https://jamesdefabia.github.io/docs/api/

<https://www.youtube.com/watch?v=wtKef83kmUA&list=PL0hSJrxggIQoKLETBSmgbbvE4FO_eEgoB>

<https://www.youtube.com/watch?v=_bG7jJH10gI&list=PLd8alL65M1GZ2jyltUiH5rYFZ-Wx3vGrp>

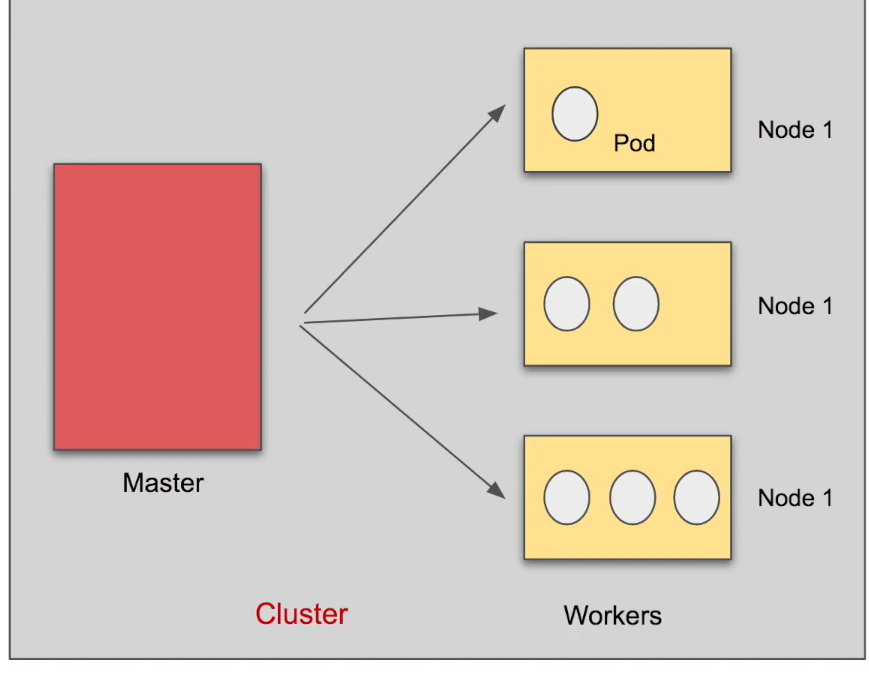
<https://wangwei1237.github.io/Kubernetes-in-Action-Second-Edition/docs/Introducing_Kubernetes_1.html>

K8s is a container orchestration engine which is used to automate the containerized applications.

Features:

* Automatic bin packing
* Service discovery and load balancing
* Storage orchestration
* Self-healing
* Automatic roll out and revert.
* Secret and config management
* Scaling

Architecture:



Diagram

Description automatically generated

► Get status of different components

► create a pod/deployment

► layers of abstraction

► change the pod/deployment

► debugging pods

► delete pod/deployment

► CRUD by applying configuration file

Shortforms:

* Po-pods
* Deploy-deployment
* Svc-service
* ns-namespaces
* ing-ingresses

useful commands:

* minikube start --vm-driver=<vm\_name>
* kubectl get nodes -> get the status of all the nodes present in the cluster
* kubectl create deployment <name> --image=<Img-name>
* kubectl create -f <filename>
* kubectl get replicaset
* kubectl edit deployment <depl-name> -> to change the deployment configuration.
* kubectl logs <pod-name>
* kubectl exec -> like in docker
* kubectl delete deployment <depl-name>
* kubectl delete -f <file-name>
* kubectl explain <resource>.<res>.<res>……..

k8s config file has 3 main parts:

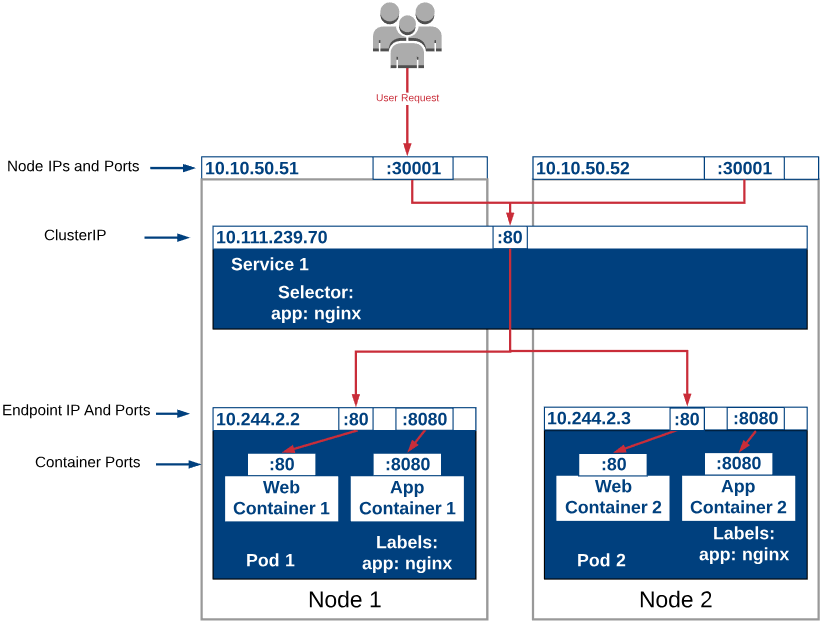
1. meta data
2. spec
3. status

to know the api version:

* kubectl api-resources -> gives all the resources.
* Kubectl api-versions -> gives the versions.

Namespace:

* kubectl get po --namespace kube-system
* -n -> --namespace
* -A -> --all-namespaces
* kubectl create namespace <name>
* kubectl apply -f <file> -n <name-space>
* kubectl config set-context --current --namespace <name>



* nodePort : The port on the node where external traffic will come in on
* port : The port **of this service**
* targetPort: The target port on the pod(s) to forward traffic to

init container: execute before our container started and delete .

To avoid the down time:

* strategy:

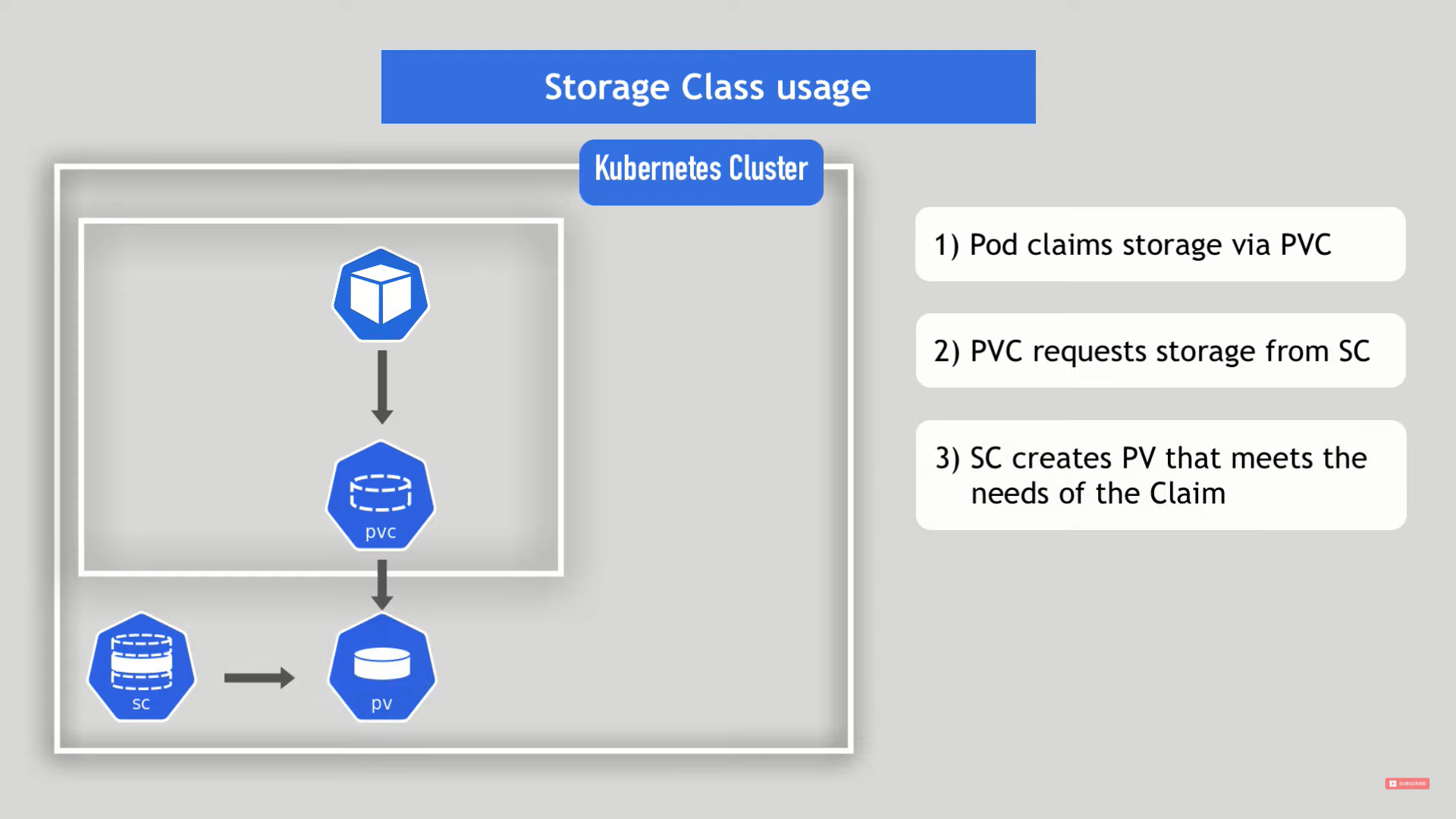
type: RollingUpdate

maxUnavailable=1

maxSurge=0

kubectl rollout status deploy <deploy-name>

**Volumes:**



storage

Dynamic provisioning of persistent volume: using storage class we can create the pv with the required resources.

Static provisioning of persistent volume: administrator has to create the pv and k8s will select the appropriate pv for pvc based on the required resource.

volumeBindingMode: Immediate (bind the pvc with pv even there was no pod)

volumeBindingMode: WaitForFirstConsumer (bind the pvc with pv, when pvc is attached with the pod)

**HELM:**

<https://www.youtube.com/watch?v=x77NzZxj670&list=PLSwo-wAGP1b8svO5fbAr7ko2Buz6GuH1g>

<https://github.com/DeekshithSN/Helm_charts>

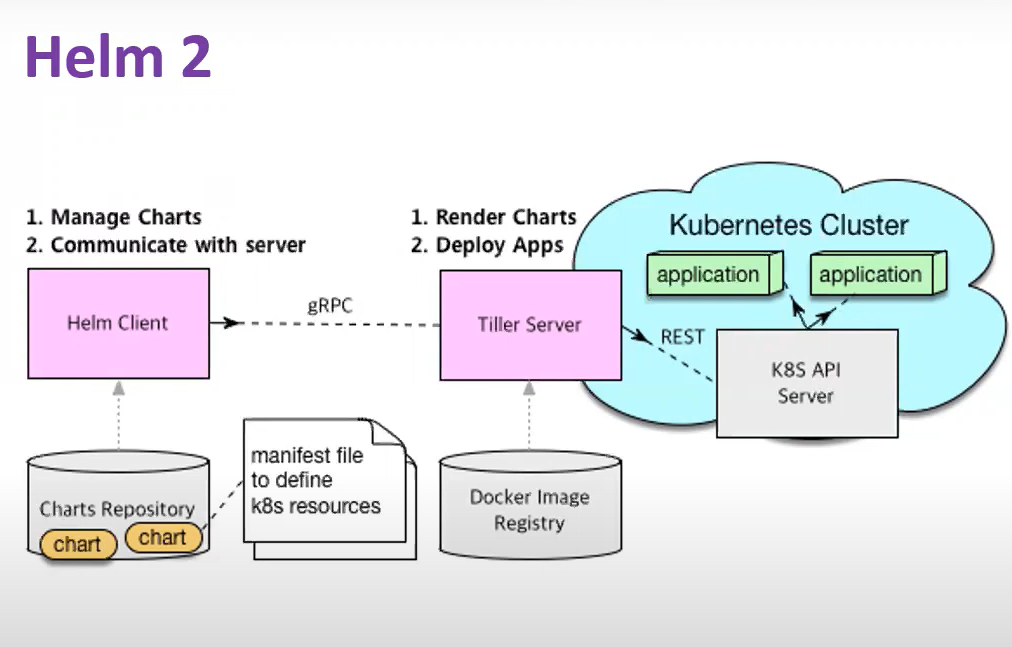
* packaging manager
* templating engine

helm charts - > group of k8s yaml files

we can download standard yaml config files with their name ex: mysql

chart – like class

release – like object



A diagram of a diagram

Description automatically generated with low confidence

* helm template <chart-name> (shows the filled values in the template files)
* helm lint <chart-name> (shows the errors-kind of compliation)
* helm install <release-name> <chart-name>
* helm list -a
* helm upgrade <release-name> <chart-name>
* helm rollback <release-name> <version>

flow control:

if:

{{if eq .Values. “name”}}

Key:true

{{end}}

With:

{{- with .Values.}} scope {{end}}

Looping:

{{- range .values.}}

{{end}}

{{- range $index, $val := .Values.favorite}}