**s1. Define what Kubernetes is. What is its purpose?**

It refers to a system that does automating and scaling as well as the management of containerized applications. It handles the grouping of containers, which provides application for the logical units for management as well as discovery.

**2. Could you please provide a definition of a Docker?**

Docker refers to an open-source platform, which handles software development. The main advantage with it is it packages the settings into particular apparatus and then allows their portability within the systems running a Linux operating system.

**3. Give a description of Orchestration when it comes to software?**

The service orchestration alludes to integration of multiple services for them to allow the automation of the processes or synchronizing the information on a timely basis. As such, the point-to-point integration may be used as one such path for the least amount of resistance.

**4. How would you say that Kubernetes is related to Docker?**

Docker is responsible for the management of the lifecycle of containers and these are manually linked and orchestrated with Kubernetes.

**5. What are the scenarios in which a Java Developer is going to use Docker?**

Following scenarios are those a java developer can use docker for:

• Running UAT’s with the use of Docker.   
• Sharing development workspace, with pre-configured development environment.  
• Continuous integration is a popular use case for Docker. The teams looking build and deploy their applications may use Docker combined with ecosystem tools such as Jenkins to drive applications from development, testing, staging and into production without the need to change any code.

**6. What are Daemon sets?**

The Daemon set is a set of pods that is run only once on a host. It can be used for host layer attributes like a network, host monitoring or the storage plugin or things, which you would not want to run more than once on a host.

**7. What is the “Master”?**

Master refers to a central point of control, which gives a unified view of a cluster. There is a single master node, which controls different minions. Master servers then work together to accept user requests and determine the best means of scheduling the workload containers, authenticate clients and nodes as well as adjust on the cluster wide networking and managing the scaling and health checking of responsibilities.

**8. Could you please describe Kubernetes minions.**

The minion is an element that transforms itself into a worker element, which is a node. The node could refer to a physical machine or VM, which is dependent on the particular cluster. Each node has the services, which are required to run the pods and they are monitored through the main components. The services here may include that of the container runtime such as Kubelet.

**9. What are labels and annotations when it comes to Kubernetes?**

A label in Kubernetes is a meaningful type of tag word, which is attached to the Kubernetes objects in order to make them as part of a group. The Labels may be used for working on different instances for the purposes of management or even routing purposes. For one, the controller-based objects may use the labels to mark the pods they would operate on. The microservices use labels to understand the structure of the backend pods they route the requests toward. The labels are some of the key value pairs. Each unit may have more than one label but each unit may only have one entry for each of the keys. The key is most commonly utilized as an identifier or unique ID. However, at the same time may classify the objects using other criteria according to public access, application versions and the developmental stages.

The annotations attach arbitrary key value information to the Kubernetes object. The levels, however, ought to be utilized for meaningful information in order to match a pod with selection criteria, so the annotations have less structured data. The annotations are a means for adding more metadata to the object, which is not helpful for the selection purposes.

**10. What are the node server components for Kubernetes?**

In Kubernetes the server do their work through running containers, which are known as nodes. The execution of tasks and reporting the status to the master would be the main objective of the Node server.

The main process of the Kubernetes node, which does some of the significant container operations:

• The Kubelet is the node-daemon, which communicates with Kubernetes master for all the machines, which are a part of a Kubernetes cluster.  
• It regularly accesses the controller in order to check and report on the status of the cluster.  
• It merges the available CPU, memory and disk for a node into the large Kubernetes cluster. It also communicates the state of the containers back up to the API server for control loops in order to observe the current state of the containers.

**11. Can you please tell me what the difference is between deploying applications on dedicated hosts versus on Kubernetes containers?**

This kind of architecture is generally going to have an operating system associated with it. And that OS will have a kernel that will have various libraries installed on it for the application to work. In this framework, it is possible to have N number (meaning any number) of applications and all of them are going to share the available libraries. Each are present and accessed within the operating system, though, while deploying the applications in containers, the architecture can sometimes be a bit different. This architecture approach will have a kernel and that may be the only thing, which will be common between all of the applications. If there is a particular one, which requires Java, then that one will get access to it. The individual blocks, which can be seen on the right side of the diagram, are containerized and these are isolated from the other applications. That means the applications would have the required libraries and binaries, which are isolated from the system and these, cannot be encroached by other applications.

