Deployments

ow

Explain

- Deployment
- Api
- Replica set
- Command line options
- Update strategies
- Rollback

Deployments

- A deployment provides declarative update for pods and replica set.
- In deployment, describe desired state. Deployment controller changes the actual state to the desired state
- It maintains the replica set.

example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

..

- : The .spec.selector.matchLabels field is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". All of the requirements, from both matchLabels and matchExpressions, must be satisfied in order to match.

```
selector:
  matchLabels:
    tier: frontend
  matchExpressions:
    - {key: name, operator: In, values: [payroll, web]}
    - {key: environment, operator: NotIn, values: [dev]}
```

..

- In—Label’s value must match one of the specified values.
- NotIn—Label’s value must not match any of the specified values.

Try Lab

- Yaml and commands
  - kubectl apply -f https://k8s.io/examples/controllers/nginx-deployment.yaml
  - kubectl get deloyment
  - kubectl rollout status deployment/nginx-deployment
  - kubectl get pods --show-labels

Updating a deployment

- Deployment rollout triggers, if .spec.template changes such as
  - Labels change
  - Image change
- Image change can be done using
  - Kubectl command
- Check rollout status
- Try commands from the doc

Rollover

- If the Deployment is updated, the existing ReplicaSet that controls Pods whose labels match .spec.selector but whose template does not match .spec.template are scaled down. Eventually, the new ReplicaSet is scaled to .spec.replicas and all old ReplicaSets is scaled to 0.
- suppose you create a Deployment to create 5 replicas of nginx:1.14.2, but then update the Deployment to create 5 replicas of nginx:1.16.1, when only 3 replicas of nginx:1.14.2 had been created. In that case, the Deployment immediately starts killing the 3 nginx:1.14.2 Pods that it had created, and starts creating nginx:1.16.1 Pods. It does not wait for the 5 replicas of nginx:1.14.2 to be created before changing course

Label selector update

- It is generally discouraged to make label selector updates and it is suggested to plan your selectors up front. In any case, if you need to perform a label selector update, exercise great caution and make sure you have grasped all of the implications
- Selector additions require the Pod template labels in the Deployment spec to be updated with the new label too, otherwise a validation error is returned
- Selector removals removes an existing key from the Deployment selector -- do not require any changes in the Pod template labels. Existing ReplicaSets are not orphaned, and a new ReplicaSet is not created, but note that the removed label still exists in any existing Pods and ReplicaSets.

Rolling Back a Deployment

- Sometimes, you may want to rollback a Deployment; for example, when the Deployment is not stable, such as crash looping. By default, all of the Deployment's rollout history is kept in the system so that you can rollback anytime you want (you can change that by modifying revision history limit).
- Lab 3 perform

Strategy

- .spec.strategy specifies the strategy used to replace old Pods by new ones. .spec.strategy.type can be "Recreate" or "RollingUpdate". "RollingUpdate" is the default value.
- Recreate Deployment
  - All existing Pods are killed before new ones are created when .spec.strategy.type==Recreate.
- Rolling Update Deployment
  - The Deployment updates Pods in a rolling update fashion when .spec.strategy.type==RollingUpdate. You can specify maxUnavailable and maxSurge to control the rolling update process

Max Unavailable

- .spec.strategy.rollingUpdate.maxUnavailable is an optional field that specifies the maximum number of Pods that can be unavailable during the update process. The value can be an absolute number (for example, 5) or a percentage of desired Pods (for example, 10%). The absolute number is calculated from percentage by rounding down. The value cannot be 0 if .spec.strategy.rollingUpdate.maxSurge is 0. The default value is 25%.

- For example, when this value is set to 30%, the old ReplicaSet can be scaled down to 70% of desired Pods immediately when the rolling update starts. Once new Pods are ready, old ReplicaSet can be scaled down further, followed by scaling up the new ReplicaSet, ensuring that the total number of Pods available at all times during the update is at least 70% of the desired Pods.

Max Surge

- .spec.strategy.rollingUpdate.maxSurge is an optional field that specifies the maximum number of Pods that can be created over the desired number of Pods. The value can be an absolute number (for example, 5) or a percentage of desired Pods (for example, 10%). The value cannot be 0 if MaxUnavailable is 0. The absolute number is calculated from the percentage by rounding up. The default value is 25%.

- For example, when this value is set to 30%, the new ReplicaSet can be scaled up immediately when the rolling update starts, such that the total number of old and new Pods does not exceed 130% of desired Pods. Once old Pods have been killed, the new ReplicaSet can be scaled up further, ensuring that the total number of Pods running at any time during the update is at most 130% of desired Pods