1. **What is pod:**

A Pod is the smallest deployable unit that can be deployed and managed by Kubernetes. In other words, if you need to run a single container in Kubernetes, then you need to create a Pod for that container. At the same time, a Pod can contain more than one container, if these containers are relatively tightly coupled. In a pre-container world, they would have executed on the same server.

<https://linchpiner.github.io/k8s-multi-container-pods.html>

1. What is K8 :

Kubernetes (K8s) is an open-source container orchestration system for automating deployment, scaling, and management of containerized applications.

1. **What is Deployments vs StatefulSets:**

Deployment is the easiest and most used resource for deploying your application. It is a Kubernetes controller that matches the current state of your cluster to the desired state mentioned in the Deployment manifest. e.g. If you create a deployment with 1 replica, it will check that the desired state of ReplicaSet is 1 and current state is 0, so it will create a ReplicaSet, which will further create the pod. If you create a deployment with name counter, it will create a ReplicaSet with name counter-<replica-set-id>, which will further create a Pod with name counter-<replica-set->-<pod-id>.

Deployments are usually used for stateless applications. However, you can save the state of deployment by attaching a Persistent Volume to it and make it stateful, but all the pods of a deployment will be sharing the same Volume and data across all of them will be same.

A StatefulSet is another Kubernetes controller that manages pods just like Deployments. But it differs from a **Deployment in that it is more suited for stateful apps**. A stateful application requires pods with a unique identity (for example, hostname). One pod should be able to reach other pods with well-defined names.

**Stable, unique name**

**Stable. PVC**

**Ordered, creation and scalling**

<https://medium.com/stakater/k8s-deployments-vs-statefulsets-vs-daemonsets-60582f0c62d4>

1. What is ingress

An API object that manages external access to the services in a cluster, typically HTTP.

[Ingress](https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#ingress-v1-networking-k8s-io) exposes HTTP and HTTPS routes from outside the cluster to [services](https://kubernetes.io/docs/concepts/services-networking/service/) within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

An Ingress may be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting.

**Ingress**,In applications **we can route the traffic to the diff ports using same DNS**. Whereas **load balancer can expose to one port for one service**.

1. What is nginx
2. What is readiness vs liveness probes

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>

**Liveness and readiness** probes are the tools to monitor the pods and its health and take appropriate actions in case of failures.

* **Liveness** Probes: checks your containers are alive
* **Readiness** Probes: checks your containers are able to do productive work

1. **Liveness** :

A liveness probe sends a signal to OpenShift that the container is either alive (passing) or dead (failing). If the container is alive, then OpenShift does nothing because the current state is good. If the container is dead, then OpenShift attempts to heal the application by restarting it.

1. **Readiness:**

OpenShift [services](https://docs.openshift.com/container-platform/3.11/architecture/core_concepts/pods_and_services.html#services) use readiness probes to know whether the container being probed is ready to start receiving network traffic. If your container enters a state where it is still alive but cannot handle incoming network traffic (a common scenario during startup), you want the readiness probe to fail. That way, OpenShift will not send network traffic to a container that isn't ready for it. If OpenShift did prematurely send network traffic to the container, it could cause the load balancer (or router) to return a 502 error to the client and terminate the request; either that or the client would get a "connection refused" error message.

If the liveness probe succeeds while the readiness probe fails, OpenShift knows that the container is not ready to receive network traffic but is working to become ready.

1. What is [Chaoskube :](https://kubedex.com/resource/chaoskube/#:~:text=Chaoskube%20is%20an%20open%20source,pods%20in%20your%20Kubernetes%20cluster.)

Chaoskube is an open source Chaos Testing tool. chaoskube periodically kills random pods in your Kubernetes cluster.

1. K8 realtime prob and solution :
   1. **Problem** : whenever we migrated k8 cluster to newer version somehow all the nodes were showing unhealthy(not ready state).
   2. **Solution** : we have communicated with support team and implementation team to find out why were nodes stoped.so after restarted also it never come up. At finally we came to know after long debug we saw that callico network pods are not running due to that all the nodes are in not ready state.

**SO** we described the calico pods to see the errors and find that the issue with **resource quota**. It should require min 100M to start CNI calico pods up but we have given 10M for them as they were small pods.

**Kubectl get pods –all-namespaces | grep -f calico**

**Kubectl describe deployment calico-kube-controllers -n kube-system**

1. Kubectl drain :

You can use kubectl drain to **safely evict all of your pods from a node before you perform maintenance on the node** (e.g. kernel upgrade, hardware maintenance, etc.). ... It is then safe to bring down the node by powering down its physical machine or, if running on a cloud platform, deleting its virtual machine.

