

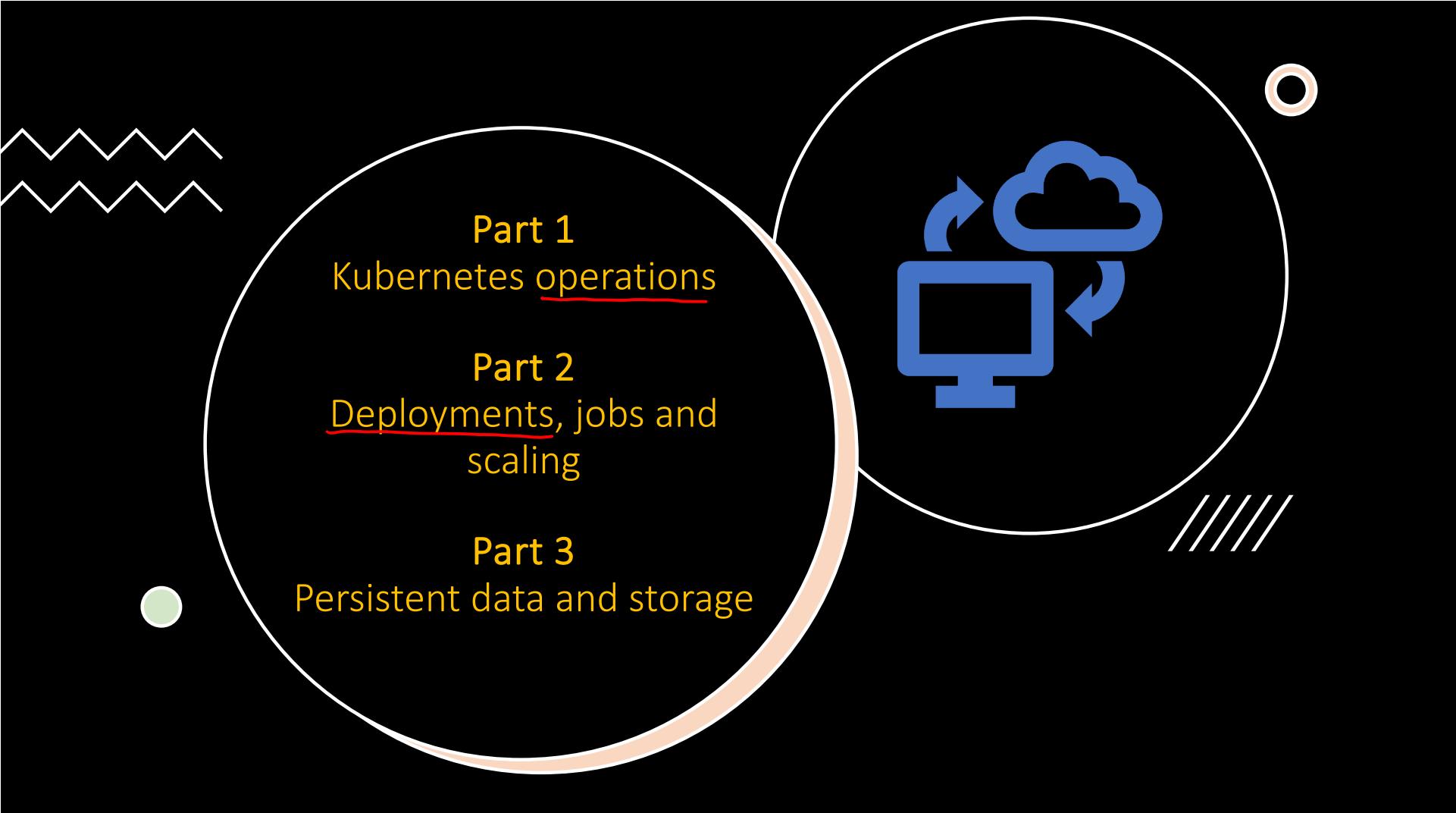


CSCI 5409 Cloud Computing  
Fall, 2023  
Instructor: Dr. Lu Yang

Kubernetes Workload &  
Production (1)  
Oct 13, 2023

## Housekeeping and Feedback

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

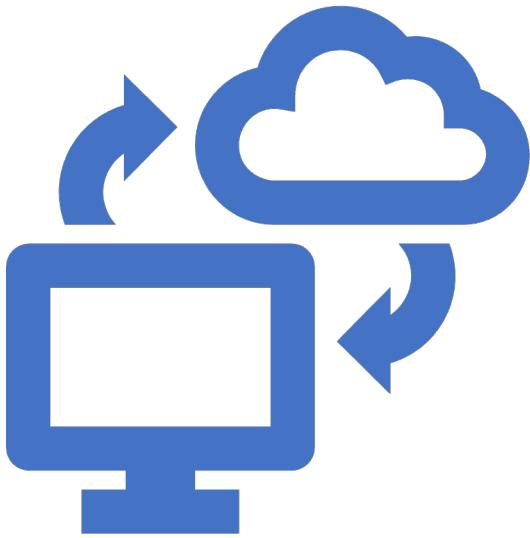


**Part 1**  
Kubernetes operations

**Part 2**  
Deployments, jobs and  