**12. Describe the meaning of a cluster in Kubernetes?**

The master and the node machinery refer to the ones run by the Kubernetes cluster orchestration framework. A container cluster would be the foundation of the Container Engine. The objects that represent the applications, which are containerized, are then run atop the cluster.

**13. What is a Swarm in Docker?**

The docker Swarm is a clustering and scheduling tool for the Docker containers. When it comes to Swarm, the IT administrators and developers would establish and manage a cluster of Docker nodes as part of the single virtual system.

**14. What is OpenShift?**

OpenShift online is Red Hat’s public cloud application development and hosting platform which provides automation for management, provision and the scaling of application so it is possible to focus on writing the code for the business or big idea.

**15. What does the nodes status contain?**

Some of the following things would be the main components of the node status.

• Address  
• Condition  
• Capacity  
• Info

**16. What are Pods in Kubernetes?**

A Kubernetes pod is a particular group of containers, which are deployed, in the same host. The Pods have the capacity to operate on a level, which is higher as compared to the individual containers. That is because the pods have the group of containers, which work together to produce an artifact or to process a particular set of work.

**17. Give a description of what a Namespace is when it comes to Kubernetes?**

The Namespace can be used in different environments with different users, who operate across a number of projects or even teams. They refer to the process of dividing the cluster resources between the different utilizations. For the future iterations, the objects within a similar Namespace would have a similar access control policy.

**18. Provide a definition of a node within Kubernetes?**

The node is the worker element within the framework and it was previously known as a minion. It could be a virtual or physical type of machine but this is dependent on the cluster. The nodes have the services needed for running the pods and they are managed via the main components. The services of each of the node may include such things as the Kubelet, Docker and Kube Proxy.

**19. What are the reasons for the utilization of Docker?**

It provides a similar capability without considerations of the overhead, which would be there with a virtual machine. Docker makes it possible to place the environment and configuration into code before its deployment. A similar Docker configuration could be used also for the different types of environments. That would then decouple the needs of the infrastructure from that of the environment.

**20. What is a Docker in Cloud?**

The node alludes to an individual Linux hosting platform utilized for the deployment and running of the applications. The cloud does not provide hosting services so the applications, containers and services may run on the particular hosts. The hosts do hail from different origins like the virtual machines, physical servers or the cloud providers.

**21. Can you please discuss the cluster of containers in Kubernetes.**

A container cluster concerns a set of machine elements that are nodes. The clusters initiate particular routes so the containers running on the nodes are able to communicate with each other. The server of the Kubernetes API does not function on these nodes. The container engine is the one which provides hosting for the API server.

**22. Can you please tell me what a Container Orchestration is?**

If you consider an engineering scenario where there is a need for 5 to 6 microservices for the application for a moment. These micro-services are then placed within individual containers but they are not going to be able to communicate without the container orchestration. Therefore as orchestration means the combination of all of the instruments playing together in harmony in music, container orchestration would mean all of the services within the individual containers working together in order to fulfill the requirements of a single particular server.

**23. What is the significance of Container Orchestration?**

Consider there are 5 to 6 micro-services for a single application performing different tasks and all of them are living within containers. In order to make sure these containers communicate with one other properly, there is a need for what is called container orchestration, which is built right into Kubernetes.

**24. What are some of the different attributes of Kubernetes?**

• Automated scheduling: Kubernetes allows advanced scheduler to launch container on the particular cluster nodes.  
• Automated rollouts and rollback: Kubernetes supports the different rollouts and rollbacks for the desired state of the containerized application.  
• Self-healing features: rescheduling, replacing and restarting the containers, which are dead.  
• Horizontal scaling and load balancing: Kubernetes may scale up and down the application according to the requirements.

