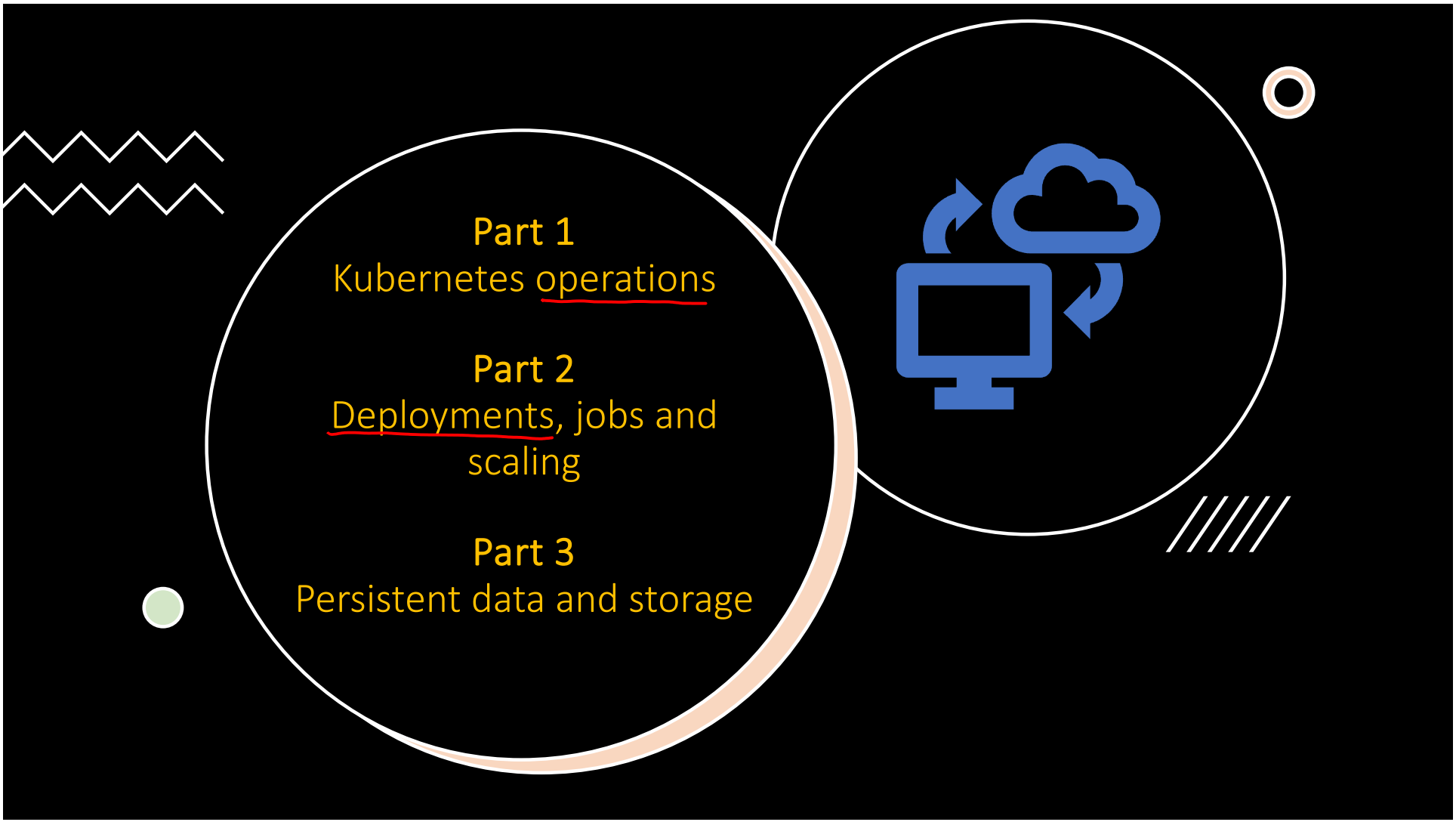
CSCI 5409 Cloud Computing
Fall, 2023
Instructor: Dr. Lu Yang

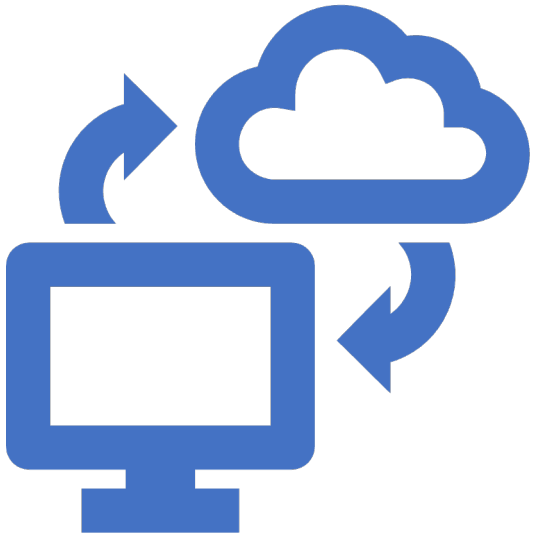
Kubernetes Workload & Production (1)

Oct 13, 2023

Housekeeping and Feedback

- Start recording
- Start playing with GKE
- Ask questions in the Teams channel


Part 1
Kubernetes operations
Part 2
Deployments, jobs and scaling
Part 3
Persistent data and storage



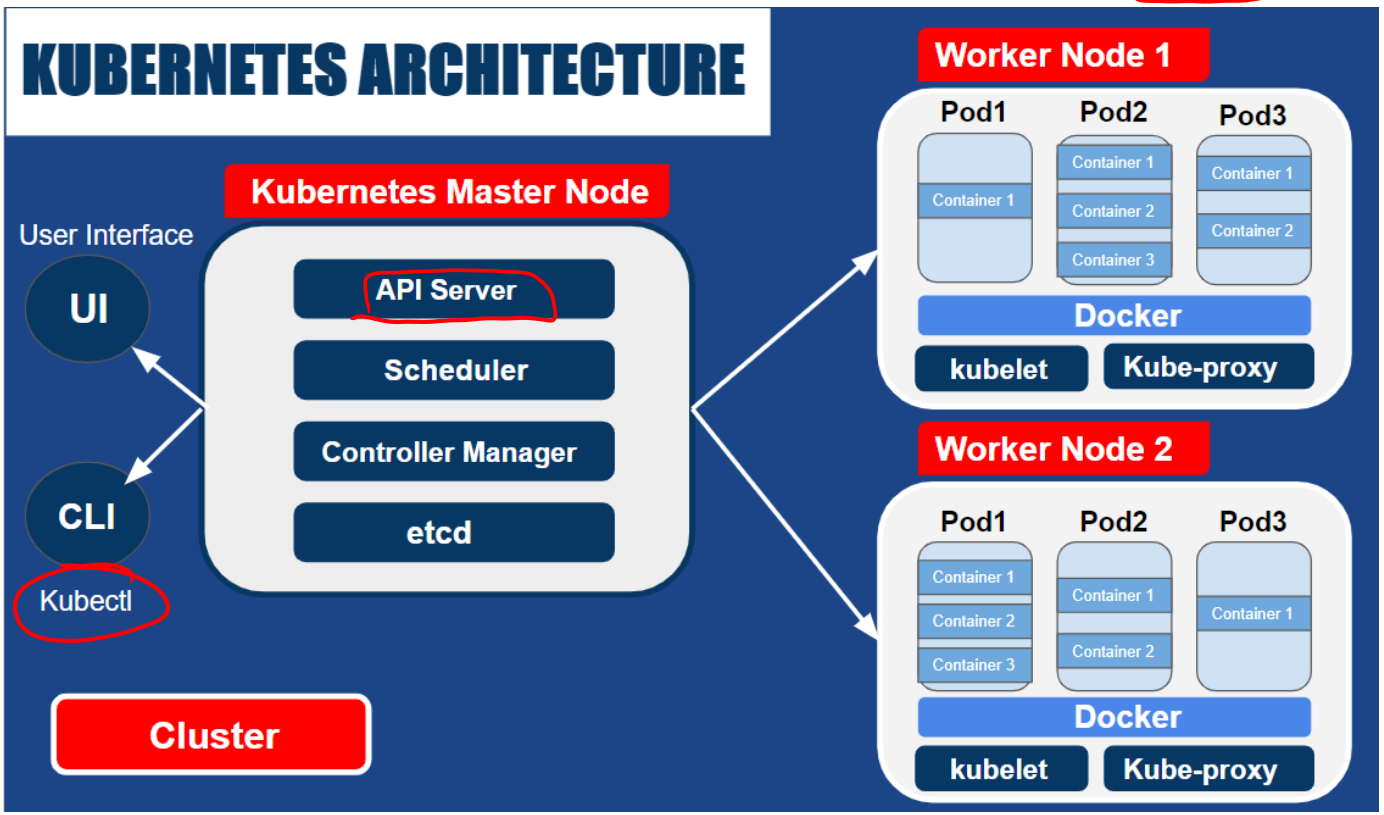
Part 1 Kubernetes operation

- The kubectl command
- Introspection
- Summary

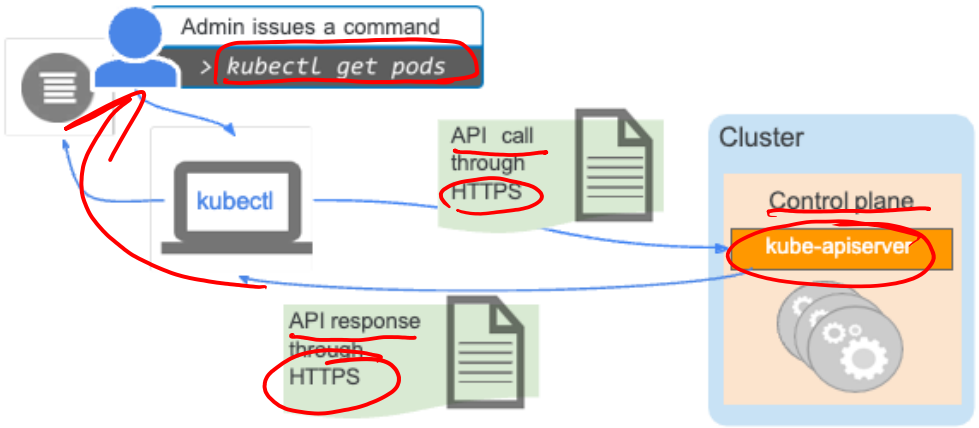
Part 1 Kubernetes operation

The kubectl command

Kubectl transforms your command-line entries into API calls



Use kubectl to see a list of Pods in a cluster



https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

kubectl must be configured first

- Relies on a config file: $HOME/.kube/config.
- Config file contains:
  - Target cluster name
  - Credentials for the cluster
- Current config: kubectl config view.