scaling

**Part 3**  
Persistent data and storage





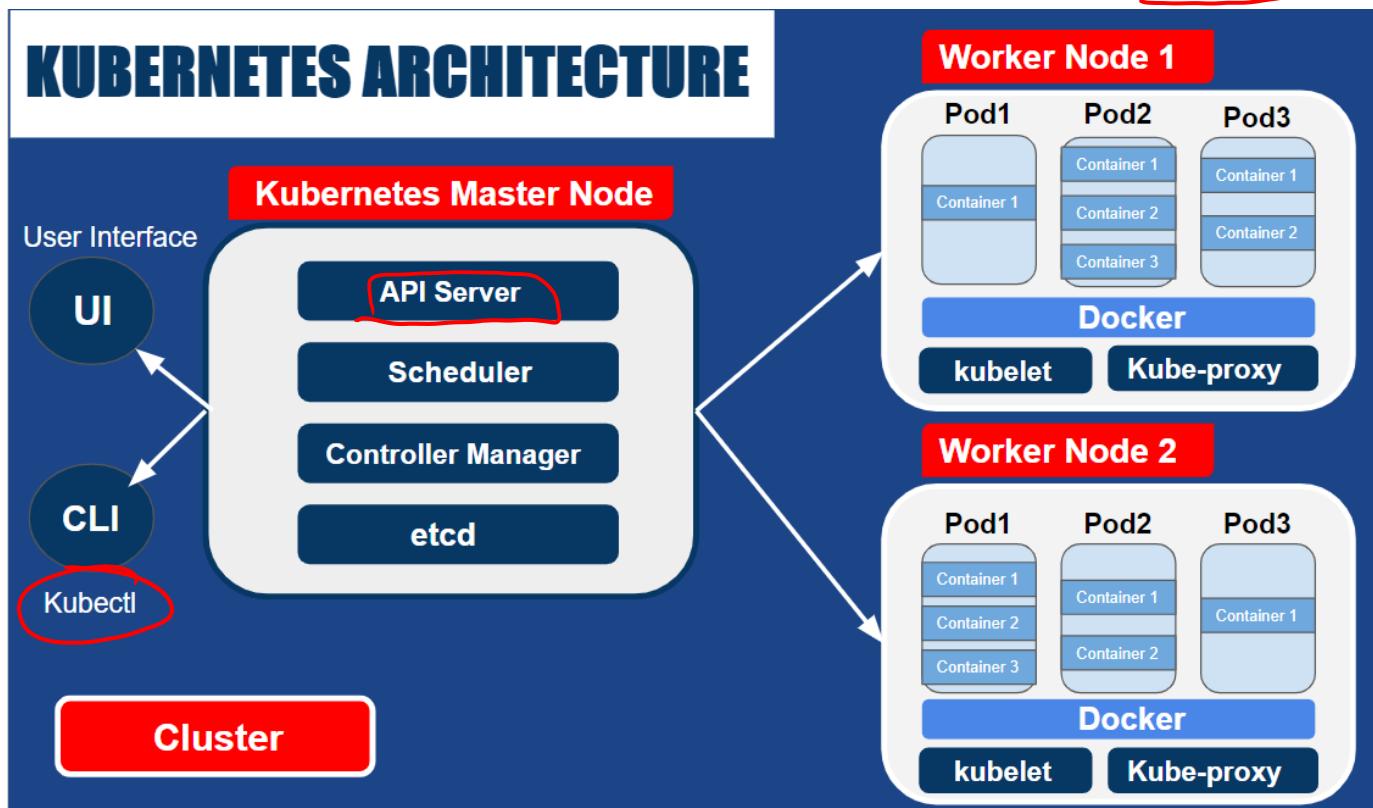
## Part 1 Kubernetes operation

- The kubectl command
- Introspection
- Summary

## Part 1 Kubernetes operation

### The kubectl command

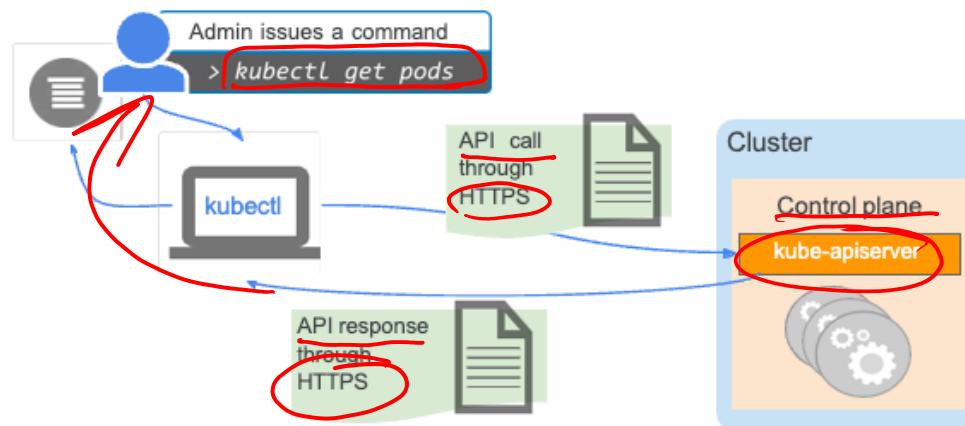
Kubectl transforms your command-line entries into API calls