<https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/>

1. What is kubectl cordon?

**Cordon the existing node pool**: This operation marks the nodes in the existing node pool ( default-pool ) as unschedulable. **Kubernetes stops scheduling new Pods to these nodes once you mark them as unschedulable.**

First will create 3 nodes with 1.8 version and will drian one of the existing 1.6 version node and other 2 we will cordon which stops scheduling new pods to 1.6 so that new schedules will go to newer version nodes.

1. what is taint and toleration :

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes

1. **kubectl taint nodes node1 key1=value1:NoSchedule**
2. When you submit a workload to run in a cluster, the scheduler determines where to place the Pods associated with the workload. The scheduler is free to place a Pod on any node that satisfies the Pod's CPU, memory, and custom resource requirements.
3. If your cluster runs a variety of workloads, you might want to exercise some control over which workloads can run on a particular pool of nodes.
4. A **node taint** lets you mark a node so that the scheduler avoids or prevents using it for certain Pods. A complementary feature, *tolerations*, lets you designate Pods that can be used on "tainted" nodes (***tolerations will schedule workloads on taint nodes only, not other nodes***).

Node taints are *key-value pairs* associated with an *effect*. Here are the available effects:

1. NoSchedule: Pods that do not tolerate this taint are not scheduled on the node; existing Pods are not evicted from the node.
2. PreferNoSchedule: Kubernetes avoids scheduling Pods that do not tolerate this taint onto the node.
3. NoExecute: Pod is evicted from the node if it is already running on the node, and is not scheduled onto the node if it is not yet running on the node.
4. **node affinity an pod affinity**
5. You can constrain a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/) so that it can only run on particular set of [Node(s)](https://kubernetes.io/docs/concepts/architecture/nodes/). There are several ways to do this and the recommended approaches all use [label selectors](https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/) to facilitate the selection.
6. nodeSelector is the simplest recommended form of node selection constraint. nodeSelector is a field of PodSpec. It specifies a map of key-value pairs. For the pod to be eligible to run on a node

🡪 kubectl label nodes kubernetes-foo-node-1.c.arobinson.internal **disktype=ssd.**

**nodeSelector**:

**disktype: ssd**

1. **Affinity:**

Node affinity is conceptually similar to nodeSelector -- it allows you to constrain which nodes your pod is eligible to be scheduled on, based on labels on the node.

There are currently two types of node affinity, called requiredDuringSchedulingIgnoredDuringExecution and preferredDuringSchedulingIgnoredDuringExecution. You can think of them as "hard" and "soft" respectively, in the sense that the former specifies rules that *must* be met for a pod to be scheduled onto a node (similar to nodeSelector but using a more expressive syntax), while the latter specifies *preferences* that the scheduler will try to enforce but will not guarantee. 

**spec**:

**affinity**:

**nodeAffinity**:

**requiredDuringSchedulingIgnoredDuringExecution**:

**nodeSelectorTerms**:

- **matchExpressions**:

- **key**: kubernetes.io/e2e-az-name

**operator**: In

**values**:

- e2e-az1

- e2e-az2

**preferredDuringSchedulingIgnoredDuringExecution**:

- **weight**: 1

**preference**:

**matchExpressions**:

- **key**: another-node-label-key

**operator**: In

**values**:

- another-node-label-value

b) pod affinity :

Pod affinity/anti-affinity allows you to constrain which nodes your pod is eligible to be scheduled on based on the labels on other pods. A [label](https://docs.openshift.com/container-platform/3.6/architecture/core_concepts/pods_and_services.html#labels) is a key/value pair.

Pod affinity can tell the scheduler to locate a new pod on the same node as other pods if the label selector on the new pod matches the label on the current pod.