cloud shell

Connecting to a Google Kubernetes Engine cluster

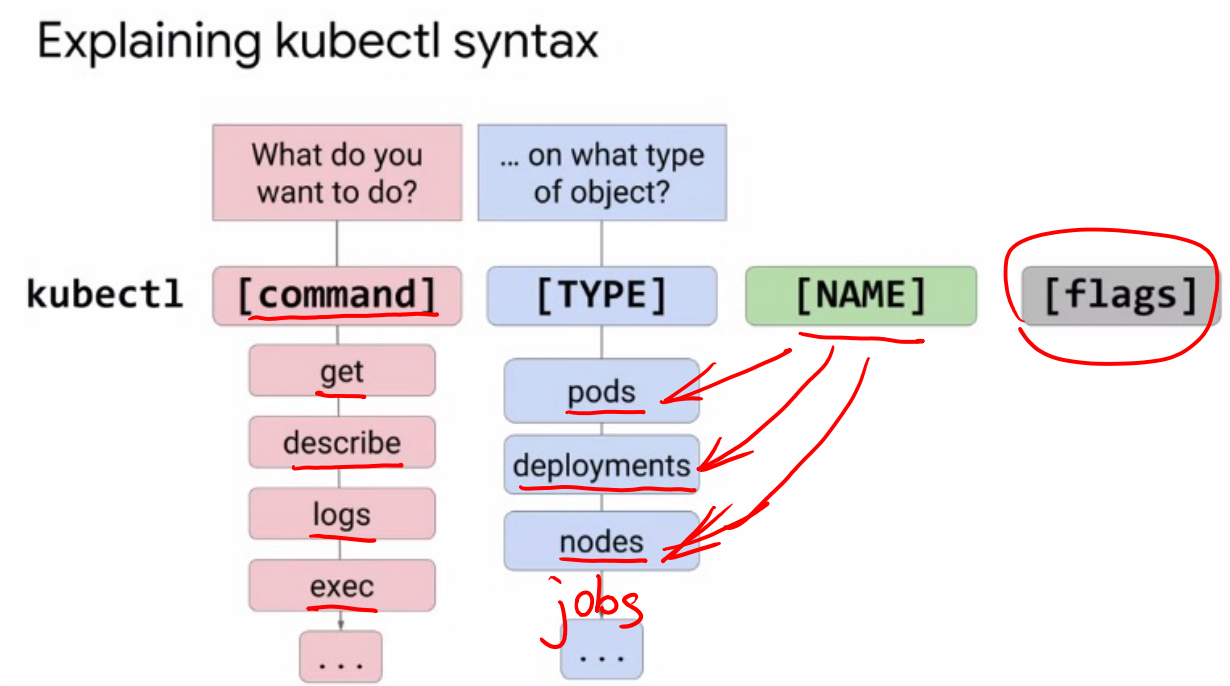
```
$ gcloud container clusters \
  get-credentials [CLUSTER_NAME] \
  --zone [ZONE_NAME]
```

gcloud vs. kubectl

internal

Part 1 Kubernetes operation

The kubectl command

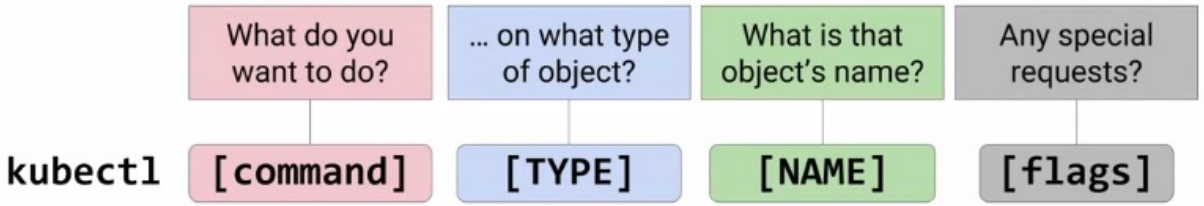


https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Part 1 Kubernetes operation

The kubectl command

Explaining kubectl syntax



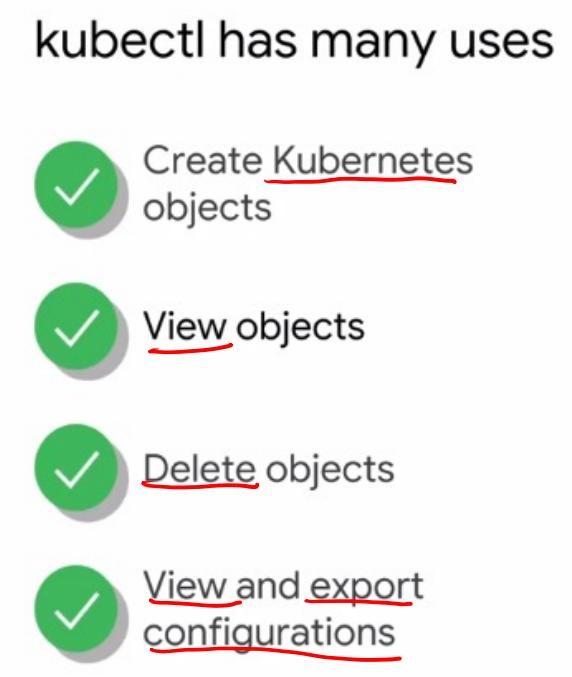

```
kubectl get pod my-test-app -o=yaml
```

```
kubectl get pods -o=wide
```

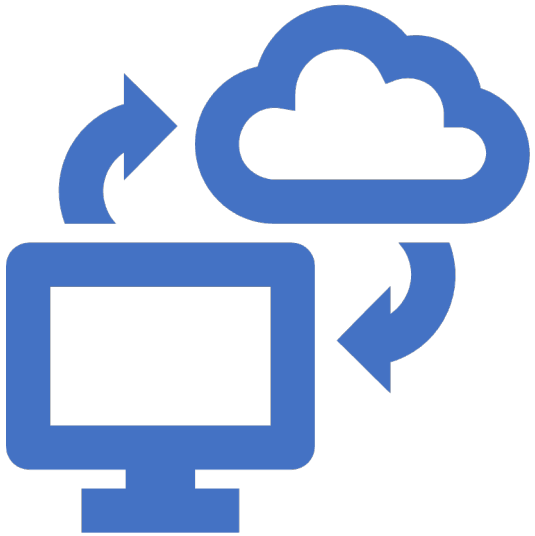
https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Part 1 Kubernetes operation

The kubectl command



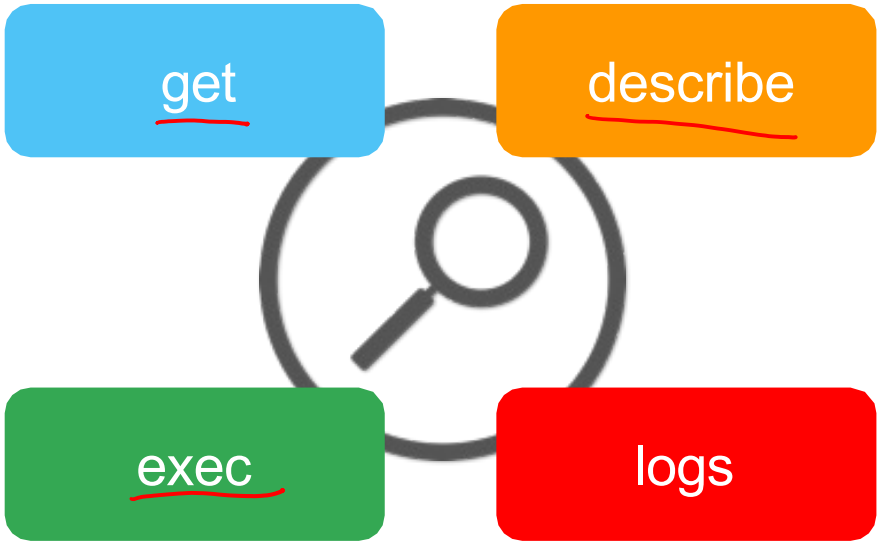
https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/



Part 1 Kubernetes operation

- The kubectl command
- Introspection
- Summary

Use kubectl to gather info about your app



https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Part 1 Kubernetes operation