## Part 1 Kubernetes operation

### The kubectl command

Use kubectl to see a list of Pods in a cluster



[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 1 Kubernetes operation

### The kubectl command

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*

## Part 1 Kubernetes operation

### The kubectl command

Connecting to a Google Kubernetes Engine cluster

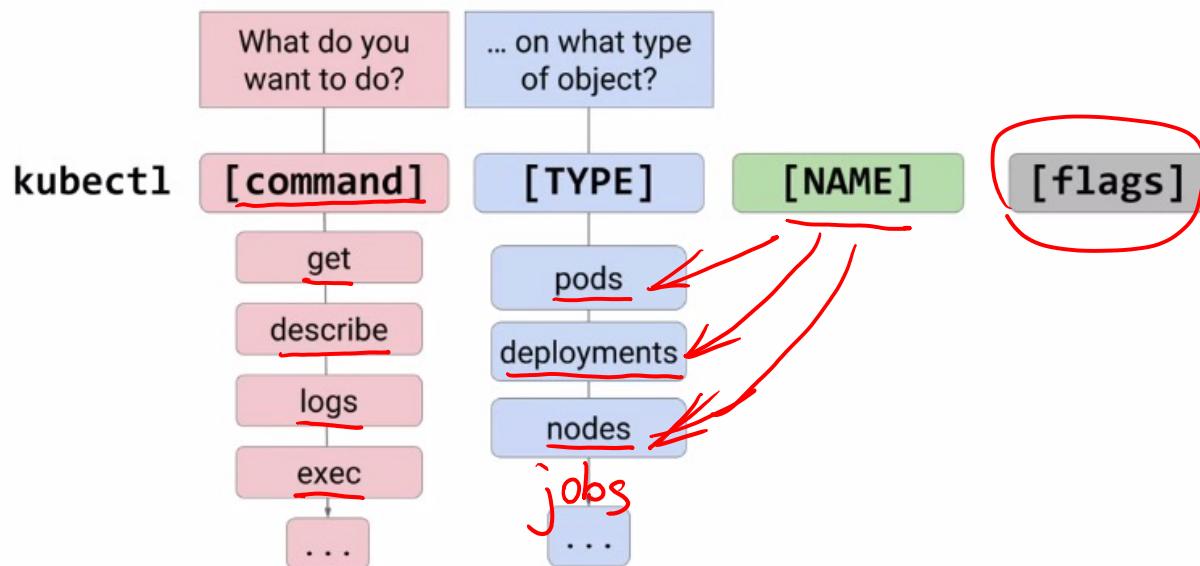
```
$ gcloud container clusters \  
get-credentials [CLUSTER_NAME] \  
--zone [ZONE_NAME]
```

gcloud vs. kubectl  
*internal*

## Part 1 Kubernetes operation

### The kubectl command

#### Explaining kubectl syntax

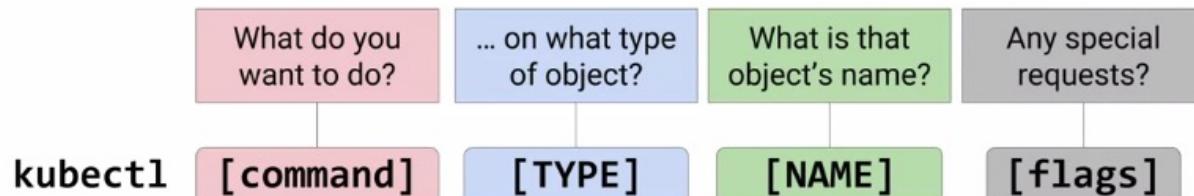


[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## 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/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 1 Kubernetes operation