**spec:**

**affinity:**

**podAffinity:**

**requiredDuringSchedulingIgnoredDuringExecution:**

**- labelSelector:**

**matchExpressions:**

**- key: security**

**operator: In**

**values:**

**- S1**

**topologyKey: failure-domain.beta.kubernetes.io/zone**

**containers:**

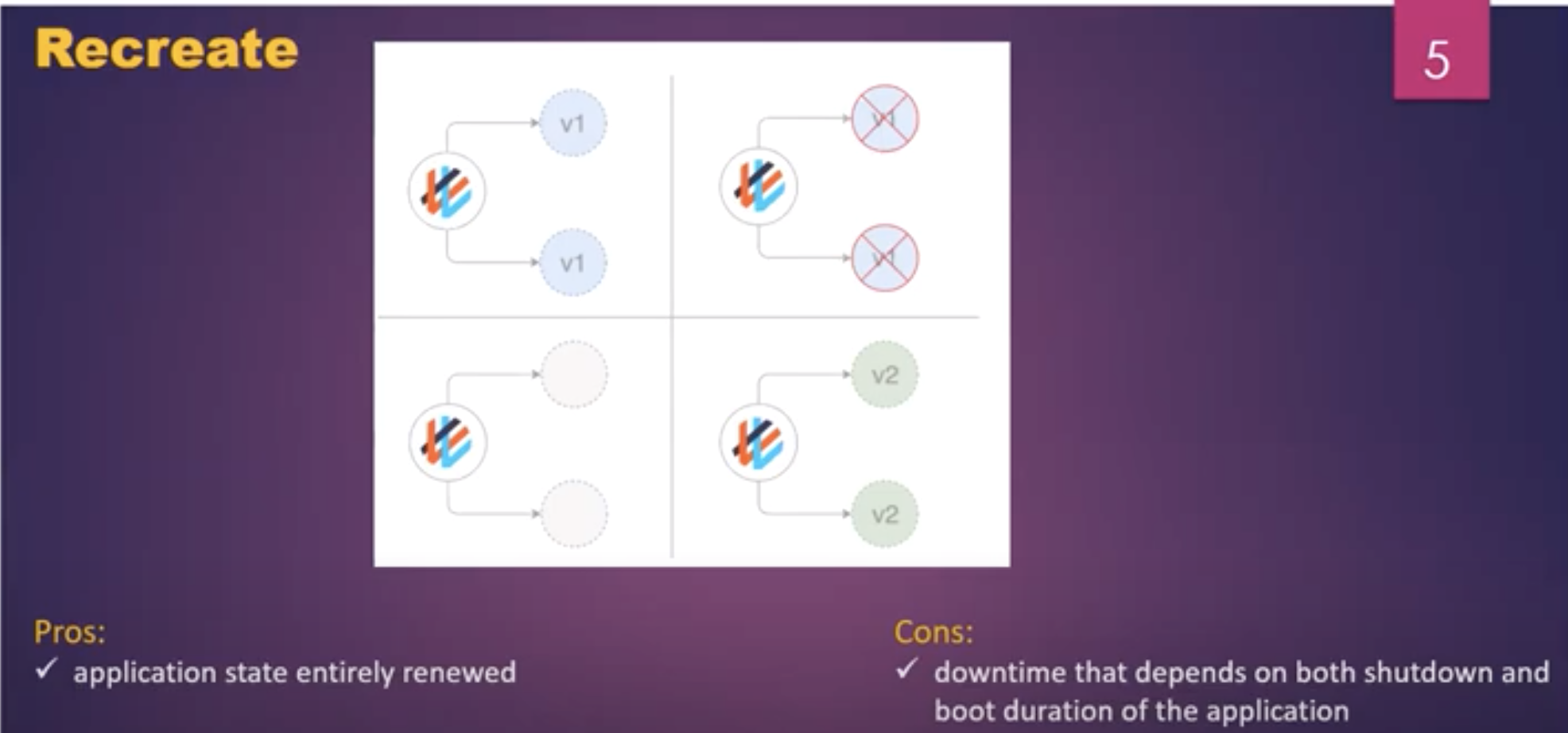
**- name: with-pod-affinity**

**image: docker.io/ocpqe/hello-pod**

**example** : if we hav]e cache / in cache memory process so wherever we have these incache memeroy we have to deploy our webservers. So with help of pod affinity we can deploy.

14 ) how to check the logs of a pod or deployment?

1. kubectl get pod nginx-deployment-1006230814-6winp -o yaml
2. kubectl get events --namespace=my-namespace
3. kubectl logs **${**POD\_NAME**}** **${**CONTAINER\_NAME**}**
4. kubectl logs --previous **${**POD\_NAME**}** **${**CONTAINER\_NAME**}**



1. what are deployment strategy used in your project
2. #**1 Recreate deployment model:**
   1. It’s simple deployment , it kills the older version of application and will create new version.
   2. There will be down time.
3. #**2 Rolling deployment :**
   1. it is default and standard deployment strategy in k8 . whenever we deploy new version pod , first the new version will start creating and upon successful and then the older version of pod will come down. So that we will have 0 down time.
   2. Mainly uses for statefullset application, as the pods will come down slowly one after another.