Introspection

The listing: "get"

get | describe | exec | logs

Getting a list of Pods objects

```
$ kubectl get pods
```

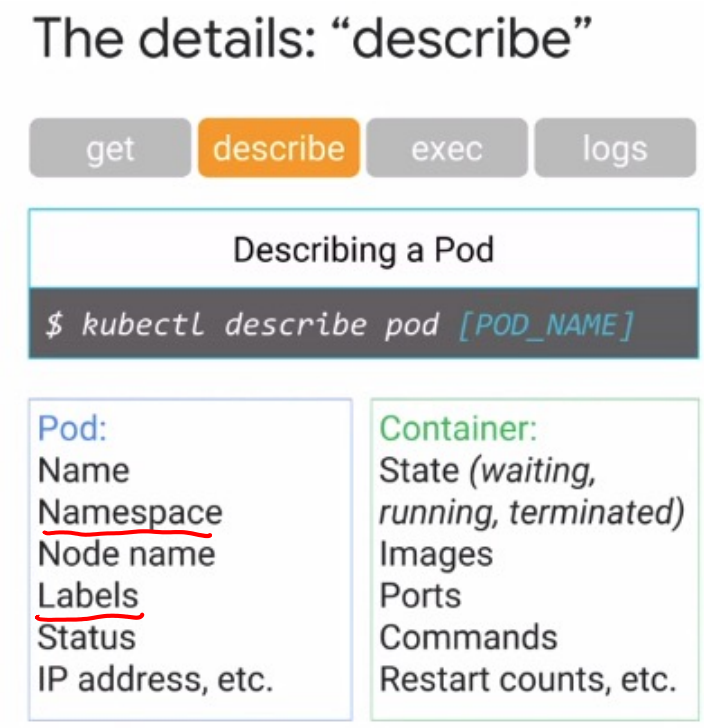
Pod phases:

- Pending
- Running
- Succeeded
- Failed
- Unknown
- CrashLoopBackOff code

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Part 1 Kubernetes operation

Introspection



https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Part 1 Kubernetes operation

Introspection

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Working inside a Pod

Running a command within a Pod

```
$ kubectl exec -it [POD_NAME] -- [command]
```

```
$ kubectl exec -it demo -- /bin/bash
root@demo:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys  tmp  usr  var
root@demo:/#
```

Pass terminal's standard input (stdin) to the container.

Display the container's standard output (stdout) in your terminal window.

Uses the flags -i and -t (or -it).

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Part 1 Kubernetes operation

Introspection

The history: “logs”

get | describe | exec | **logs**

Getting logs for a Pod

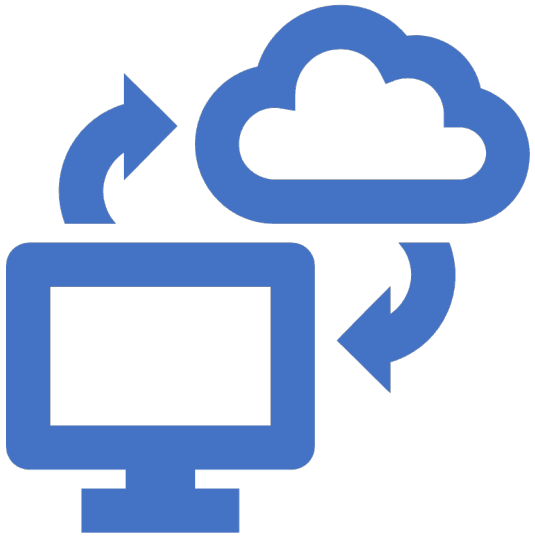
```
$ kubectl logs [POD_NAME]
```

The Pod logs include:

- `stdout`:
  Standard output on the console

- `stderr`:
  Error messages

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

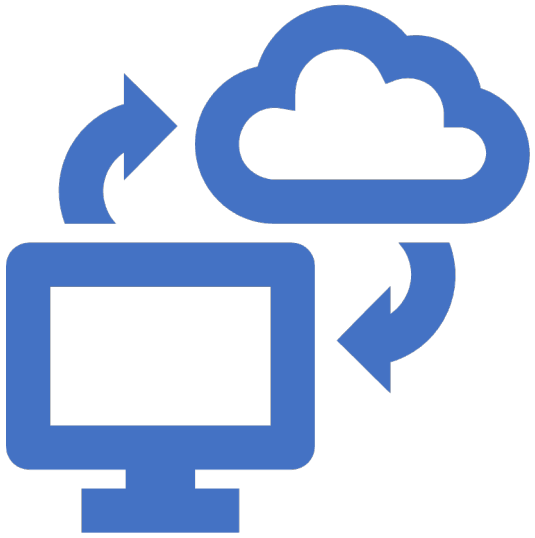


Part 1 Kubernetes operation

- The kubectl command
- Introspection
- Summary

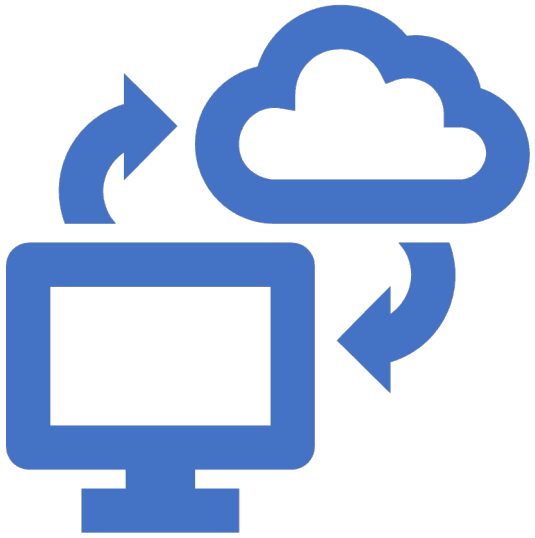
Summary

- kubectl is a utility used by administrators to control Kubernetes clusters.
- kubectl syntax is composed of several parts.
- kubectl has many uses.
- Use kubectl to gather info about your app.



Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- Summary



Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine
  Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools

Deployments declare the state of Pods

Roll out updates to the Pods

Roll back Pods to previous revision

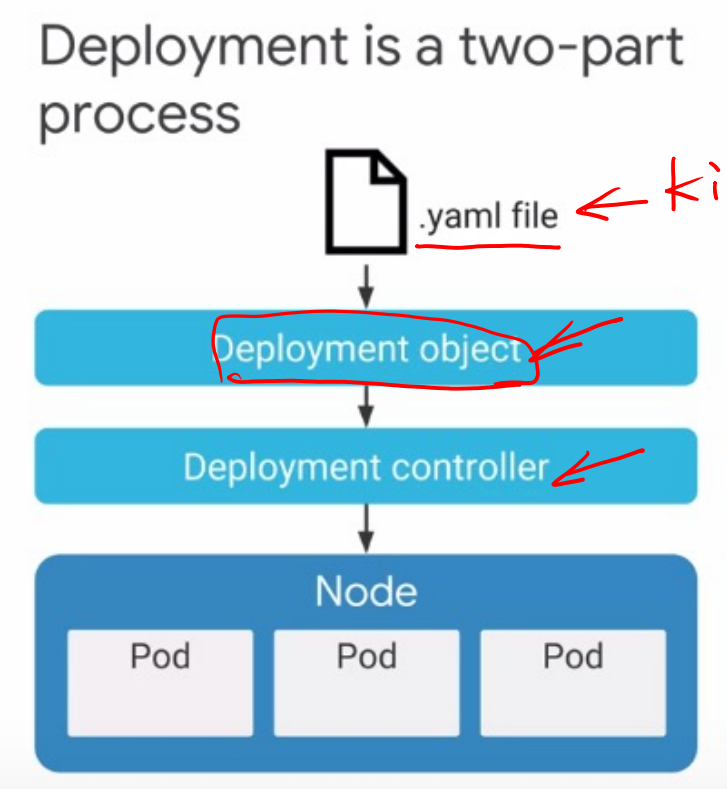
Scale or autoscale Pods

Well-suited for stateless applications

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Part 2 Deployments, jobs, and scaling

Deployments



https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Deployment object file in YAML format

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: gcr.io/demo/my-app:1.0
        ports:
        - containerPort: 8080
```

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Deployment object file in YAML format

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

There are three ways to create a Deployment

1

```
$ kubectl apply -f [DEPLOYMENT_FILE]
```

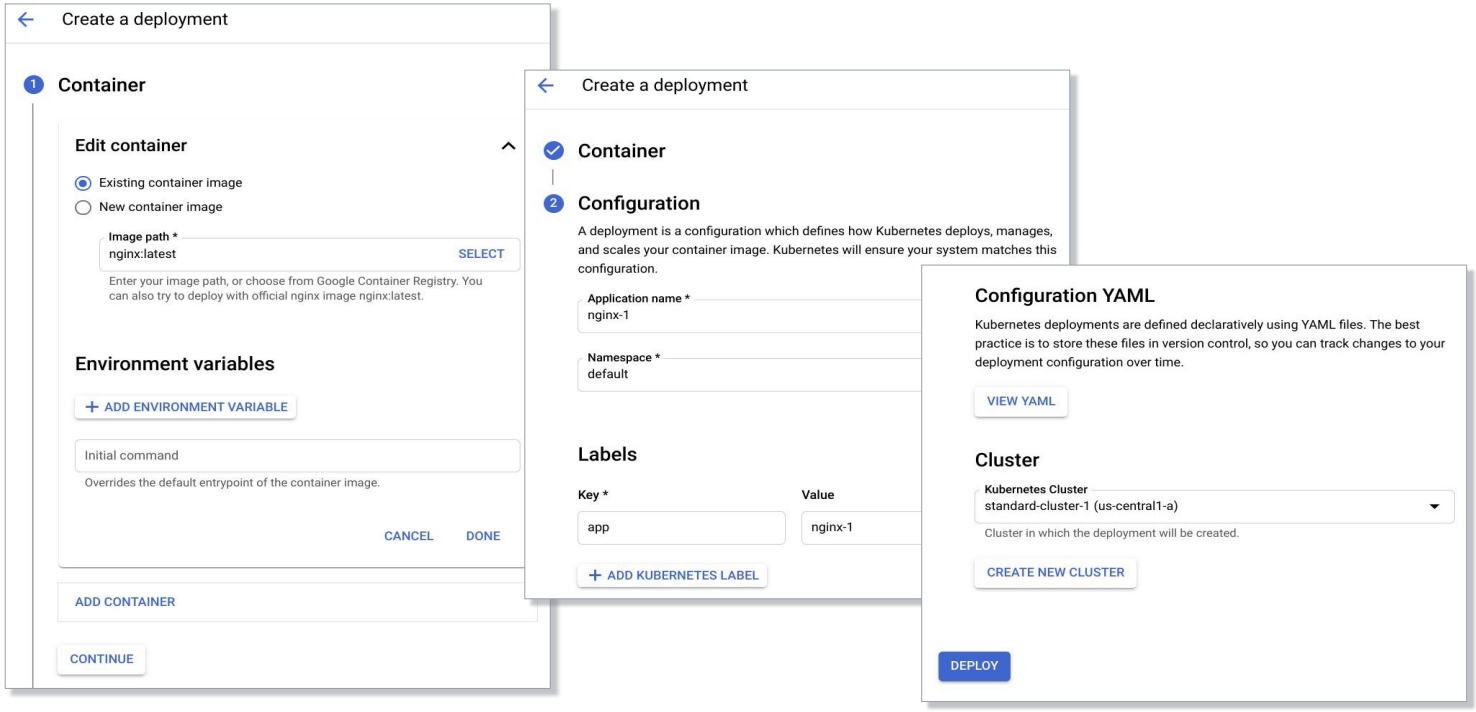
2