**25. What are the means by which Kubernetes simplifies containerized Deployment?**

If you can think about a typical application, it would essentially be a cluster of containers running across different hosts. These containers would need to communicate with each other. Because Kubernetes is a cloud agnostic and may run on private as well as, public providers it has to be ones choice to simplify the containerized deployment.

**26. Can you please tell me what some of the main advantages of Kubernetes is?**

With container orchestration tool Kubernetes, it has become easy for one to handle the containers. You may respond to different customer demands through deploying the applications in a faster manner and in a way, which is predictable. So there is:

• Automated rollback  
• Automated scheduling   
• Horizontal scaling  
• Auto healing capabilities

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

• The installation structure of Kubernetes is complex though if it were installed then the cluster would be robust. However, the Docker Swarm installation process happens to be simple though the cluster is not robust per se.   
• Kubernetes is incredibly scalable. It was essentially built for large scale. However, the Docker swarm scales are five times faster than Kubernetes and is very scalable.  
• Kubernetes may also do the process of the auto scaling though the Docker swarm cannot do the process of the auto scaling.

**28. Can you please tell me what a ‘Heapster’ is with Kubernetes?**

A Heapster refers to a cluster wide aggregator of data that is provided by Kubelet running on each one of the nodes. This container management tool is supported natively on Kubernetes clusters and runs as a pod in the same way as any other pod within the cluster would run. Because of this it discovers all nodes within the cluster and queries the usage information from the Kubernetes nodes.

**29. Can you please tell me what a Google Container Engine is?**

A Google Container Engine refers to an open source management platform for the Docker containers and the clusters; it was built by Google for engineer operations. The Kubernetes based engine supports the clusters that run within the Google’s public cloud services, which is often required for large-scale applications.

**30. What positive things can you say about clusters within Kubernetes?**

The fundamental thesis behind Kubernetes is it is possible to enforce the desired state management. As a result it is possible to feed the cluster services a particular configuration. This is going to then go to the cluster services to go out and run the configuration within the [configured] infrastructure. As such, the deployment file is going to have all of the configurations, which required nourishment within the cluster services. The file will also require feeding to the API and so it would mean the cluster services retained the means for scheduling the pods in the appropriate setting and making sure the appropriate pods are running. That way the worker nodes, Kubelet and the API make up the Kubernet cluster.

**31. Describe Minikube?**

This is a tool, which makes it more efficient to run the Kubernetes system from a local point. It can run be described as running a single node of Kubernetes cluster within a virtual machine or VM.

**32. What is Kubectl?**

Kubectl refers to the platform which an engineer can pass commands to the cluster. By doing so, it provides the CLI with the means to run commands against the Kubernetes cluster via different create and manage commands on the Kubernetes component.

**33. What is Kubelet?**

The Kublet refers to an agent service that runs on each node and allows the slave to communicate with the master node. Therefore, Kubelet works on the the containers that are provided to it within the PodSpec and makes sure the containers prescribed within the Podspec are health and running adequately.

**34. What are K8s?**

This is another term for Kubernetes, whereby (K-eight characters-S), would be the open source orchestration framework for the containerized applications. It just another name for what we call Kubernetes.

**35. Can you please tell me what Kube Proxy is?**

The Kubernetes network proxy is a command or agent can be run on each one of the Kubernetes nodes. The service cluster type IPs and ports are then found through some Docker links for compatible environmental elements that specify the ports which are opened by a service type proxy. This is an option, which allows cluster DNS for the IPS cluster.

**36. What is the process that runs on Kubernetes Master Node?**

Kube-apiserver process runs on Kubernetes master node.

**37. Can you discuss how the master node works in Kubernetes.**

Kubernetes master controls the nodes and the containers are within the nodes. These individual containers are stored within pods and inside each pod, based according to the configuration and requirements. Because of this, if the Kubernetes pods have to be deployed, then they may either be accessed using a user interface or command line tool. These pods would be scheduled to run on the nodes and based on the source requirements, the pods are allocated to see these nodes. The job of the kube APIserver is to make certain there is absolute communication between the Kubernetes node and its master components.