A picture containing graphical user interface

Description automatically generated

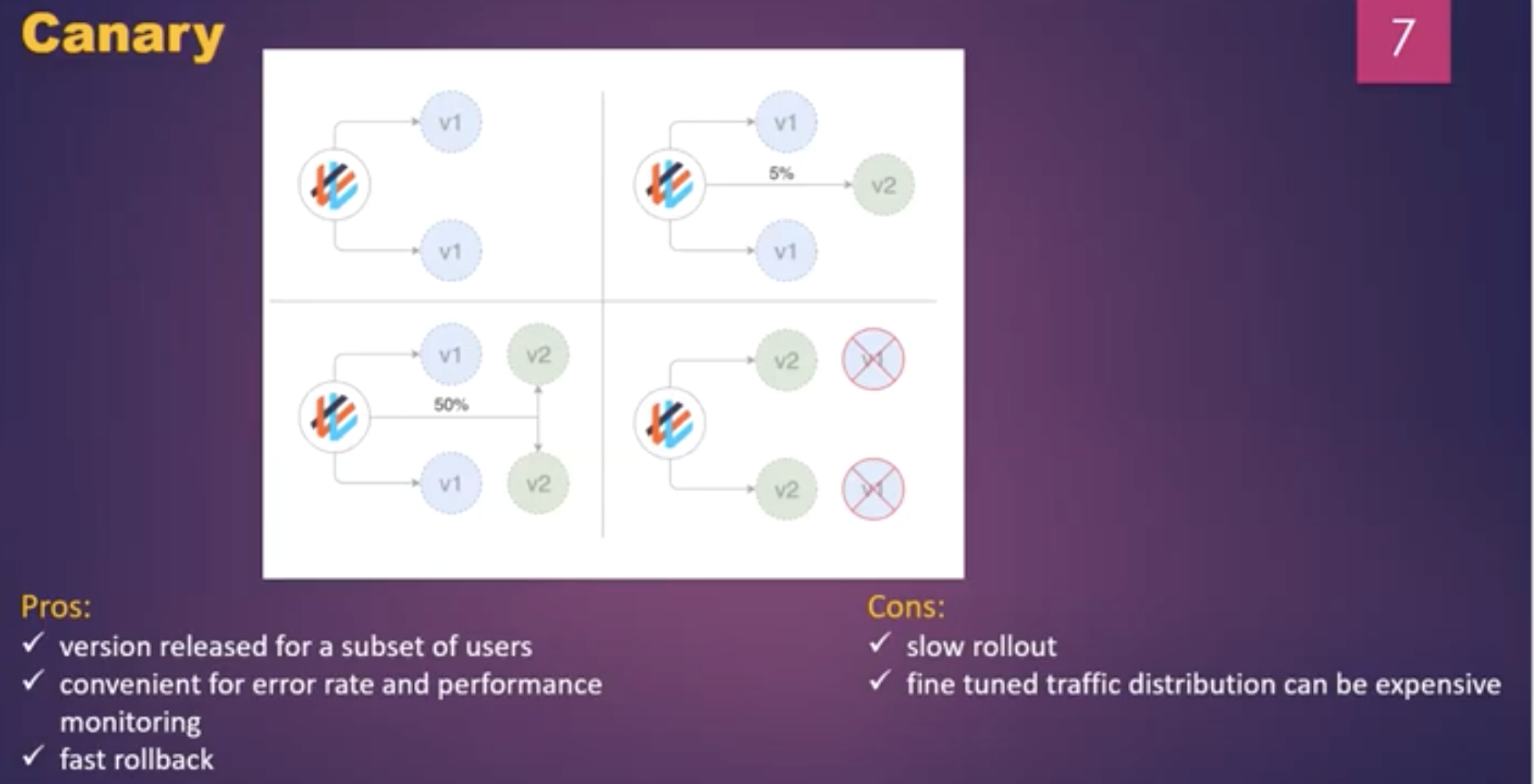
1. #**3 Blue/green deployment pattern:**
   1. We will have 2 versions of deployments at the same time V1 and V2, V1 will be for user and V2 is for testing once V2 is tested properly by QA team, then we will switch V2 as V1 for consumers and V1 will be scaling down.

Graphical user interface

Description automatically generated

Kubectl patch command to switch b/w the versions.