```
$ kubectl create deployment \
[DEPLOYMENT_NAME] \
  --image [IMAGE]:[TAG] \
  --replicas 3 \
  --labels [KEY]=[VALUE] \
  --port 8080 \
```

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

There are three ways to create a Deployment

3



https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

Use kubectl to inspect your Deployment, or output the Deployment config in a YAML format

```
$ kubectl get deployment [DEPLOYMENT_NAME]
```

```
master $ kubectl get deployment nginx-deployment
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3         3         3            3           3m
```

```
$ kubectl get deployment [DEPLOYMENT_NAME] -o yaml > this.yaml
```

describe
logs

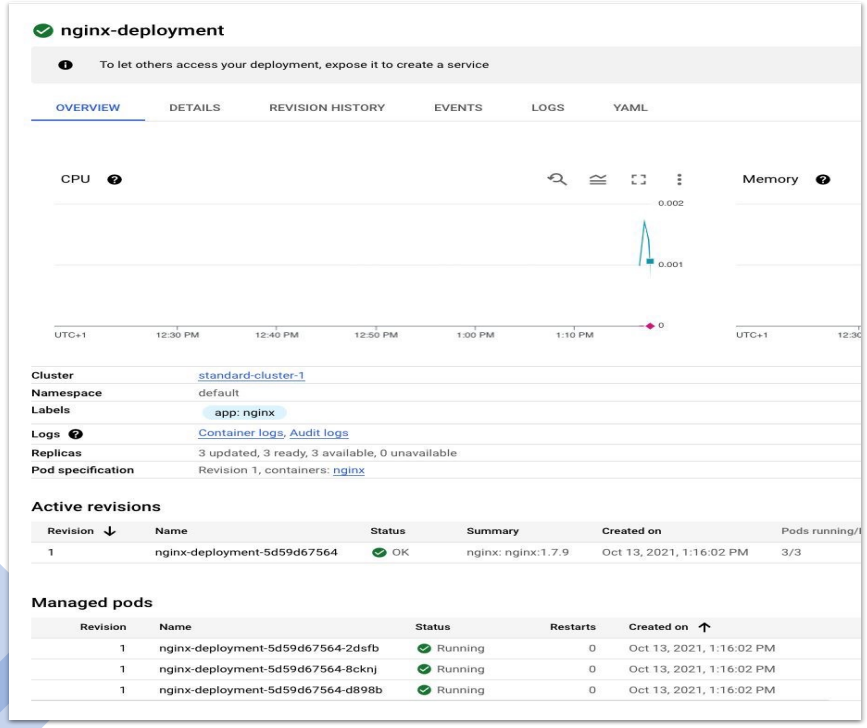
https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

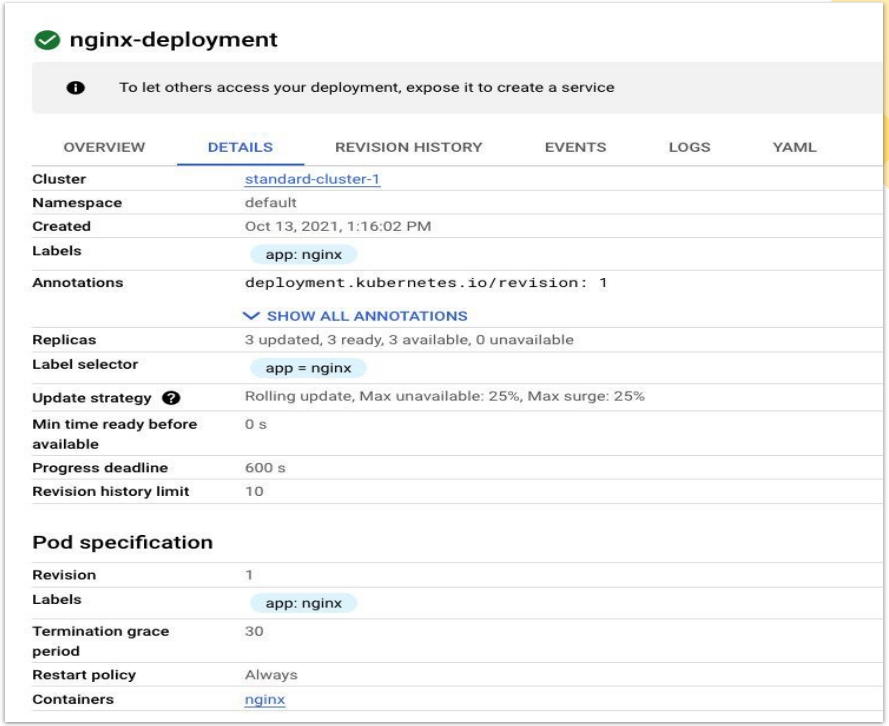
Use the ‘describe’ command to get detailed info

```
$ kubectl describe deployment [DEPLOYMENT_NAME]
```

```
master $ kubectl describe deployment nginx-deployment
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Fri, 12 Oct 2018 15:23:46 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision=1
Selector:               app=nginx
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.15.4
    Port:         80/TCP
    Host Port:    0/TCP
```

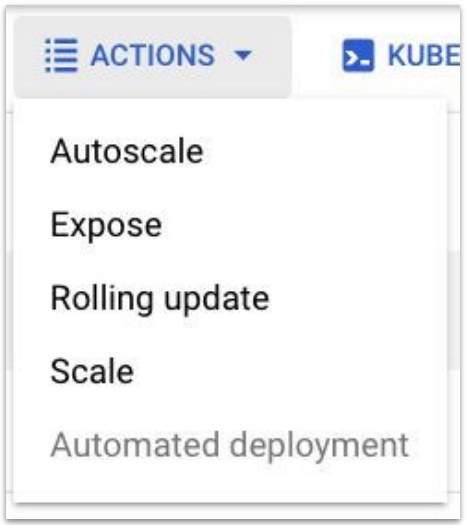
Or use the Cloud Console

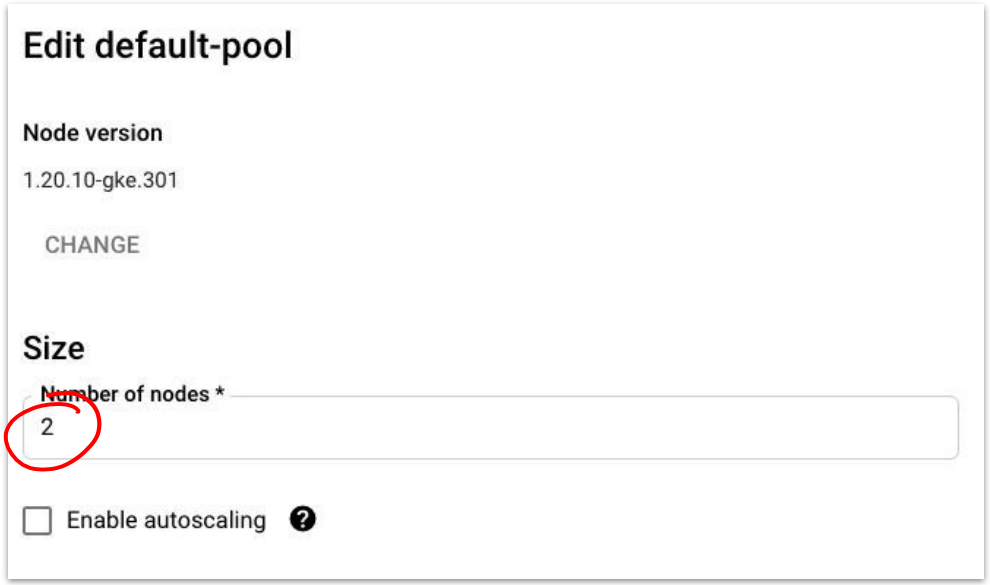




You can scale the Deployment manually

