**Creating a Deployment**

A Deployment is a higher level abstraction which sits on top of replica sets and allows you to manage the way applications are deployed, rolled back at a controlled rate.

Deployment has mainly two responsibilities,

* Provide Fault Tolerance: Maintain the number of replicas for a type of service/app. Schedule/delete pods to meet the desired count.
* Update Strategy: Define a release strategy and update the pods accordingly.

1. /k8s-code/projects/instavote/dev/
2. cp vote-rs.yaml vote-deploy.yaml

Deployment spec (deployment.spec) contains everything that replica set has + strategy. Lets add it as follows,

File: vote-deploy.yaml

1. apiVersion: apps/v1
2. kind: Deployment
3. metadata:
4. name: vote
5. spec:
6. strategy:
7. type: RollingUpdate
8. rollingUpdate:
9. maxSurge: 2
10. maxUnavailable: 1
11. revisionHistoryLimit: 4
12. paused: false
13. replicas: 5
14. minReadySeconds: 20
15. selector:
16. matchLabels:
17. role: vote
18. matchExpressions:
19. - {key: version, operator: In, values: [v1, v2, v3]}
20. template:
21. metadata:
22. name: vote
23. labels:
24. app: python
25. role: vote
26. version: v1
27. spec:
28. containers:
29. - name: app
30. image: schoolofdevops/vote:v1
31. ports:
32. - containerPort: 80
33. protocol: TCP

This time, start monitoring with --show-labels options added.

1. watch -n 1 kubectl get pod,deploy,rs,svc --show-labels

Lets create the Deployment. Do monitor the labels of the pod while applying this.

kubectl apply -f vote-deploy.yaml

Observe the chances to pod labels, specifically the **pod-template-hash**.

Now that the deployment is created. To validate,

1. kubectl get deployment
2. kubectl get rs --show-labels
3. kubectl get deploy,pods,rs
4. kubectl rollout status deployment/vote
5. kubectl get pods --show-labels

Sample Output

1. kubectl get deployments
2. NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
3. vote 3 3 3 1 3m