1. **#4 Canary deployment:**
   1. Canary deployments are **a pattern for rolling out releases to a subset of users or servers**. The idea is to first deploy the change to a small subset of servers, test it, and then roll the change out to the rest of the servers.

****

1. What is service in OSE/K8:

A Kubernetes [service](http://kubernetes.io/docs/user-guide/services) serves as an internal load balancer. It identifies a set of replicated [pods](https://docs.openshift.com/container-platform/3.11/architecture/core_concepts/pods_and_services.html#pods) in order to proxy the connections it receives to them.

The default service **clusterIP** addresses are from the OpenShift Container Platform internal network and they are used to permit pods to access each other.

**clusterIP** – example we have DB’s which sholdnt access out side world. So we create a service for db with clusterIP which we can access inside cluster. Which is no exposed outside and which internally have connection b/w ports.

1. what is init container

An [init container](https://kubernetes.io/docs/concepts/workloads/pods/init-containers/) is a container in a pod that is started before the pod app containers are started. Init containers can share volumes, perform network operations, and perform computations before the remaining containers start. Init containers can also block or delay the startup of application containers until some precondition is met.

When a pod starts, after the network and volumes are initialized, the init containers are started in order. Each init container must exit successfully before the next is invoked. If an init container fails to start (due to the runtime) or exits with failure, it is retried according to the pod [restart policy](https://docs.openshift.com/container-platform/3.11/architecture/core_concepts/pods_and_services.html#admin-manage-pod-restart).

**A pod cannot be ready until all init containers have succeeded.**

**spec:**

**containers:**

**- name: myapp-container**

**image: busybox**

**command: ['sh', '-c', 'echo The app is running! && sleep 3600']**

**initContainers:**

**- name: init-myservice**

**image: busybox**

**command: ['sh', '-c', 'until nslookup myservice; do echo waiting for myservice; sleep 2; done;']**

1. What is sessionAffinity Kubernetes?

sessionAffinity to “ClientIP”, **Kubernetes will direct traffic from any given controller to a specific backend pod.** This can be customized further by having the ability to set timeouts for these sticky sessions.

Session affinity overrides the load-balancing algorithm by **directing all requests in a session to a specific application server**

1. **Rolling deployment**

A rolling update deployment pattern allows continuous deployment in the active application. The strategy reduces app downtime, which is crucial for critical applications.

1. difference between headfull and headless service

A headless service is **a service with a service IP but instead of load-balancing it will return the IPs of our associated Pods**. This allows us to interact directly with the Pods instead of a proxy. It's as simple as specifying None for.

Sometimes you don't need load-balancing and a single Service IP. In this case, you can create what are termed "headless" Services, by explicitly specifying "**None**" for the cluster IP (.spec.**clusterIP**).  
 **headless service is similar to the normal services, but doesn’t have a cluster IP instead it uses headless service. this service enables you to directly reach pods without needed of accessing it thru proxy.**

When u create a service with clusterIP/nodeport it will act as a load balancer when u send request it will go to any pod. Where if I don’t want that kind of behaviour like pod to pod communication. if I send a request it will go/redirect it to particular pod in that case will go with headless.

When we use headless it will create DNS entry for each and every POD and by which will able to reach/communicate to each pod separately. Mainly it

Example : mysql db with 3 replications u wil have single service for each pod with headless.

Pod to pod communication or if we want to communicate perticluar pod and then copy that perticluar pod data to across other replicas to sync…….this is the use of headless

<https://dev.to/kaoskater08/building-a-headless-service-in-kubernetes-3bk8>

1. how to rollback the deployment?

kubectl rollout **undo** deployment.v1.apps/nginx-deployment **--to-revision=2**

Rollback to an earlier Deployment revision if the current state of the Deployment is not stable. Each rollback updates the revision of the Deployment.

10) difference between secret and configmap

22) pod is in pending state ,what are the possible reasons?

If a Pod is stuck in Pending it means **that it cannot be scheduled onto a node**. Generally this is because there are insufficient resources of one type or another that prevent scheduling. Look at the output of the kubectl describe/logs.

1. Kubernetes architecture
2. Copying local application/config files to application pods :

Using configmaps we

Using healm, by packaging them and loading them into pods

1. What is static pods :
2. Why deamonsets are required in k8
3. Pod level High availability - probes
4. Can you run pods on master node?