```
$ kubectl scale deployment [DEPLOYMENT_NAME] --replicas=5
```

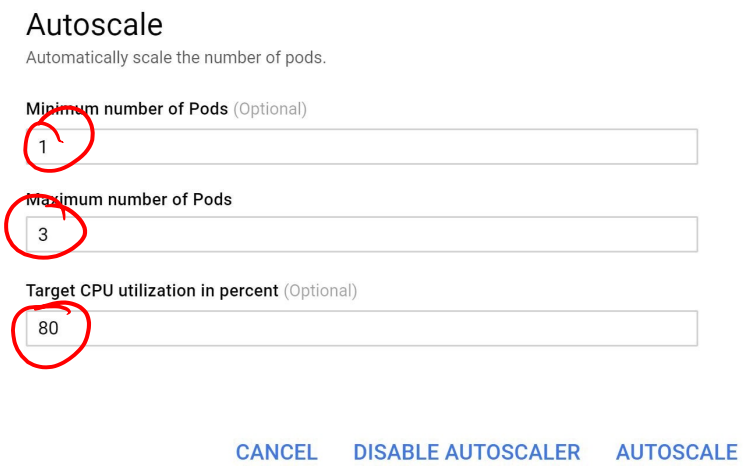




https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

You can scale the Deployment automatically

```
$ kubectl autoscale deployment [DEPLOYMENT_NAME] --min=5 --max=15
--cpu-percent=75
```



You can update a Deployment in different ways

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  template: spec:
      containers:
      - name: my-app
        image: gcr.io/demo/my-app:1.0
        ports:
        - containerPort: 8080
```

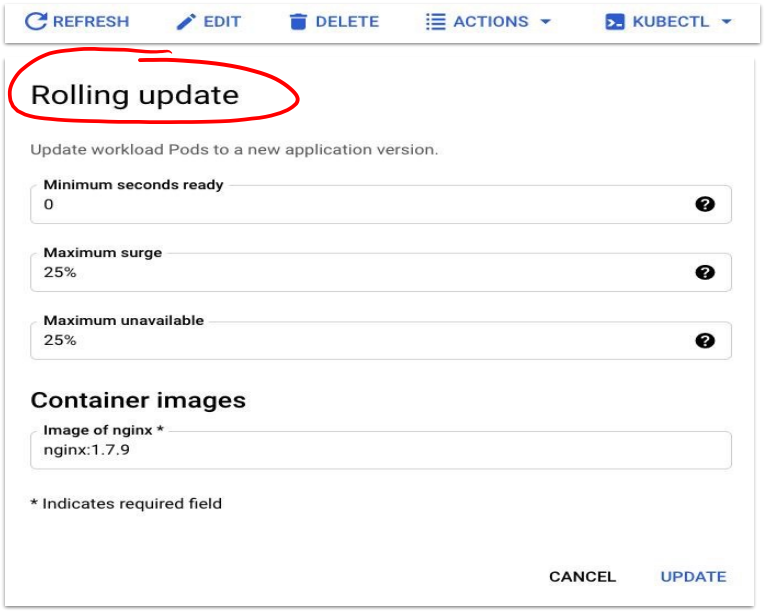
```
$ kubectl apply -f [DEPLOYMENT_FILE]
```

```
$ kubectl set image deployment [DEPLOYMENT_NAME] [IMAGE] [IMAGE]:[TAG]
```

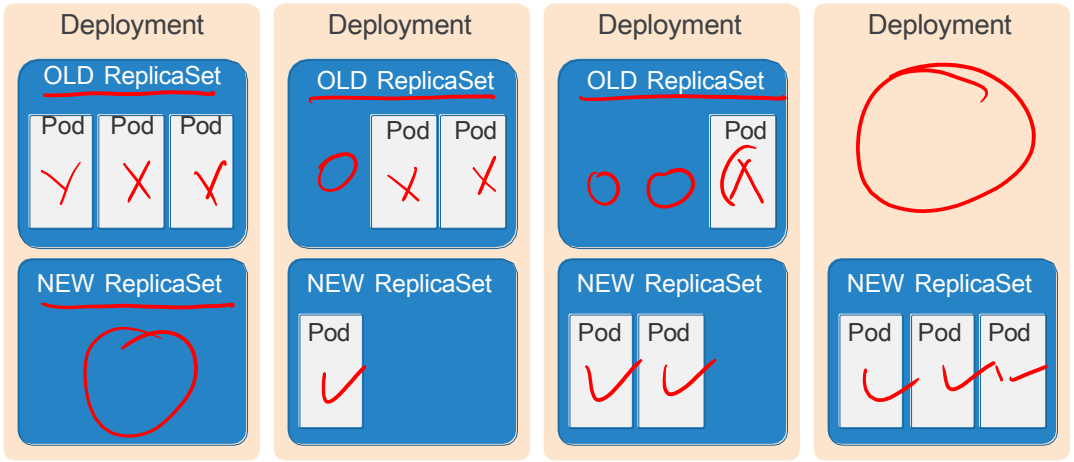
```
$ kubectl edit \ deployment/[DEPLOYMENT_NAME]
```

https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/

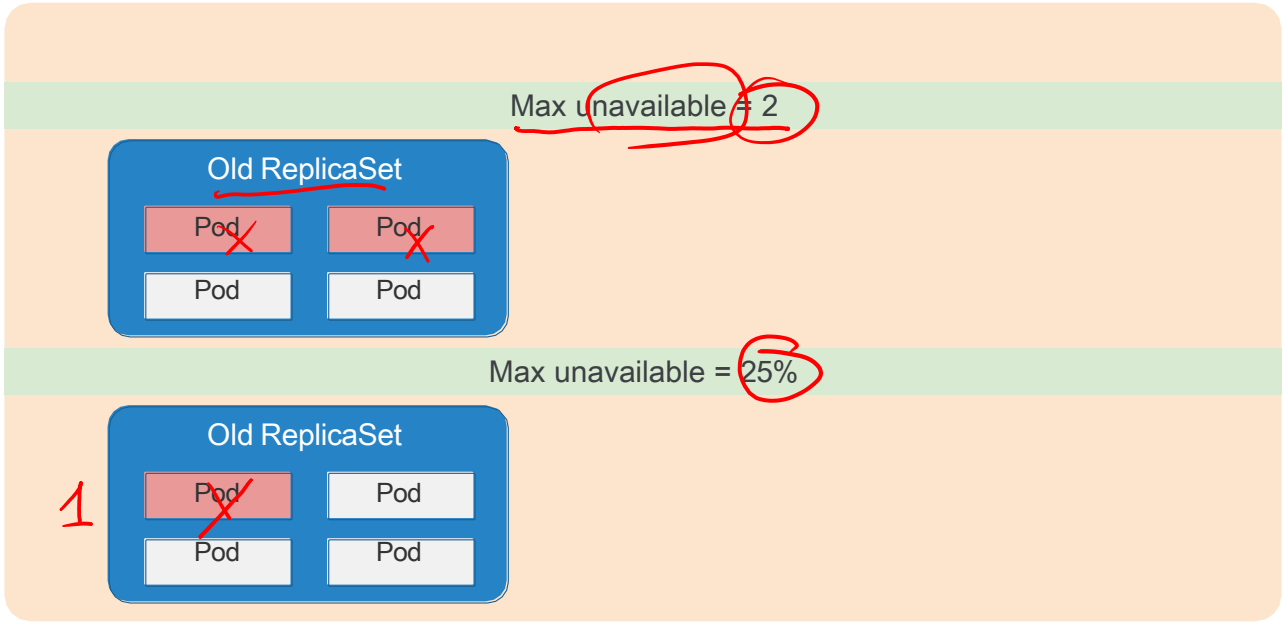
You can update a Deployment in different ways



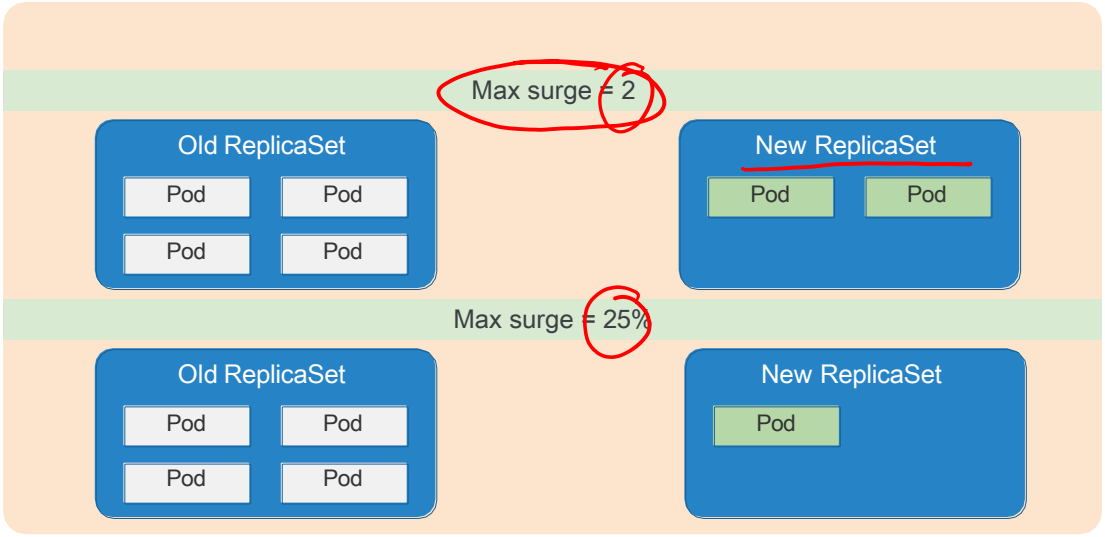
The process behind updating a Deployment