**38. What is the role of the Kube apiserver and the Kube scheduler?**

The Kube apiserver follows the scale out architecture plan and is the front end which comes to the master node control panel. That would expose all the APIs of the Kubernetes Master Node components. It is responsible for the establishment of communication between the Kubernetes node and the Kubernetes master components. The Kube scheduler is at its core, responsible for the distribution and management of the workload on the worker nodes. It selects the most suitable nodes to run the unscheduled pod depending on the resource needs and keeps track of the overall resource utilization. It makes certain the workload is not scheduled on the nodes that may already be full.

**39. Give a short description of the process validating the configured information for API objects like the Pods.**

Kube apiserver process verifies and modifies the data for the API services.

**40. Can you please tell me the general use of the Kube**[**controller manager**](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/)**.**

Very simply put, the Kube APIerver procedures verify and modify the data for the API objects.

**41. Kubernetes Objects are made up of?**

Kubernetes objects are made of Pod, Service and volume.

**42. Determine the Kubernetes controllers.**

The Kubernetes controllers include Deployment controller and Replicaset.

**43. What is ECTD?**

ECTD is written is a tool that is written in the Go Programming language (created by Google) and is a distributed key value store. Because of this function, the ECTD stores of the configuration data of the Kubernetes cluster which shows the state of the cluster at any time.

**44. Describe the different types of services within Kubernetes.**

• Cluster IP: this function exposes the services on a clusters internal IP address. It is also the default service type and makes the service only reachable from inside of the cluster.  
• Node Port: it is a Cluster IP service to which Node Port service is going to route and is automatically created. It also exposes the service on each Node IP at a static port.   
• External Name: this service maps the contents of the External Name field through returning a CNAME record with that particular value. There is no proxying of any sort, which is set up.  
• Load Balancer: this one exposes the services from an external perspective with the use of a [cloud provider’s](https://www.zdnet.com/article/top-cloud-providers-2018-how-aws-microsoft-google-ibm-oracle-alibaba-stack-up/) load balancer. The services to which the external load balancer are going to route are automatically created.

**45. Can you please tell me what the Cloud Controller manager is?**

The Cloud Controller manager is there for persistent storage, abstracting the clouds code base from the main Kubernetes code base and the overall management of the communication with underlying cloud services. It may be split into different containers depending on the cloud platform that is being run by the engineers. Then it allows the cloud vendors and Kubernetes code to be built upon without an element of dependency. Therefore, the cloud vendor can develop their code and then connect with the Kubernetes cloud controller manager while running Kubernetes. There are different forms of cloud controller manager and they include the following:

• Node controller: this one checks and confirms the node is deleted in a proper manner after having been stopped.  
• Volume controller: this one manages the storage volume and interacts with the cloud provider in order to orchestrate the volume.  
• Service controller: the service controller is there for the management of the cloud providing the load balancers.  
• Route controller: the route controller manages the traffic routes within the underlying cloud infrastructure.

**46. What is ingress Network and what are the ways in which it works?**

The Ingress Network is a collection of engineering rules which act as an entry point to the Kubernetes cluster. This allows for inbound connections that can be configured to provide the services externally through URLs (like POST endpoints), which are reachable through virtual hosting or through a load balancer. Ingress, therefore is an API object, which handles the external access to the services within a cluster usually through the means of HTTP and would be the best way of exposing service. The working of the ingress network can be illustrated through the following example. The two nodes have the POD and root network namespacing which is utilizing a Linux bridge. There should be a new virtual Ethernet device described as flannel0 (a network plugin) added to the root network. You might consider now that would like the packet to go from the first to the fourth type of pod.

• It is then passed to CBR0, and that is what brings the ARP request for allotting the destination and it is found none would have the destination IP address.  
• At this time the bridge may sends the packet to flannel0 as the node’s route table is configured with flannel0.  
• That mean the flannel daemon communicates with the API server of Kubernetes to know all of the pod IPs and their particular respective nodes for mapping for the pods IPs to the node IPs.   
• The job of the network plugin is to wrap the packet within a UDP that has extra headers. This subsequently changes the sources and destination IPs to their respective nodes. Then it sends the packet through eth0.   
• Because the route table is knowledgeable about the way to route traffic between the different nodes, it gives the packet to the destination node2.  
• The packet will get to eth0 of node2, and then goes back to flannel0 for de-capsulation. It would still emit back in the root network namespace. Additionally, the packet is forwarded to the Linux bridge to make an ARP request ascertaining the IP, which belonging to veth1. • The packet would cross the root network and reach the particular destination, which is Pod4.

