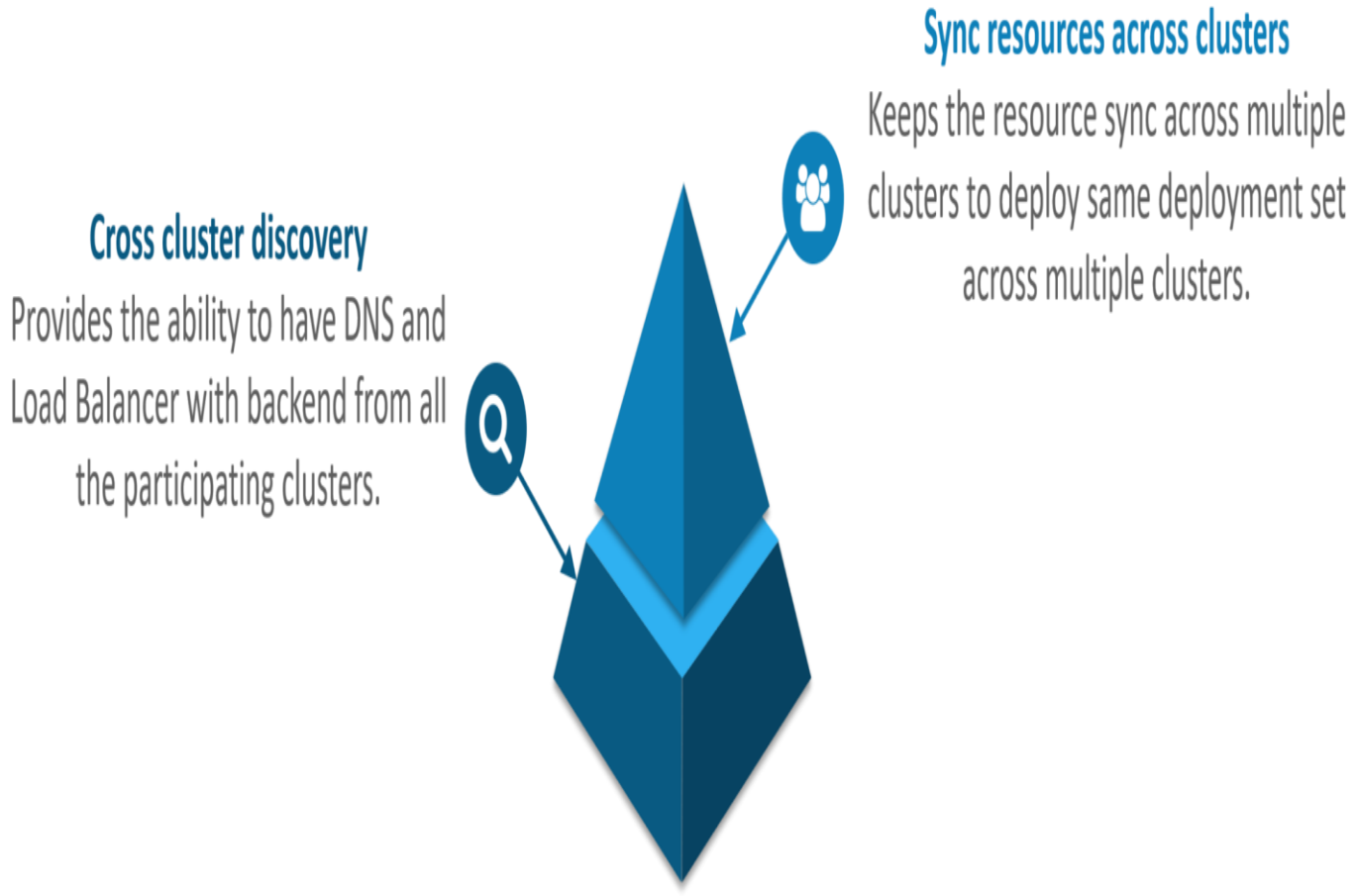
The following are the best security measures that you can follow while using Kubernetes:



Q15. What are federated clusters?

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.



cenario 1: Suppose a company built on monolithic architecture handles numerous products. Now, as the company expands in today's scaling industry, their monolithic architecture started causing problems.

How do you think the company shifted from monolithic to microservices and deploy their services containers?

Solution:

As the company's goal is to shift from their monolithic application to microservices, they can end up building piece by piece, in parallel and just switch configurations in the background. Then they can put each of these built-in microservices on the Kubernetes platform. So, they can start by migrating their services once or twice and monitor them to make sure everything is running stable. Once they feel everything is going good, then they can migrate the rest of the application into their Kubernetes cluster.

Scenario 2: Consider a multinational company with a very much distributed system, with a large number of data centers, virtual machines, and many employees working on various tasks.

How do you think can such a company manage all the tasks in a consistent way with Kubernetes?

Solution:

As all of us know that I.T. departments launch thousands of containers, with tasks running across a numerous number of nodes across the world in a distributed system.

In such a situation the company can use something that offers them agility, scale-out capability, and DevOps practice to the cloud-based applications.

So, the company can, therefore, use Kubernetes to customize their scheduling architecture and support multiple container formats. This makes it possible for the affinity between container tasks that gives greater efficiency with an extensive support for various container networking solutions and container storage.