- Set parameters for your rolling update strategy



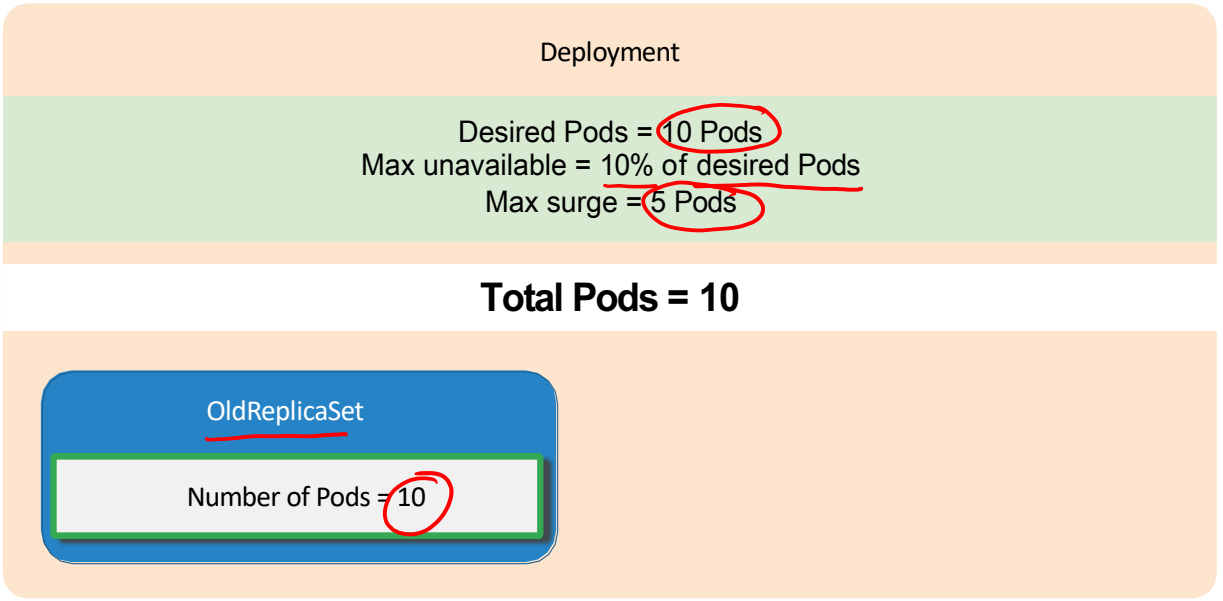
Set parameters for your rolling update strategy



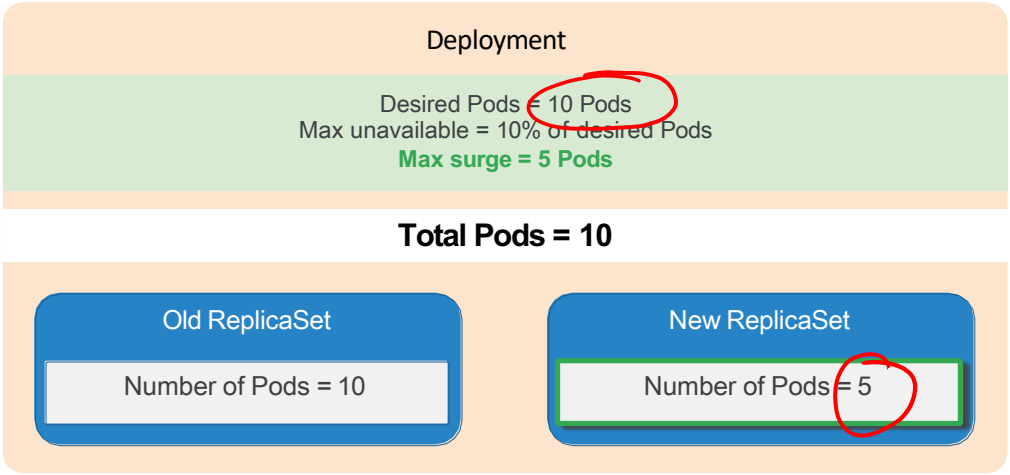
An example of a rolling update strategy

```
[…]
kind: deployment
spec:
  replicas: 10
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 5
      maxUnavailable: 10%
[…]
```