**47. Can you please tell me what some of the disadvantages of Kubernetes is?**

• It is hard to install and configure   
• It takes time to start running and gain traction   
• There are no placements available as yet   
• It is not simple to manage the services

**48. Can you please tell me what a headless service is?**

A headless service is almost the same as a ‘Normal’ service, but does not necessarily have a Cluster IP associated with it. This allows you to directly reach the pods without having to access it through the proxy.

**49. Can you please tell me what federated clusters are?**

Multiple clusters may be managed as a single cluster or with the assistance of federated clusters. It is possible to create multiple clusters within the data center or cloud and then use the federation for controlling or managing them in one place. The federated clusters are able to achieve this by doing some of the following:

• Syncing resources across the different clusters: this keeps the resource sync across the different clusters in order to deploy the same deployment set across the different clusters.  
• Cross Cluster, discover: this provides the ability to have DNS and Load Balancer with backend from the participating clusters.

**50. What is the difference between the replication controller and a replica set?**

The replica set and replication controller generally do the same things for the engineering operators. Both of them have the job of making certain that a specified number of pod replicas are running at the same time. The difference is there with the use of selectors to replicate the pods. The replica Set utilizes Set Based selectors though the replication controllers utilize Equity Based selectors.

• Equity Based Selectors: this type of selector allows for the filtering of label keys and values. That means the equity-based selector is only going to search for the pods that are going to have the same exact phrase as compared to the label. An example in this case is considering the label key claims app=nginx.  
• Selector based selectors: this selector allows for the filtering of keys according to a set of values. That means the selector-based selector is going to search for the pods whose label has been mentioned within the set. For example, if the label key says app within (nginx, Apache or NPS). With this selector, if the application has any of the nginx, Apache or NPS settings then the selector is going to return it as true.

**51. Can you tell me how you get a static IP for a Kubernetes load balancer?**

The Kubernetes Master is allowed to assign a new IP static address. It is possible to get a static IP for Kubernetes load balancer through changing the DNS records.

**Kubectl** controls the Kubernetes Cluster. It is one of the key components of Kubernetes which runs on the workstation on any machine when the setup is done. It has the capability to manage the nodes in the cluster.

**Kubectl** commands are used to interact and manage Kubernetes objects and the cluster. In this chapter, we will discuss a few commands used in Kubernetes via kubectl.

**kubectl annotate** − It updates the annotation on a resource.

$kubectl annotate [--overwrite] (-f FILENAME | TYPE NAME) KEY\_1=VAL\_1 ...

KEY\_N = VAL\_N [--resource-version = version]

For example,

kubectl annotate pods tomcat description = 'my frontend'

**kubectl api-versions** − It prints the supported versions of API on the cluster.

$ kubectl api-version;

**kubectl apply** − It has the capability to configure a resource by file or stdin.

$ kubectl apply –f <filename>

**kubectl attach** − This attaches things to the running container.

$ kubectl attach <pod> –c <container>

$ kubectl attach 123456-7890 -c tomcat-conatiner

**kubectl autoscale** − This is used to auto scale pods which are defined such as Deployment, replica set, Replication Controller.