If you want to be able to schedule pods on the Kubernetes control-plane node, you **need to remove a taint on the master nodes**. This will remove the node-role.kubernetes.io/master taint from any nodes that have it, including the control-plane node, meaning that the scheduler will then be able to schedule pods everywhere

1. resources vs resource quota

LimitRange/**resource is for managing constraints at a pod and container level within the project.** An individual Pod or Container that requests resources outside of these LimitRange constraints will be rejected, whereas a **ResourceQuota** **only applies to all of the namespace**/project's objects in aggregate

1. calico n/w in k8:

it’s a n/w rules, example if we have 4 nodes if we schedule one pod with 10.00.0.1 and if we schedule other pod on other node which shouldn’t be conflicted with IP. So these kind of things calico will take care.

1. RBAC:

We can enable system what can be access or what cant be access depending on role.

1. Volumes in K8 PV vs PVC vs empty dir:

So a persistent volume (PV) is the "physical" volume on the host machine that stores your persistent data. A persistent volume claim (PVC) is a *request* for the platform to create a PV for you, and you attach PVs to your pods via a PVC

Pod -> PVC -> PV -> Host machine

admin will create some PV space for you. and then we will request some space by using PVC for your POD.

**Persistent Volume** - Available storage let's say you have 100Gi

**Persistent Volume Claim** – pvc is a request for storage by a user which can be attained from PV, let's say you request 10Gi you'll get it but if you request 110Gi you won't get it.

It is similar to a Pod. Pods consume node resources and PVCs consume PV resources.

**Empty Dir volume**: is local volume system. In order to use any files in init containers /application, we use empty dir by creating mount path.

1. Can we have similar types of container is same pod? Like can we have two nginx containers into same pod

Same type of containers are not possible, we can have diff type of containers in same pod.

1. Diff types of containers?
   1. Init containers
   2. Sidecar containers
2. Scallings in k8 – when app traffic grows how ur dealing ? and on what basis we are gonna scale pods?
   1. **VPA** – it increases the memory to existing pod, it wil delete and recreate with new resources
   2. **HPA** – horizontal pod autoscaler :
      * whenever we setup HPA , It will communicate with metrics server and the metrics server will communicate to kube API server to get the data.
      * Depeneds on conditions we can auto spin up the pods like if the resource limit reaches or cpu usage reaches 80% of the usage, the HPA will scale the pods.
3. If u got a requirement , which one will you choose KubeADM or Cloud provider?
4. Lets say we have application and db , so what are all the objects u use ?

Deployment

Service

Service with clusterIP – dp

Statefullset for - DB

PVC, PV

Configmaps

1. Why cant we use statefullsets for applications as well. Why we are using deployment for **applications** ?
2. What is **S**ervice Account?

If have 5 teams, which contains BA,developer,QA ppl, where everyone shouldn’t have same account

1. What is a Kubernetes operator?

A [Kubernetes operator](https://www.redhat.com/en/resources/oreilly-kubernetes-operators-automation-ebook?intcmp=701f2000001OMH6AAO) is a method of packaging, deploying, and managing a Kubernetes application. A Kubernetes application is both deployed on [Kubernetes](https://www.redhat.com/en/topics/containers/what-is-kubernetes) and managed using the Kubernetes API (application programming interface) and kubectl tooling

In Kubernetes, controllers of the control plane implement control loops that repeatedly compare the desired state of the cluster to its actual state. If the cluster's actual state doesn’t match the desired state, then the controller takes action to fix the problem.

**the operator, we can do Something customised work u wanted to like which is not available in k8. We design our own object and we can use it.**

1. What are the monitoring tools for K8
   1. Prometheus
   2. ELK
   3. grafana
2. diff b/w deployment vs statefullset vs replicaset
3. **deployment :** it is dependent on replica sets – whenever u deploy deployment it uses replica sets to create pods.
   * + We use it for versioning’s
     + Rolling out new version
     + Roll back previous version
     + Use it web applications
4. **Replica sets** : defining how many pods we wanted to create for our application. The application always runs the same num of replica sets even if someone killed , automatically it will start again
5. **Statefullset** : deployment and stateful sets are mostly similar , but the only changes the stateful set pods names have indexes example if we give pod name nginx. The pod names will start nginx-0, nginx-1 like this and if any pod goes down automatically the pod will come with same pod name, unlike deployment where names get changes after restart.
   * + **Use it for databases , where if we wanted to store pod data we wil go with stateful sets.**
6. **Do u think the IP will change when the pod is deleted/restarted in stateful set?**
7. Services will not be deleted right, the DNS name is mater it will remain same.
8. For a StatefulSet to work, it needs a Headless Service. A Headless Service does not have an IP address. Internally, it creates the necessary endpoints to expose pods with DNS names. The StatefulSet definition includes a reference to the Headless Service, but you have to create it separately.
9. The Pods' ordinals, hostnames, SRV records, and A record names have not changed, **but the IP addresses associated with the Pods may have changed.** In the cluster used for this tutorial, they have. This is why it is important **not to configure other applications to connect to Pods in a StatefulSet by IP address**
10. What are the reasons to not spin up the pod in particular node?
11. Taint and tolerations to the node. If the node is tainted so pod will not schedule on specific node
12. Can be Resources – if the resource limit is exceeded the pod wil never come up
13. **HostPort** if its been already used , if we tried to re use it amd the pod will not come up
14. Might be image name wrong in yaml file.
15. what are the basic check we do if the container mot starts.