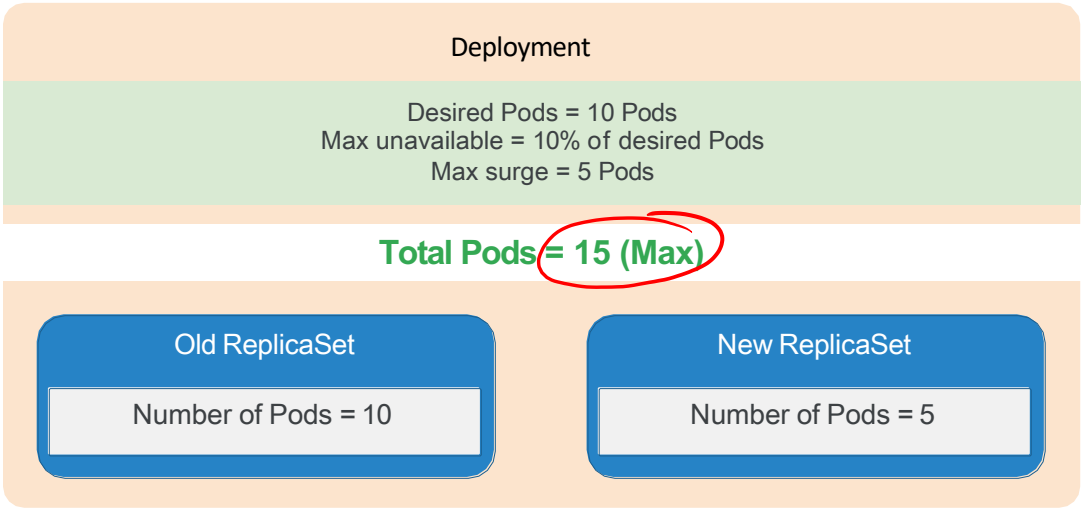
An example of a rolling update strategy



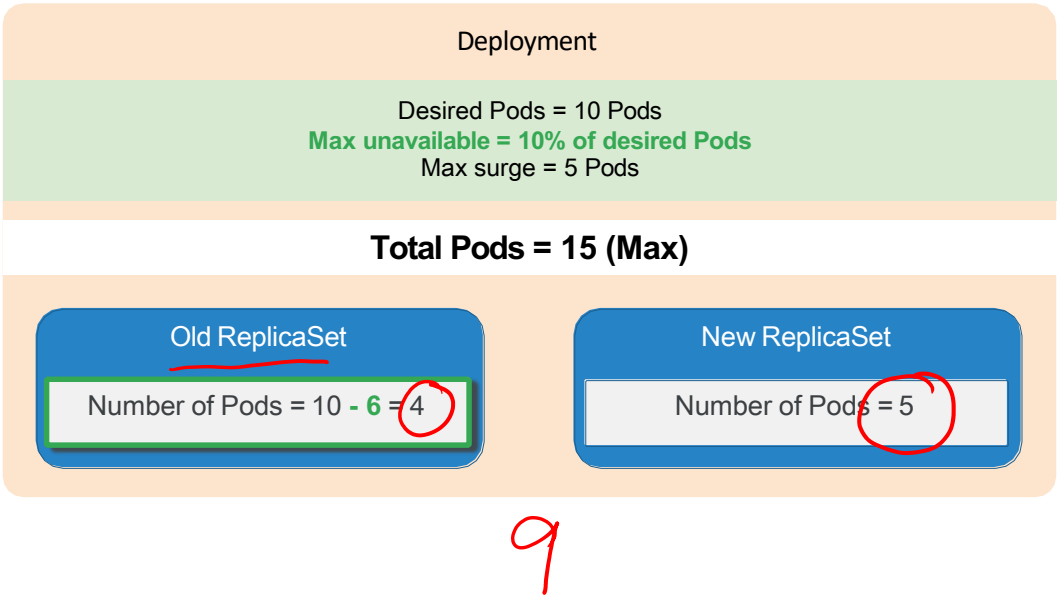
An example of a rolling update strategy



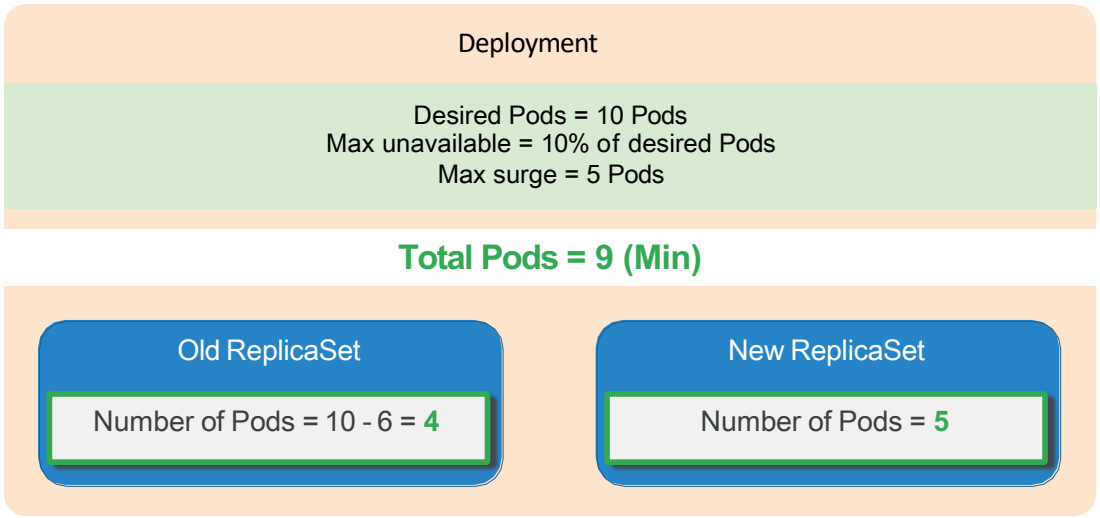
An example of a rolling update strategy



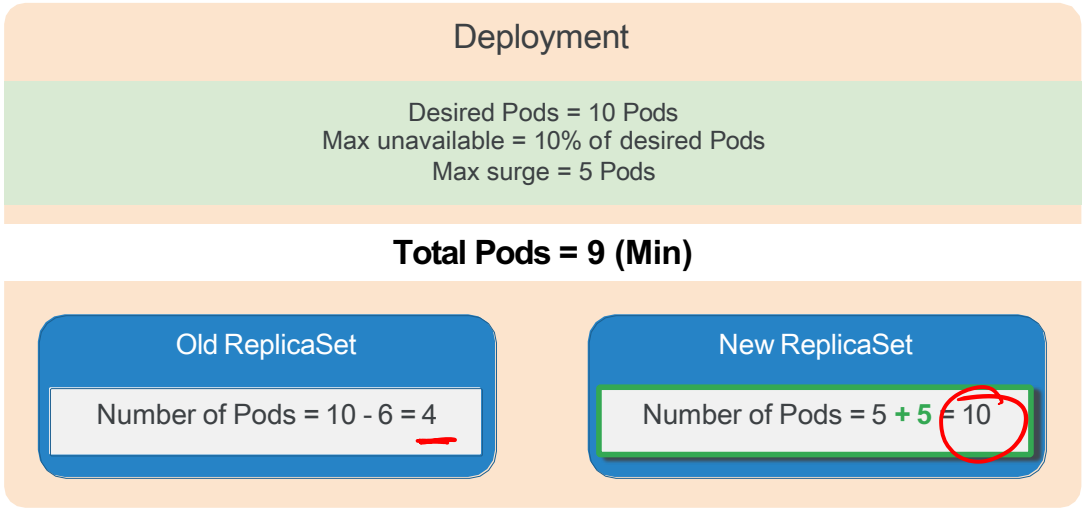
An example of a rolling update strategy



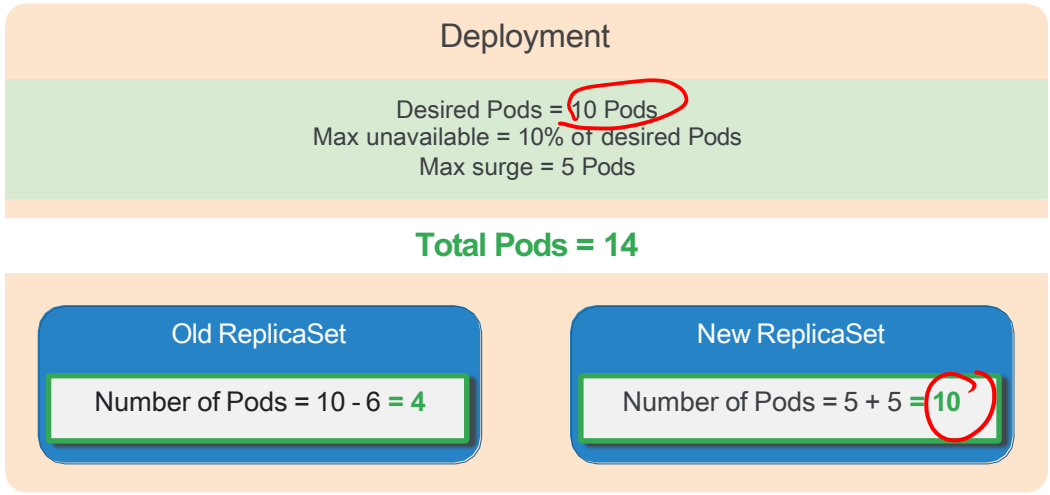
An example of a rolling update strategy



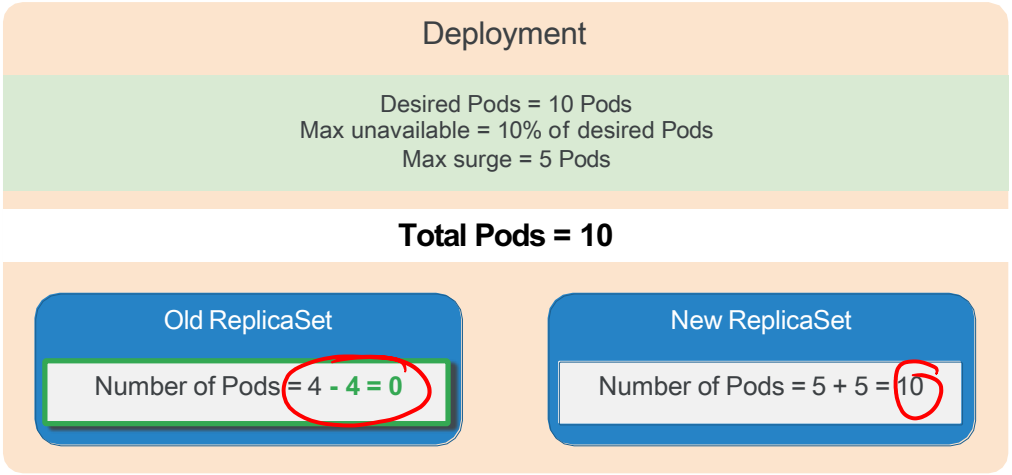
An example of a rolling update strategy



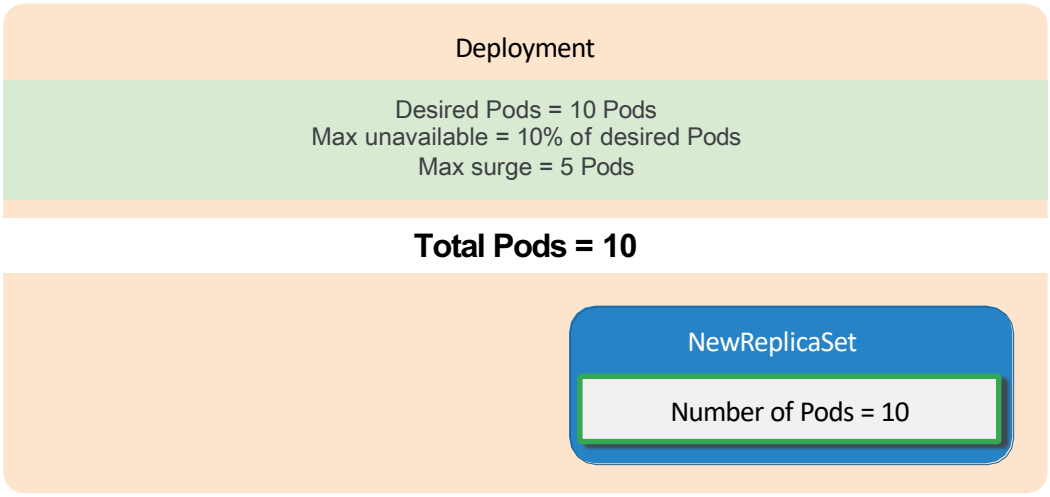
An example of a rolling update strategy



An example of a rolling update strategy



An example of a rolling update strategy



Applying a Recreate strategy

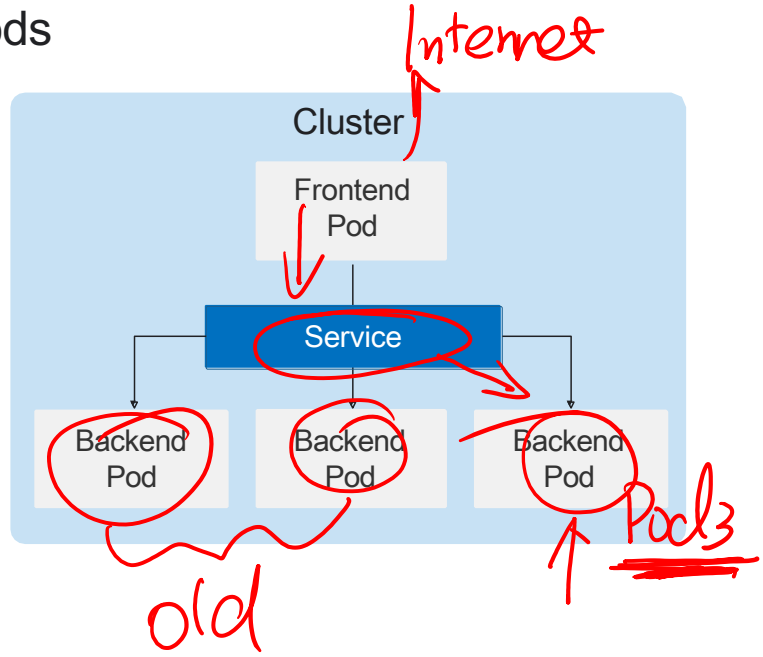
```
[…]
kind: deployment
spec:
    replicas: 10
    strategy:
        type: Recreate
[…]
```