$ kubectl autoscale (-f FILENAME | TYPE NAME | TYPE/NAME) [--min = MINPODS] --

max = MAXPODS [--cpu-percent = CPU] [flags]

$ kubectl autoscale deployment foo --min = 2 --max = 10

**kubectl cluster-info** − It displays the cluster Info.

$ kubectl cluster-info

**kubectl cluster-info dump** − It dumps relevant information regarding cluster for debugging and diagnosis.

$ kubectl cluster-info dump

$ kubectl cluster-info dump --output-directory = /path/to/cluster-state

**kubectl config** − Modifies the kubeconfig file.

$ kubectl config <SUBCOMMAD>

$ kubectl config –-kubeconfig <String of File name>

**kubectl config current-context** − It displays the current context.

$ kubectl config current-context

#deploys the current context

**kubectl config delete-cluster** − Deletes the specified cluster from kubeconfig.

$ kubectl config delete-cluster <Cluster Name>

**kubectl config delete-context** − Deletes a specified context from kubeconfig.

$ kubectl config delete-context <Context Name>

**kubectl config get-clusters** − Displays cluster defined in the kubeconfig.

$ kubectl config get-cluster

$ kubectl config get-cluster <Cluser Name>

**kubectl config get-contexts** − Describes one or many contexts.

$ kubectl config get-context <Context Name>

**kubectl config set-cluster** − Sets the cluster entry in Kubernetes.

$ kubectl config set-cluster NAME [--server = server] [--certificateauthority =

path/to/certificate/authority] [--insecure-skip-tls-verify = true]

**kubectl config set-context** − Sets a context entry in kubernetes entrypoint.

$ kubectl config set-context NAME [--cluster = cluster\_nickname] [--

user = user\_nickname] [--namespace = namespace]

$ kubectl config set-context prod –user = vipin-mishra

**kubectl config set-credentials** − Sets a user entry in kubeconfig.

$ kubectl config set-credentials cluster-admin --username = vipin --

password = uXFGweU9l35qcif

**kubectl config set** − Sets an individual value in kubeconfig file.

$ kubectl config set PROPERTY\_NAME PROPERTY\_VALUE

**kubectl config unset** − It unsets a specific component in kubectl.

$ kubectl config unset PROPERTY\_NAME PROPERTY\_VALUE

**kubectl config use-context** − Sets the current context in kubectl file.

$ kubectl config use-context <Context Name>

**kubectl config view**

$ kubectl config view

$ kubectl config view –o jsonpath='{.users[?(@.name == "e2e")].user.password}'

**kubectl cp** − Copy files and directories to and from containers.

$ kubectl cp <Files from source> <Files to Destinatiion>

$ kubectl cp /tmp/foo <some-pod>:/tmp/bar -c <specific-container>

**kubectl create** − To create resource by filename of or stdin. To do this, JSON or YAML formats are accepted.

$ kubectl create –f <File Name>

$ cat <file name> | kubectl create –f -

In the same way, we can create multiple things as listed using the **create**command along with **kubectl**.

* deployment
* namespace
* quota
* secret docker-registry
* secret
* secret generic
* secret tls
* serviceaccount
* service clusterip
* service loadbalancer
* service nodeport

**kubectl delete** − Deletes resources by file name, stdin, resource and names.

$ kubectl delete –f ([-f FILENAME] | TYPE [(NAME | -l label | --all)])

**kubectl describe** − Describes any particular resource in kubernetes. Shows details of resource or a group of resources.

$ kubectl describe <type> <type name>

$ kubectl describe pod tomcat

**kubectl drain** − This is used to drain a node for maintenance purpose. It prepares the node for maintenance. This will mark the node as unavailable so that it should not be assigned with a new container which will be created.

$ kubectl drain tomcat –force

**kubectl edit** − It is used to end the resources on the server. This allows to directly edit a resource which one can receive via the command line tool.

$ kubectl edit <Resource/Name | File Name)

Ex.

$ kubectl edit rc/tomcat

**kubectl exec** − This helps to execute a command in the container.

$ kubectl exec POD <-c CONTAINER > -- COMMAND < args...>

$ kubectl exec tomcat 123-5-456 date

**kubectl expose** − This is used to expose the Kubernetes objects such as pod, replication controller, and service as a new Kubernetes service. This has the capability to expose it via a running container or from a **yaml** file.