### The kubectl command

kubectl has many uses



Create Kubernetes  
objects



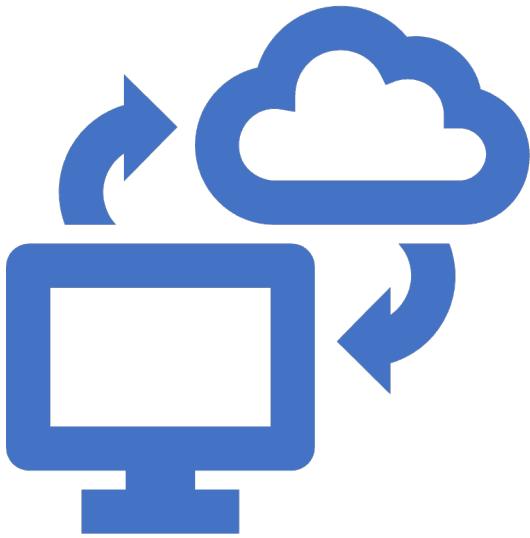
View objects



Delete objects



View and export  
configurations



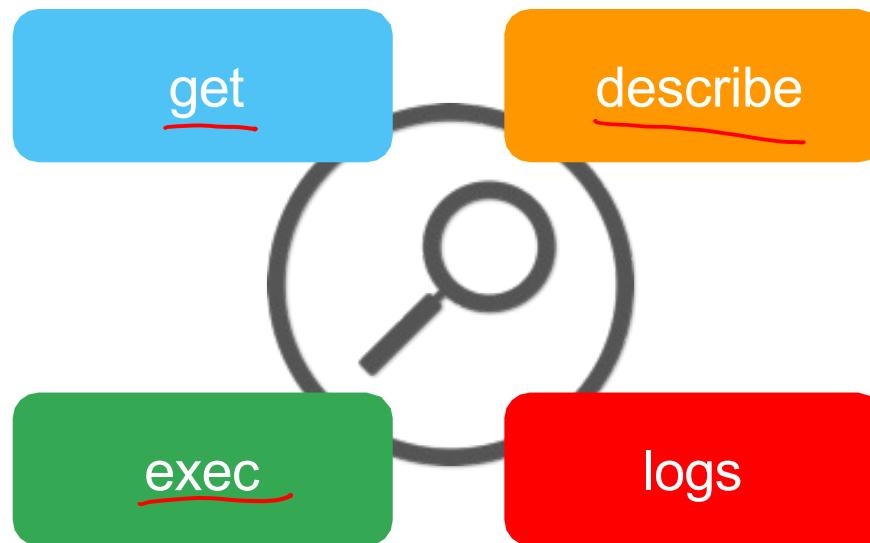
## Part 1 Kubernetes operation

- The kubectl command
- Introspection
- Summary

## Part 1 Kubernetes operation

### Introspection

Use kubectl to gather info about your app



[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](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 *code*
- CrashLoopBackOff *code*

[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 1 Kubernetes operation

### Introspection

#### The details: “describe”

get    describe    exec    logs

##### Describing a Pod

```
$ kubectl describe pod [POD_NAME]
```

###### Pod:

Name  
Namespace  
Node name  
Labels  
Status  
IP address, etc.

###### Container:

State (*waiting, running, terminated*)  
Images  
Ports  
Commands  
Restart counts, etc.

[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 1 Kubernetes operation

### Introspection

#### Interaction: “exec”

get    describe    exec    logs

##### Running a command within a Pod

```
$ kubectl exec [POD_NAME] -- [command]  
$ kubectl exec demo env  
$ kubectl exec demo ps aux  
$ kubectl exec demo cat /proc/1/mounts  
$ kubectl exec demo ls /
```

```
$ kubectl exec demo ls /  
bin  
boot  
dev  
etc  
home  
lib  
lib64  
media  
mnt  
opt  
proc  
root
```

[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 1 Kubernetes operation

### Introspection

## 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/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 1 Kubernetes operation

### Introspection

#### The history: “logs”

get    describe    exec    **logs**

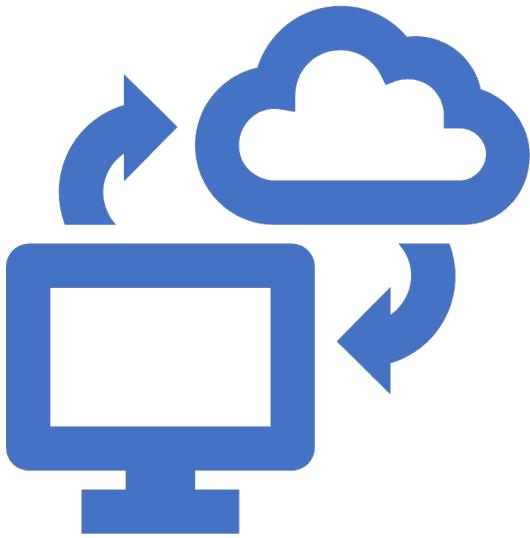
Getting logs for a Pod

```
$ 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/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)



## Part 1 Kubernetes operation

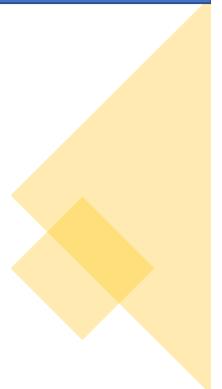
- The kubectl command
- Introspection
- Summary

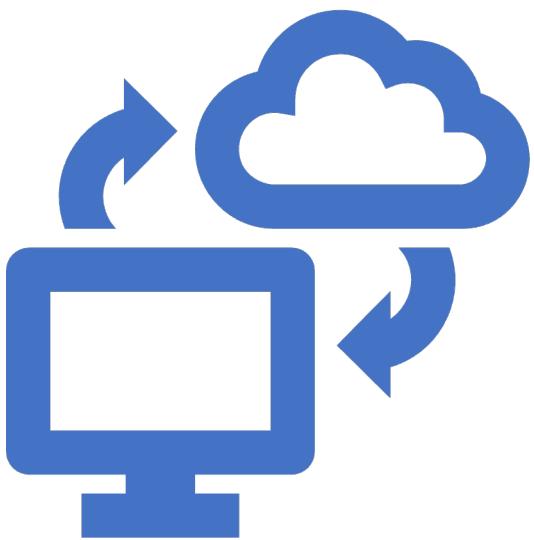
## Part 1 Kubernetes operation

### Summary

#### Summary

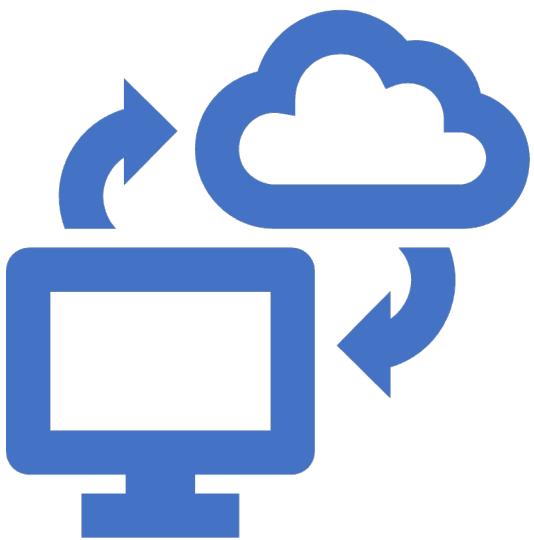
- kubectl is a utility used by administrators to control Kubernetes clusters.
- kubectl syntax is composed of several parts.
- kubectl has many uses.
- Use kubectl to gather info about your app.





## 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

## Part 2 Deployments, jobs, and scaling

### Deployments

Deployments declare the state of Pods



Roll out updates to  
the Pods



Roll back Pods to  
previous revision



Scale or autoscale  
Pods

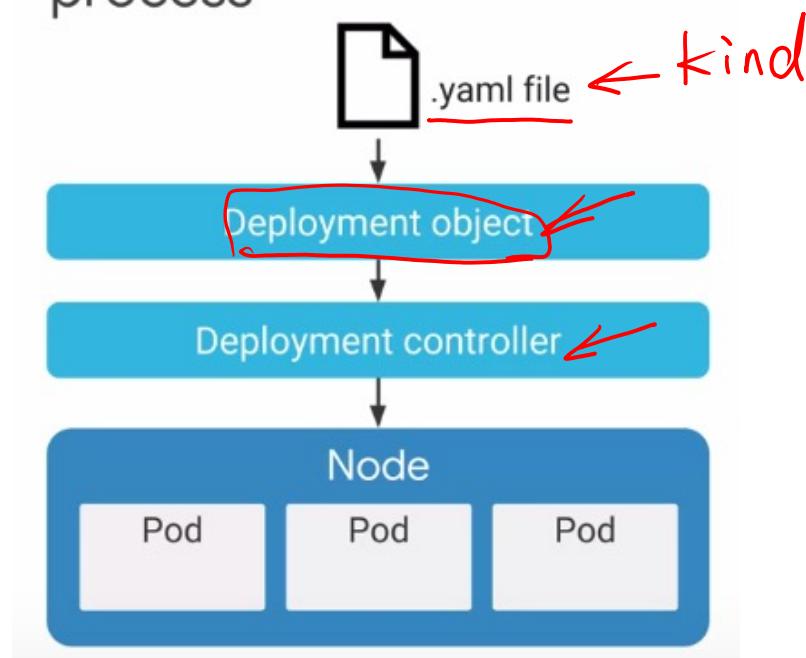


Well-suited for  
stateless applications

## Part 2 Deployments, jobs, and scaling

### Deployments

Deployment is a two-part process



[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 2 Deployments, jobs, and scaling

### Deployments

#### 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/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 2 Deployments, jobs, and scaling

### Deployments

#### Deployment object file in YAML format



[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 2 Deployments, jobs, and scaling

### Deployments

There are three ways to create a Deployment

cloud shell

.yaml

1

