|  |  |  |
| --- | --- | --- |
| Replication Controller |  | |
| Replication Sets |  | |
| Deployment Sets | Rather than doing rolling updates, if youi want to deploy new version of App, Deployment sets is the ideal option | |
| Daemon Sets | A DaemonSet makes sure it creates as many pods as there are nodes and deploys  each one on each node.  a DaemonSet doesn’t have any notion of a desired replica count.  that nodes can be made unschedulable,  preventing pods from being deployed to them.A DaemonSet will deploy pods  even to such nodes, because the unschedulable attribute is only used by the  Scheduler, whereas pods managed by a DaemonSet bypass the Scheduler  completely. | |
| Job |  | |
| Cronjob |  | |
| Stateful Sets | Stateful sets are similar to replica sets, but underlying pod has their own identity. Each Pod would have its own persistent Volume claim and the PVC would be used for another POD is there is POD failure and deleted manually. To crate Stateful sets we need following things.   PersistentVolumes for storing your data files (you’ll need to create these only if  the cluster doesn’t support dynamic provisioning of PersistentVolumes).   A governing Service required by the StatefulSet.   The StatefulSet itself | |
| Services | A Kubernetes Service is a resource you create to make a single, constant point of  entry to a group of pods providing the same service.  Following sessionAffinity flag would always allow requests from same client id to sent to same POD  apiVersion: v1  kind: Service  spec:  sessionAffinity: ClientIP  Service Endpoint: POD IP address.  To create a service with manually managed endpoints, you need to create both a  Service and an Endpoints resource.   1. DNS :<Service Name>. default.svc.cluster.local 2. <Service Name>.default 3. <Service Name> | |
| External Services | Node Port  Load Balancer  Ingress | |
| Readiness Probes | **An Exec probe**, where a process is executed. The container’s status is determined  by the process’ exit status code.   **An HTTP GET probe**, which sends an HTTP GET request to the container and  the HTTP status code of the response determines whether the container is  ready or not.   **A TCP Socket probe**, which opens a TCP connection to a specified port of the  container. If the connection is established, the container is considered ready. An Exec probe, where a process is executed. The container’s status is determined  by the process’ exit status code. | |
| Readiness vs Liveness | Unlike liveness probes, if a container fails the readiness check, it won’t be killed or  restarted.  Liveness probes keep pods healthy by killing off unhealthy containers and replacing  them with new, healthy ones, whereas readiness probes make sure that only pods that  are ready to serve requests receive them. | |
| Headless Service | Setting the clusterIP field in a service spec to None makes the service headless, as  Kubernetes won’t assign it a cluster IP through which clients could connect to the  pods backing it  Headless services, DNS returns the pods’ IPs, clients connect directly to the pods, instead of through the service proxy.  Headless Service:  Name: nodeapp-headless.default.svc.cluster.local  Address: 10.1.0.216  Name: nodeapp-headless.default.svc.cluster.local  Address: 10.1.0.217  Name: nodeapp-headless.default.svc.cluster.local  Address: 10.1.0.215  Normal Service:  Name: nodeappservice.default.svc.cluster.local  Address: 10.104.236.113 | |
| Volumes | emptyDir(General Pod volumes)  hostPath (Node Level)  gitRepo  gcePersistentDisk, azureDisk, awsElastic-BlockStore  persistentVolumeClaim  nfs  configMap, secret, downwardAPI | |
| PersistentVolume and PersistentVolumeClaims | PersistentVolume resources are cluster-scoped  and thus cannot be created in a specific namespace, but PersistentVolumeClaims can  only be created in a specific namespace. They can then only be used by pods in the  same namespace | |
| StorageClass | In the case of persistent Volume Storage needs to be created manually. If you want your cloud provider to do it, then use StorageClass.  The StorageClass resource specifies which provisioner should be used for provisioning  the PersistentVolume when a PersistentVolumeClaim requests this StorageClass. | |
| Dynamic provisioning without specifying a storage class | You can create a PVC without specifying the storageClassName attribute and (on  Google Kubernetes Engine) a GCE Persistent Disk of type pd-standard will be provisioned  for you.  Explicitly set storageClassName to "" if you want the PVC to use a preprovisioned  PersistentVolume | |
| ConfigMaps | Config map is where you can store your configuration.  Config Map can be loaded as environment variables or Files in a Volume. | |
| Secrets | Secrets are similar to config Map, they are not stored are files, rather they are stored in Memory in basse64 encoded format. Secrets can be loaded as environment variables or Files in a Volume. | |
| Always Pull Images without using cached Images | To make sure this doesn’t happen, you need to set the container’s  imagePullPolicy property to Always | |
| Rolling Update | kubectl rolling-update kubia-v1 kubia-v2 --image=luksa/kubia:v2 | |
| deployment strategy | RollingUpdate  Recreate | |
| Graphical user interface, text, application  Description automatically generated | | |
| Deployment Set | | the length of  the revision history is limited by the revisionHistoryLimit property on the Deployment  resource. |
| Deployment Set | | The time after which the Deployment is considered failed is configurable through the  progressDeadlineSeconds property in the Deployment spec. |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Control Pane | While multiple instances of etcd and API server can be active at the same time and do perform their jobs in parallel, only a single instance of the Scheduler  and the Controller Manager may be active at a given time |

Diagram

Description automatically generated

|  |  |
| --- | --- |
|  | This pause container is the container that holds all the containers of a pod  Together.The pause container is an infrastructure container whose sole  purpose is to hold all these namespaces. All other user-defined containers of the pod  then use the namespaces of the pod infrastructure container |