Old removed first
downtime

Part 2 Deployments, jobs, and scaling

Deployments

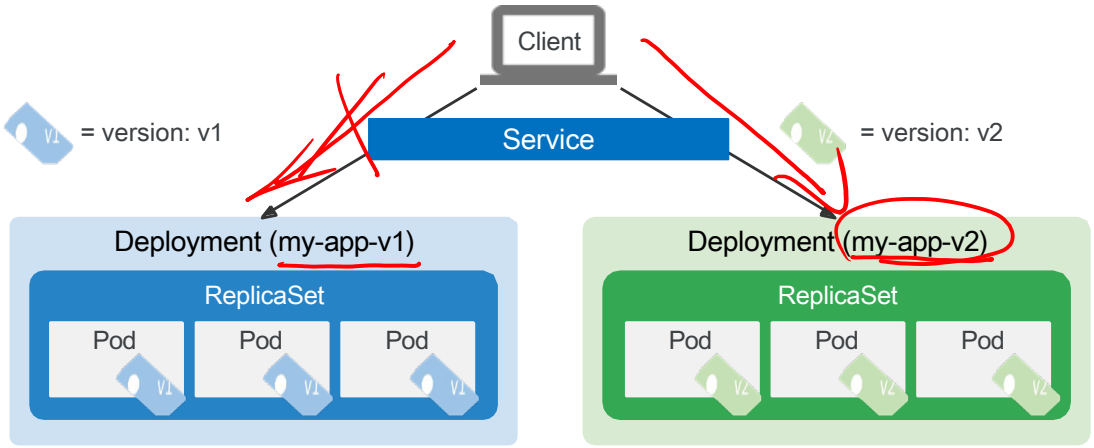
Service is a stable network representation of a set of pods



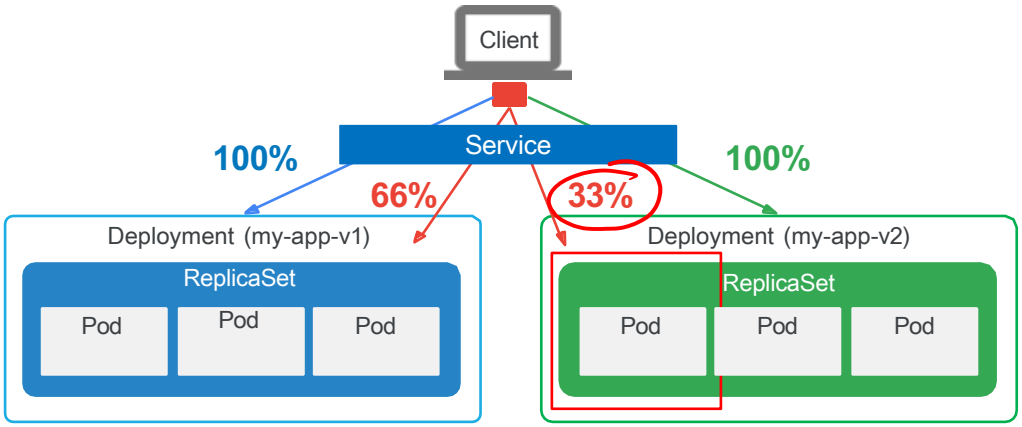
In Kubernetes a deployment is a method of launching a pod with containerized applications and ensuring that the necessary number of replicas is always running on the cluster.
On the other hand, a service is responsible for exposing an interface to those pods, which enables network access from either within the cluster or between external processes and the service.

ClusterIP
Node Port
Load Balancer

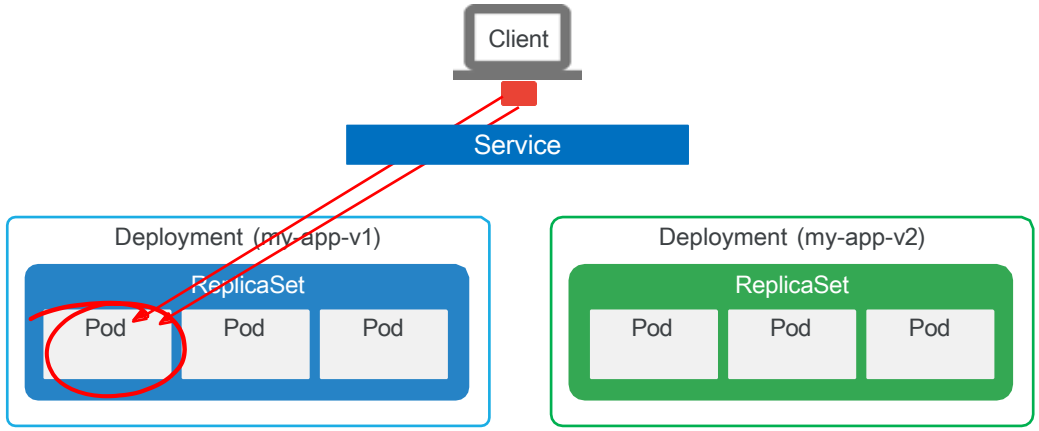
A blue/green deployment strategy ensures app services remain available



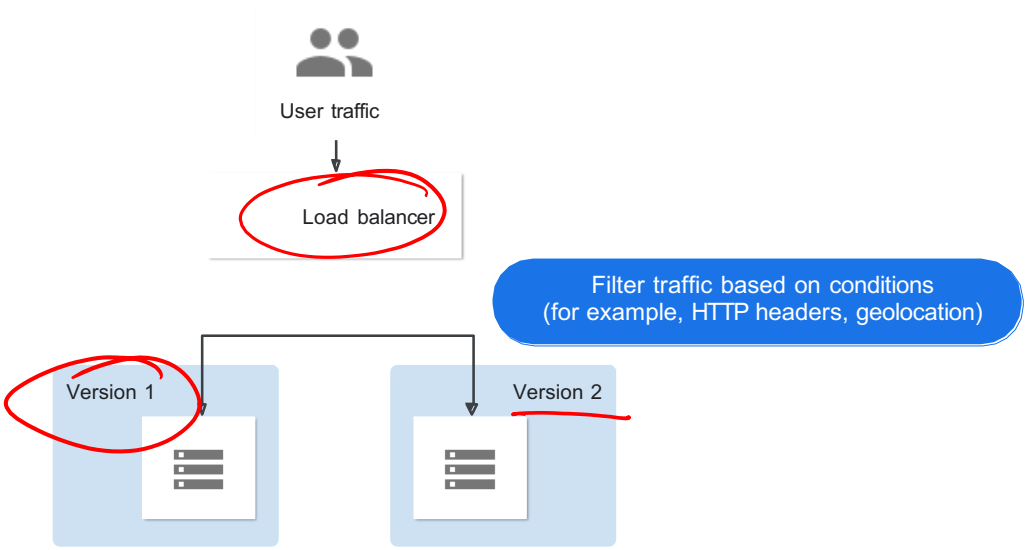
Canary deployment is an update strategy where traffic is gradually shifted to the new version



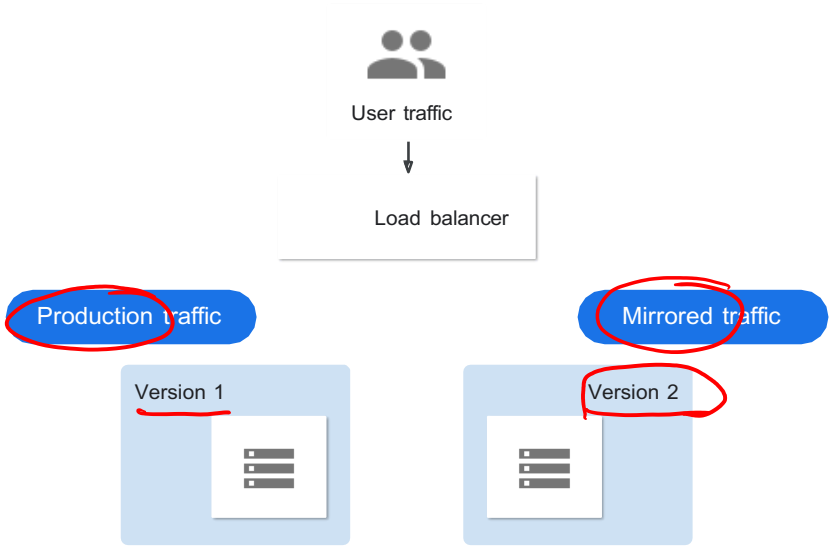
Session affinity ensures that all client requests are sent to the same Pod



A/B testing is used to measure the effectiveness of functionality in an application



Shadow testing allows you to run a new, hidden version



Choosing the right strategy

Deployment or testing pattern	Zero downtime	Real production traffic testing	Releasing to users based on conditions	Rollback duration	Impact on hardware and cloud costs
Recreate Version 1 is terminated, and Version 2 is rolled out.	✗	✗	✗	Fast but disruptive due to downtime	No extra setup required
Rolling update Version 2 is gradually rolled out and replaces Version 1.	✓	✗	✗	Slow	Can require extra setup for surge upgrades
Blue/green Version 2 is released alongside version 1; the traffic is switched to Version 2 after it is tested.	✓	✗	✗	Instant	Need to maintain blue and green environments simultaneously
Canary Version 2 is released to a subset of users, followed by a full rollout.	✓	✓	✗	Fast	No extra setup required
A/B Version 2 is released, under specific conditions, to a subset of users.	✓	✓	✓	Fast	No extra setup required
Shadow Version 2 receives real-world traffic without impacting user requests.	✓	✓	✗	Does not apply	Need to maintain parallel environments in order to capture and replay user requests

Rolling back a deployment (only works with CLI)

```
$ kubectl rollout undo deployment [DEPLOYMENT_NAME]
```

```
$ kubectl rollout undo deployment [DEPLOYMENT_NAME] --to-revision=2
```

```
$ kubectl rollout history deployment [DEPLOYMENT_NAME]
```

- Default: 10 Revision
- Change: .spec.revisionHistoryLimit

Delete a deployment

```
$ kubectl delete deployment [DEPLOYMENT_NAME]
```

GKE rolling update doc:

https://cloud.google.com/kubernetes-engine/docs/how-to/updating-apps#kubectl-apply