```
$ kubectl apply -f [DEPLOYMENT FILE]
```

2

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

## Part 2 Deployments, jobs, and scaling

### Deployments

There are three ways to create a Deployment

3

The screenshot shows the 'Create a deployment' interface with three tabs:

- Container**: Selected. It includes fields for 'Image path \*' (set to 'nginx:latest'), 'Environment variables' (with an 'ADD ENVIRONMENT VARIABLE' button), and 'Labels' (with a 'Key \*' field set to 'app' and a 'Value' field set to 'nginx-1').
- Configuration**: Shows a brief description of what a deployment is.
- Configuration YAML**: Shows a section for 'VIEW YAML' and a 'Cluster' dropdown set to 'standard-cluster-1 (us-central1-a)'.

At the bottom right of the main panel is a large blue 'DEPLOY' button.

[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 2 Deployments, jobs, and scaling

### Deployments

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

## Part 2 Deployments, jobs, and scaling

### Deployments

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
```

## Part 2 Deployments, jobs, and scaling

### Deployments

Or use the Cloud Console

**nginx-deployment**

To let others access your deployment, expose it to create a service

**OVERVIEW**   **DETAILS**   **REVISION HISTORY**   **EVENTS**   **LOGS**   **YAML**

CPU   Memory

UTC+1   12:30 PM   12:40 PM   12:50 PM   1:00 PM   1:10 PM   UTC+1   12:30

Cluster: standard-cluster-1   Namespace: default   Labels: app: nginx

Logs: Container logs, Audit logs   Replicas: 3 updated, 3 ready, 3 available, 0 unavailable   Pod specification: Revision 1, containers: nginx

Active revisions:

Revision	Name	Status	Summary	Created on	Pods running/1
1	nginx-deployment-5d59d67564	OK	nginx: nginx:1.7.9	Oct 13, 2021, 1:16:02 PM	3/3

Managed pods:

Revision	Name	Status	Restarts	Created on
1	nginx-deployment-5d59d67564-2dsfb	Running	0	Oct 13, 2021, 1:16:02 PM
1	nginx-deployment-5d59d67564-8cknj	Running	0	Oct 13, 2021, 1:16:02 PM
1	nginx-deployment-5d59d67564-d898b	Running	0	Oct 13, 2021, 1:16:02 PM

**nginx-deployment**

To let others access your deployment, expose it to create a service

**OVERVIEW**   **DETAILS**   **REVISION HISTORY**   **EVENTS**   **LOGS**   **YAML**

Cluster: standard-cluster-1   Namespace: default   Created: Oct 13, 2021, 1:16:02 PM   Labels: app: nginx   Annotations: deployment.kubernetes.io/revision: 1