Scenario 3: Consider a situation, where a company wants to increase its efficiency and the speed of its technical operations by maintaining minimal costs.

How do you think the company will try to achieve this?

Solution:

The company can implement the DevOps methodology, by building a CI/CD pipeline, but one problem that may occur here is the configurations may take time to go up and running. So, after implementing the CI/CD pipeline the company's next step should be to work in the cloud environment. Once they start working on the cloud environment, they can schedule containers on a cluster and can orchestrate with the help of Kubernetes. This kind of approach will help the company reduce their deployment time, and also get faster across various environments.

Scenario 4: Suppose a company wants to revise it's deployment methods and wants to build a platform which is much more scalable and responsive.

How do you think this company can achieve this to satisfy their customers?

Solution:

In order to give millions of clients the digital experience they would expect, the company needs a platform that is scalable, and responsive, so that they could quickly get data to the client website. Now, to do this the company should move from their private data centers (if they are using any) to any cloud environment such as AWS. Not only this, but they should also implement the microservice architecture so that they can start using Docker containers. Once they have the base framework ready, then they can start using the best orchestration platform

available i.e. Kubernetes. This would enable the teams to be autonomous in building applications and delivering them very quickly.

Scenario 5: Consider a multinational company with a very much distributed system, looking forward to solving the monolithic code base problem.

How do you think the company can solve their problem?

Solution

Well, to solve the problem, they can shift their monolithic code base to a microservice design and then each and every microservices can be considered as a container. So, all these containers can be deployed and orchestrated with the help of Kubernetes.

Scenario 6: All of us know that the shift from monolithic to microservices solves the problem from the development side, but increases the problem at the deployment side.

How can the company solve the problem on the deployment side?

Solution

The team can experiment with container orchestration platforms, such as Kubernetes and run it in data centers. So, with this, the company can generate a templated application, deploy it within five minutes, and have actual instances containerized in the staging environment at that point. This kind of Kubernetes project will have dozens of microservices running in parallel to improve the production rate as even if a node goes down, then it can be rescheduled immediately without performance impact.

Scenario 7: Suppose a company wants to optimize the distribution of its workloads, by adopting new technologies.

How can the company achieve this distribution of resources efficiently?

Solution

The solution to this problem is none other than Kubernetes. Kubernetes makes sure that the resources are optimized efficiently, and only those resources are used which are needed by that particular application. So, with the usage of the best container orchestration tool, the company can achieve the distribution of resources efficiently.

Scenario 8: Consider a carpooling company wants to increase their number of servers by simultaneously scaling their platform.

How do you think will the company deal with the servers and their installation?

Solution

The company can adopt the concept of containerization. Once they deploy all their application into containers, they can use Kubernetes for orchestration and use container monitoring tools like Prometheus to monitor the actions in containers. So, with such usage of containers, giving them better capacity planning in the data center because they will now have fewer constraints due to this abstraction between the services and the hardware they run on.

Scenario 9: Consider a scenario where a company wants to provide all the required hand-outs to its customers having various environments.

How do you think they can achieve this critical target in a dynamic manner?

Solution

The company can use Docker environments, to put together a cross-sectional team to build a web application using Kubernetes. This kind of framework will help the company achieve the goal of getting the required things into production within the shortest time frame. So, with such a machine running, the company can give the hands-outs to all the customers having various environments.

Scenario 10: Suppose a company wants to run various workloads on different cloud infrastructure from bare metal to a public cloud.

How will the company achieve this in the presence of different interfaces?

Solution

The company can decompose its infrastructure into microservices and then adopt Kubernetes. This will let the company run various workloads on different cloud infrastructures.

Q1. What are minions in Kubernetes cluster?

a. They are components of the master node.

b. They are the work-horse / worker node of the cluster.[Ans]

c. They are monitoring engine used widely in kubernetes.

d. They are docker container service.

Q2. Kubernetes cluster data is stored in which of the following?

a. Kube-apiserver

b. Kubelet

c. Etcd[Ans]

d. None of the above

Q3. Which of them is a Kubernetes Controller?

a. ReplicaSet

b. Deployment

c. Rolling Updates

d. Both ReplicaSet and Deployment[Ans]

Q4. Which of the following are core Kubernetes objects?

a. Pods

b. Services

c. Volumes

d. All of the above[Ans]

Q5. The Kubernetes Network proxy runs on which node?

a. Master Node

b. Worker Node

c. All the nodes[Ans]

d. None of the above

Q6. What are the responsibilities of a node controller?

a. To assign a CIDR block to the nodes

b. To maintain the list of nodes

c. To monitor the health of the nodes

d. All of the above[Ans]

Q7. What are the responsibilities of Replication Controller?

a. Update or delete multiple pods with a single command

b. Helps to achieve the desired state

c. Creates a new pod, if the existing pod crashes

d. All of the above[Ans]

Q8. How to define a service without a selector?

a. Specify the external name[Ans]

b. Specify an endpoint with IP Address and port

c. Just by specifying the IP address

d. Specifying the label and api-version

Q9. What did the 1.8 version of Kubernetes introduce?

a. Taints and Tolerations[Ans]

b. Cluster level Logging

c. Secrets

d. Federated Clusters

Q10. The handler invoked by Kubelet to check if a container's IP address is open or not is?

a. HTTPGetAction

b. ExecAction

c. TCPSocketAction[Ans]

d. None of the above