We can do by doing basic level loggigs like Descibing the pod to see and we can see logs of the pod.

1. How can we see the entire logs of the cluster.

We have folder where we store entire logs of the cluster but I don’t remember the name. but using fileBeat we can read the logs of the files and we can config it to ELK system. And we can see entire log info.

1. What are the components in k8:
2. **Master**:
   1. Each component of the master server is responsible for certain activity.
      * 1. **Controller manager** 🡪 we have so many controllers, it is responsible for whether the desired state is achieved or not. If we have 3 instances, those 3 should running all the time. If any of the pod is down and the current state is 2. it is the duty of the controller bring that down replica. To match with desired state.
        2. **ETCD** 🡪 key-value database which store node and pod info – it stores the entire info about the cluster .in case of cluster goes down and when it comes back with the help of ETCD the cluster come to previous state.

Let’s say any new node has created in the cluster ETCD has the info about it.

* + - 1. **Scheduler**  🡪 scheduling the workloads (not running the workloads), exp: whenever we are deploying application with 2 replicas/instances where these replicas are should run in cluster. Based on the container resources we specify the schedular will identify which node has bandwidth and schedule the pods. By default, MASTER wont schedule anything bcse its tainted. The reason its tainted the master node is mainly dedicated to maintaining control plane.
      2. **API server** 🡪 acts like a front end server for master,like master to the master. If I want to talk to the cluster its way thru API server using KUBECTL utility. Anything should access thru API server.

1. **Node:**
   * 1. **Kubelet 🡪**  they are like brokers which runs on every node. Only kubelet only talks to master. its responsible for health status of the pods/nodes to master.

* + 1. Kube-proxy : setting up the IP table rules. Exam : if we have 3 instances with ClusterIp service. Whenever a request comes to the service has to forwarded to any of the instance.

1. What if the master node goes down?

We have HA – high availability fashion in prod env, if one master node goes down then the other will come into picture and will act as a leader.

1. What is the local vs Remote storage:
2. **Local** : we need the data when the pod is there. After restarts if its not required wil go with local. Its available with pod lifecycle . example for config maps
3. **Remote**: to have ur storage even after retartes or crashes the pods. So wil go with remote . there are many remote storages available – NFS,AZURE,AWS,GCP ..etc
4. Adapter containers and side-car container
5. Side car container will help to copy any application logs/files to external sources. Like copying centralized applications log files to splunk.
6. Adapter container is used for transform the data to make it understandable . like we do using filebeat and logstash we read the logs and convert them to understandable.
7. Pod distruption budget

PDB is an limit that we can set to application,

**example** : in our application if we specify PBD is 4 pods for our application the application never come down then 4 pods. It always runs 4 pods even though there are disasters/failures/deletion of deployment.

When we apply PDB, even though we tried to delete deployment it won’t allow to delete bcse PBD is applied for 2 pods out of 4. 2 will be deleted ad another still be running.

1. What is kube-proxy
2. How to set static IP for k8 load balancer.
3. How to check what are all the resources/objects in k8.

Kubectl api-resources – this will give all the resources that are available

Kubectl explain object- which gives the desc of objects

1. If I have front end application and backemd DB – can u architect what are all the objects we use?

A picture containing text, whiteboard

Description automatically generated

1. Application –

* Will go with deployment hence it is stateless application. And with 3 replica sets for my application
* To expose the deployment, we will have different services (load balancer, nodePort, ingress)
* If any of the pods killed, the controller will start the pod again with diff name of the pod.