**Replicas**: 3 updated, 3 ready, 3 available, 0 unavailable   **Label selector**: app = nginx   **Update strategy**: Rolling update, Max unavailable: 25%, Max surge: 25%

**Min time ready before available**: 0 s   **Progress deadline**: 600 s   **Revision history limit**: 10

**Pod specification**:

Revision	Labels	Termination grace period	Restart policy	Containers
1	app: nginx	30	Always	nginx

## Part 2 Deployments, jobs, and scaling

### Deployments

You can scale the Deployment manually

```
$ kubectl scale deployment [DEPLOYMENT NAME] --replicas=5
```

The screenshot shows the Kubernetes UI interface. On the left, there is a sidebar with the 'ACTIONS' button and a 'KUBE' button. Below these are several options: 'Autoscale', 'Expose', 'Rolling update', 'Scale', and 'Automated deployment'. The 'Scale' option is highlighted. To the right, a modal window titled 'Edit default-pool' is open. It displays the 'Node version' as '1.20.10-gke.301' with a 'CHANGE' button below it. Under the 'Size' section, there is a field labeled 'Number of nodes \*' containing the value '2', which is circled in red. Below this field is a checkbox labeled 'Enable autoscaling' with a question mark icon.

[https://frankie-yanfeng.github.io/2019/07/12/Kubernetes\\_Basics\\_3-2019/](https://frankie-yanfeng.github.io/2019/07/12/Kubernetes_Basics_3-2019/)

## Part 2 Deployments, jobs, and scaling

### Deployments

You can scale the Deployment automatically

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

#### Autoscale

Automatically scale the number of pods.

Minimum number of Pods (Optional)

1

Maximum number of Pods

3

Target CPU utilization in percent (Optional)

80

[CANCEL](#) [DISABLE AUTOSCALER](#) [AUTOSCALE](#)

## Part 2 Deployments, jobs, and scaling

### Deployments

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]

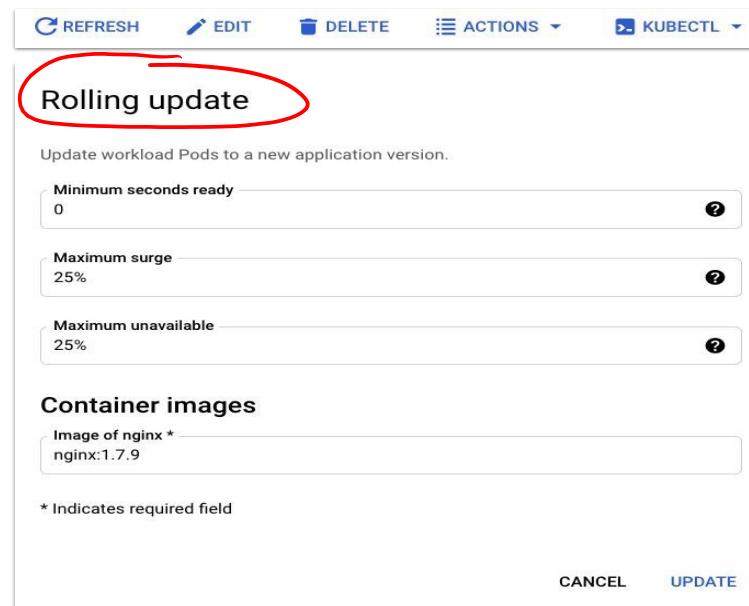
\$ kubectl set image deployment [DEPLOYMENT NAME] [IMAGE] [IMAGE]:[TAG]

\$ kubectl edit \ deployment/[DEPLOYMENT NAME]

## Part 2 Deployments, jobs, and scaling

### Deployments

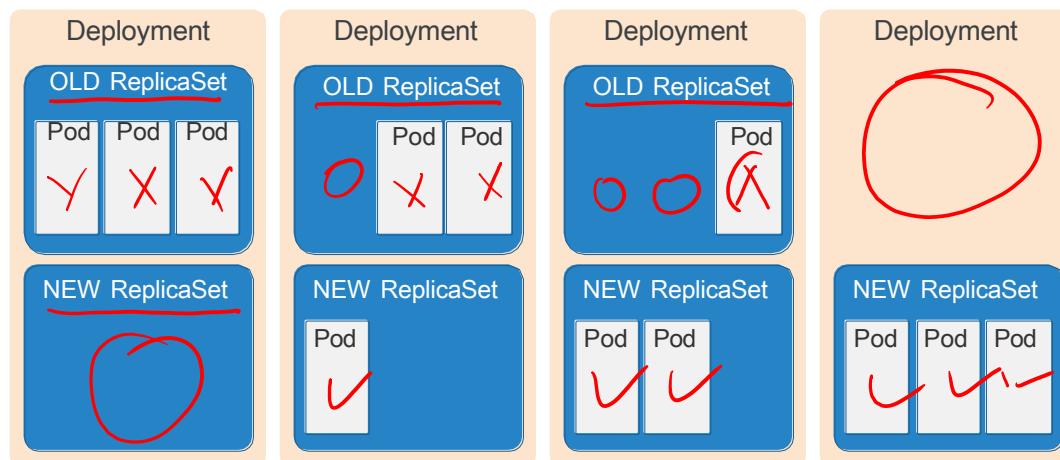
You can update a Deployment in different ways