$ kubectl expose (-f FILENAME | TYPE NAME) [--port=port] [--protocol = TCP|UDP]

[--target-port = number-or-name] [--name = name] [--external-ip = external-ip-ofservice]

[--type = type]

$ kubectl expose rc tomcat –-port=80 –target-port = 30000

$ kubectl expose –f tomcat.yaml –port = 80 –target-port =

**kubectl get** − This command is capable of fetching data on the cluster about the Kubernetes resources.

$ kubectl get [(-o|--output=)json|yaml|wide|custom-columns=...|custom-columnsfile=...|

go-template=...|go-template-file=...|jsonpath=...|jsonpath-file=...]

(TYPE [NAME | -l label] | TYPE/NAME ...) [flags]

For example,

$ kubectl get pod <pod name>

$ kubectl get service <Service name>

**kubectl logs** − They are used to get the logs of the container in a pod. Printing the logs can be defining the container name in the pod. If the POD has only one container there is no need to define its name.

$ kubectl logs [-f] [-p] POD [-c CONTAINER]

Example

$ kubectl logs tomcat.

$ kubectl logs –p –c tomcat.8

**kubectl port-forward** − They are used to forward one or more local port to pods.

$ kubectl port-forward POD [LOCAL\_PORT:]REMOTE\_PORT

[...[LOCAL\_PORT\_N:]REMOTE\_PORT\_N]

$ kubectl port-forward tomcat 3000 4000

$ kubectl port-forward tomcat 3000:5000

**kubectl replace** − Capable of replacing a resource by file name or **stdin**.

$ kubectl replace -f FILENAME

$ kubectl replace –f tomcat.yml

$ cat tomcat.yml | kubectl replace –f -

**kubectl rolling-update** − Performs a rolling update on a replication controller. Replaces the specified replication controller with a new replication controller by updating a POD at a time.

$ kubectl rolling-update OLD\_CONTROLLER\_NAME ([NEW\_CONTROLLER\_NAME] --

image = NEW\_CONTAINER\_IMAGE | -f NEW\_CONTROLLER\_SPEC)

$ kubectl rolling-update frontend-v1 –f freontend-v2.yaml

**kubectl rollout** − It is capable of managing the rollout of deployment.

$ Kubectl rollout <Sub Command>

$ kubectl rollout undo deployment/tomcat

Apart from the above, we can perform multiple tasks using the rollout such as −

* rollout history
* rollout pause
* rollout resume
* rollout status
* rollout undo

**kubectl run** − Run command has the capability to run an image on the Kubernetes cluster.

$ kubectl run NAME --image = image [--env = "key = value"] [--port = port] [--

replicas = replicas] [--dry-run = bool] [--overrides = inline-json] [--command] --

[COMMAND] [args...]

$ kubectl run tomcat --image = tomcat:7.0

$ kubectl run tomcat –-image = tomcat:7.0 –port = 5000

**kubectl scale** − It will scale the size of Kubernetes Deployments, ReplicaSet, Replication Controller, or job.

$ kubectl scale [--resource-version = version] [--current-replicas = count] --

replicas = COUNT (-f FILENAME | TYPE NAME )

$ kubectl scale –-replica = 3 rs/tomcat

$ kubectl scale –replica = 3 tomcat.yaml

**kubectl set image** − It updates the image of a pod template.

$ kubectl set image (-f FILENAME | TYPE NAME)

CONTAINER\_NAME\_1 = CONTAINER\_IMAGE\_1 ... CONTAINER\_NAME\_N = CONTAINER\_IMAGE\_N

$ kubectl set image deployment/tomcat busybox = busybox ngnix = ngnix:1.9.1

$ kubectl set image deployments, rc tomcat = tomcat6.0 --all

**kubectl set resources** − It is used to set the content of the resource. It updates resource/limits on object with pod template.

$ kubectl set resources (-f FILENAME | TYPE NAME) ([--limits = LIMITS & --

requests = REQUESTS]

$ kubectl set resources deployment tomcat -c = tomcat --

limits = cpu = 200m,memory = 512Mi

**kubectl top node** − It displays CPU/Memory/Storage usage. The top command allows you to see the resource consumption for nodes.

$ kubectl top node [node Name]

The same command can be used with a pod as well.