1. StatefullSet :

* for statefull applications we always go with statefullSet.
* If we have 3 replicas, 1 master – where happened actual data writes and 2 nodes – where read the data from master and copy it in each node.
* With statefullsets the advantage is we can have state of the pods. The pods will have defined name/unique names even after restrts the name will be same.
* When we go for statefulset is, for example always nodes takes the copy of data to their node from master. What if the master node goes down the other nodes will not able to read the data from master and after starts the node again it will come up with same name (Node-0 like this). So that the nodes will copy the data once master is up. Nodes will always look for the same name of master node.
* If the names change for master the nodes can’t find the master to have data sync.so that’s why we go for statefull deployment for DB’s.
* And also if we scale down the db’s cluster, it always scale down the pods from last index.

for writing the data to master we use, any write request it should go to master only – **Headless**

for reading data from nodes, we use the read request goes to any of the pod- **clusterIp**

1. Pod Restart Policy

A pod restart policy determines how OpenShift Container Platform responds when containers in that pod exit. The policy applies to all containers in that pod.

* Always - Tries restarting a successfully exited container on the pod continuously, with an exponential back-off delay (10s, 20s, 40s) until the pod is restarted. The default is Always.
* OnFailure - Tries restarting a failed container on the pod with an exponential back-off delay (10s, 20s, 40s) capped at 5 minutes.
* Never - Does not try to restart exited or failed containers on the pod. Pods immediately fail and exit.

1. Labels and Selectors

*Labels* are key/value pairs that are attached to objects, such as pods. Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to user.

***Exmp****: if we have application which may need to us 20 objects in deployment like , pods,deployment,service, cluster role and role binding etc….*

*Label should be same across for this deployment so that its easy to filter or identify the resources related to the deployment by using selectors.*

The label selector is the core grouping primitive/resuorces in Kubernetes. using selecters we can filter the labels resources.

To see the lables of pods

**Kubectl get pods –show-lables**

To set labels

**Kubectl label pod podname env=dev**

To remove label

**Kubectl label pod podname -l env-**

To overwrite the label

**Kubectl label pod podname env=prod –overwrite**

What if we don’t provide label name in declerative/imperative mode

**Kubectl run nginx –image=nginx:latest** – the name of the images will come as a label by default.

### **Kubectl get po -l ‘env != prd’** 🡪 which will list pods other than PRD env. This is ***Equality-based*** *selector*

**Kubectl ge po -l ‘env notin (prd**)’ 🡪 this is other way of using selectors . this is ***Set-based.***

1. Istio in K8:
2. Can we have multiple conatiners in a pod? Can we have similar conatiners in a pod? Lets say i have 4 conatiners, one of them has failed how would you check which container has failed?

🡪 yes we have multiple diff containers in same pod.

🡪 we shouldn’t have similar kind of containers in pod.

🡪 we can check details of pods/ using **kubectl describe <pod>** command.

1. Can we deploy a pod on particular node?

🡪 yes, using various way.

1) using nodeSelector 2) nodeAffinity etc..

60) how to check container logs:

If we have 2 containers in one pod.

**Docker logs <pod> -c <container Name>**

how will you import secrets to azure keyvault

how will maintain k8 secrets and where will u keep secret files?

how will you migrate applications from on-prem to azure? like how u managed to deploy services

k8 security - limit of ip access kubectl , we can use sp for deployments

csi driver kubernetes

k8 rbac vs azure rbac vs local rbac

k8 rbacs (role,rolebinding, cluster tole, clusterrolebinding, serviceAccount) -<https://www.youtube.com/watch?v=BLktpM--0jA&t=432s>

k8 upgrade -<https://www.youtube.com/watch?v=Afl72C-FEMg> (version upgrade, node image upgrade)

k8 nodepools (system, user node pools(linux, windows,etc..))

k8 scaling - HPA

azure functionsapp- yet

k8 statefullset - yet

k8 deployment strategy

k8 pv , pvc, storage classes - https://www.youtube.com/watch?v=ggIEe8wPGJM

k8 kubenet vs cni -<https://www.youtube.com/watch?v=A-eH_fu-pXg&list=PLp_fsLj4v7gTA8n2UngXfOc_5puPPmFdK&index=8>

k8 pod erros

application gateway vs ingress controler?

**k8 upgrade** - we are using cni with subnet(Ips) and have nodepools allocated for nodes and pods and when we try to upgrade cluster from unsupported version its not allow us to upgrated with 2 reasons(unsupported version and no ips in subnet). we have option now that we can have another subnet and we updrage all applications to diff subnet . but in this case bcse we have unsuported version we cant have in diff subnet all we have to do is down some application to free up ips and do upgrade.