## Part 2 Deployments, jobs, and scaling

### Deployments

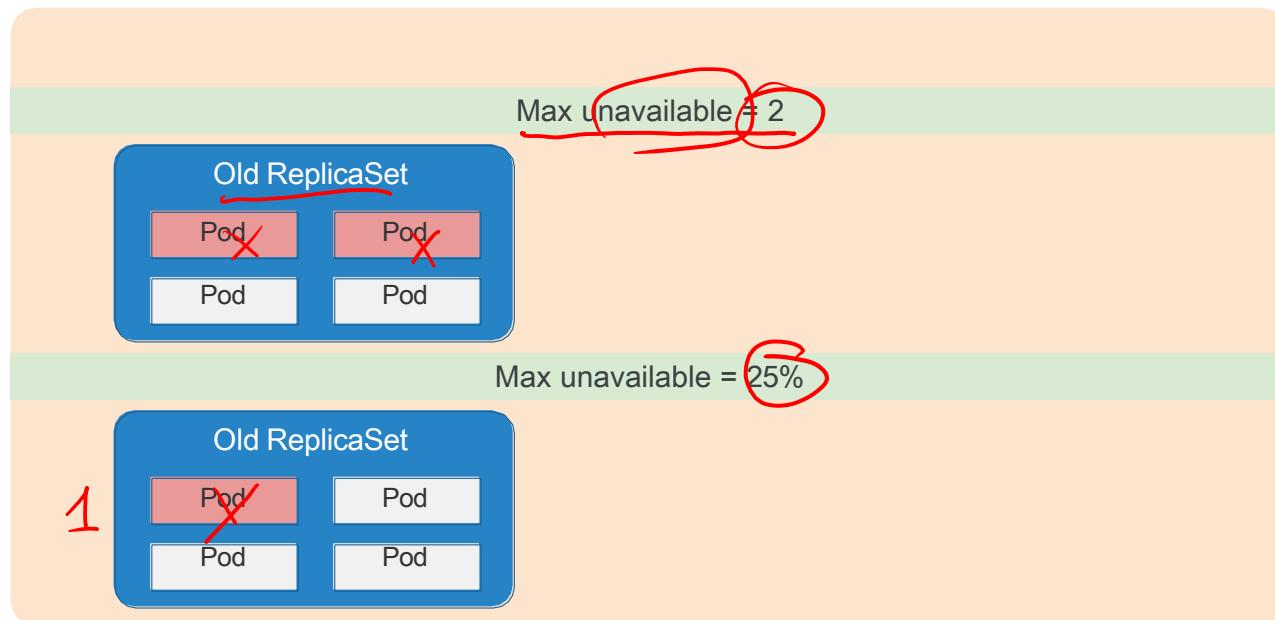
#### The process behind updating a Deployment



## Part 2 Deployments, jobs, and scaling

### Deployments

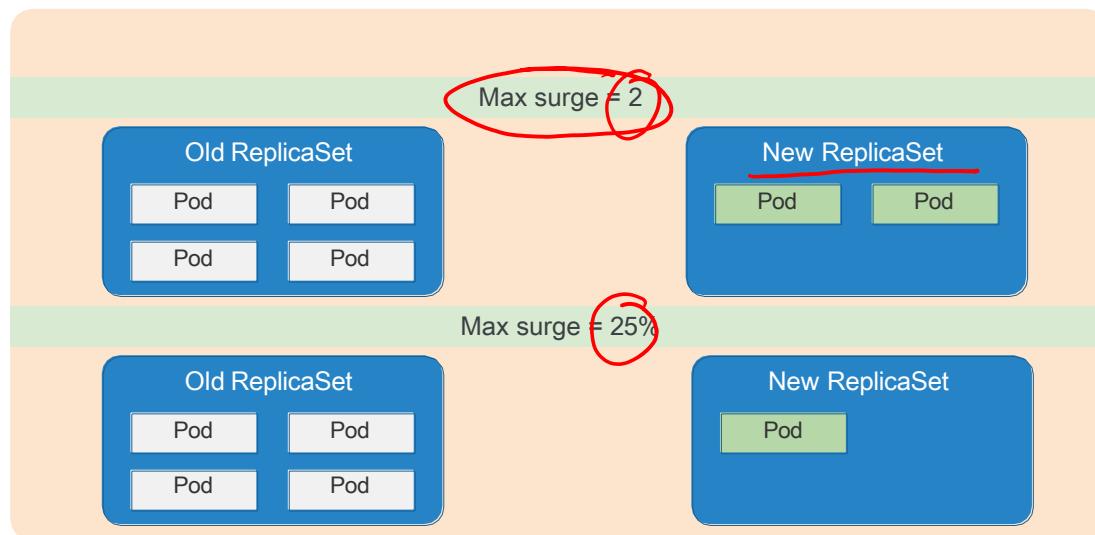
- Set parameters for your rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

Set parameters for your rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

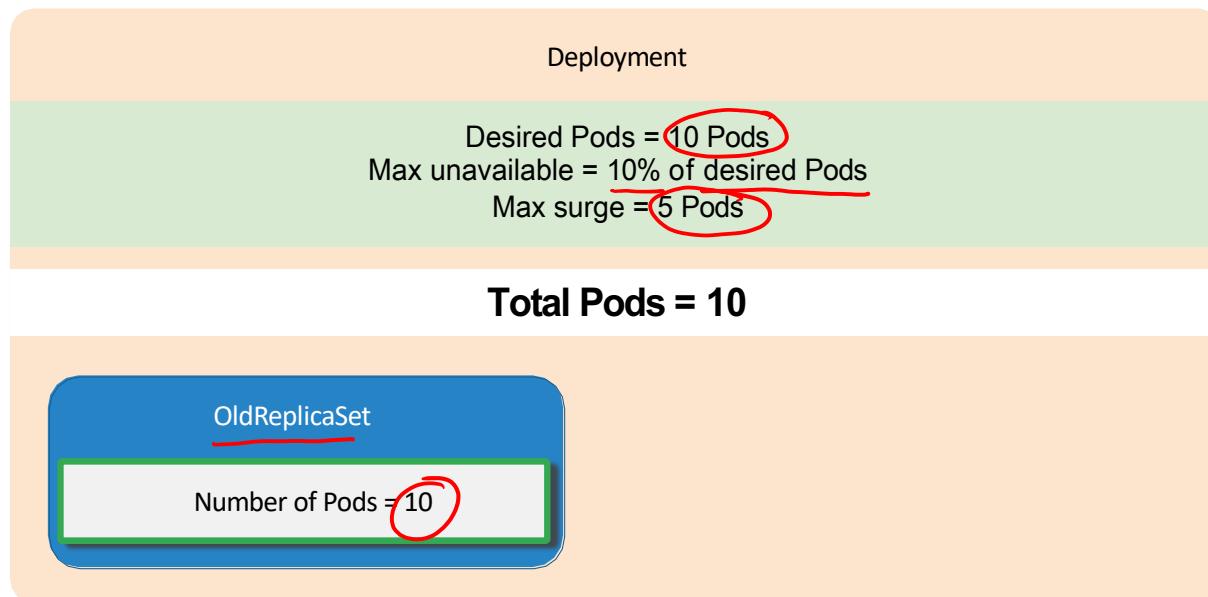
#### An example of a rolling update strategy

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

## Part 2 Deployments, jobs, and scaling

### Deployments

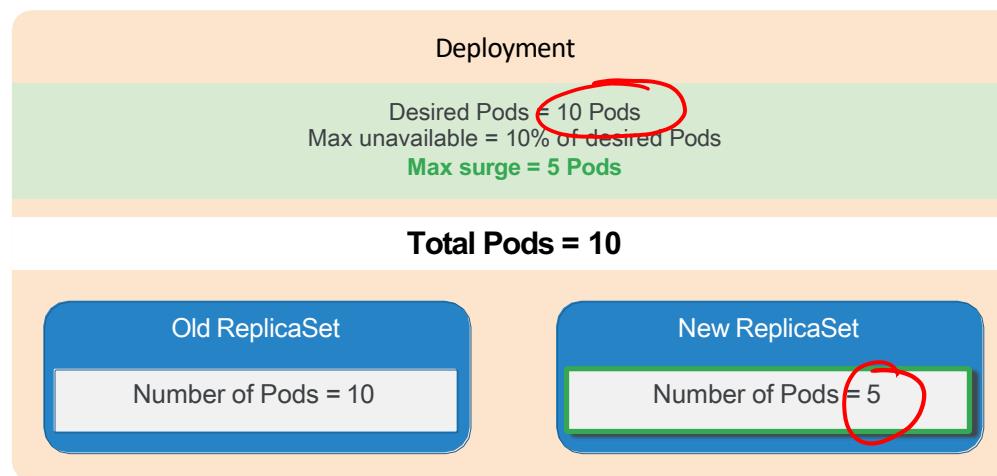
An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

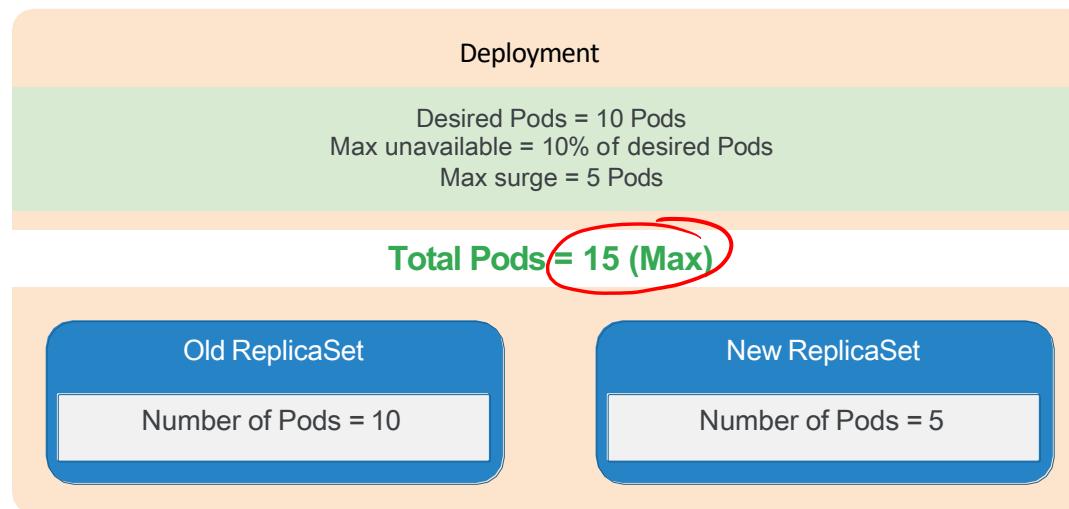
#### An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

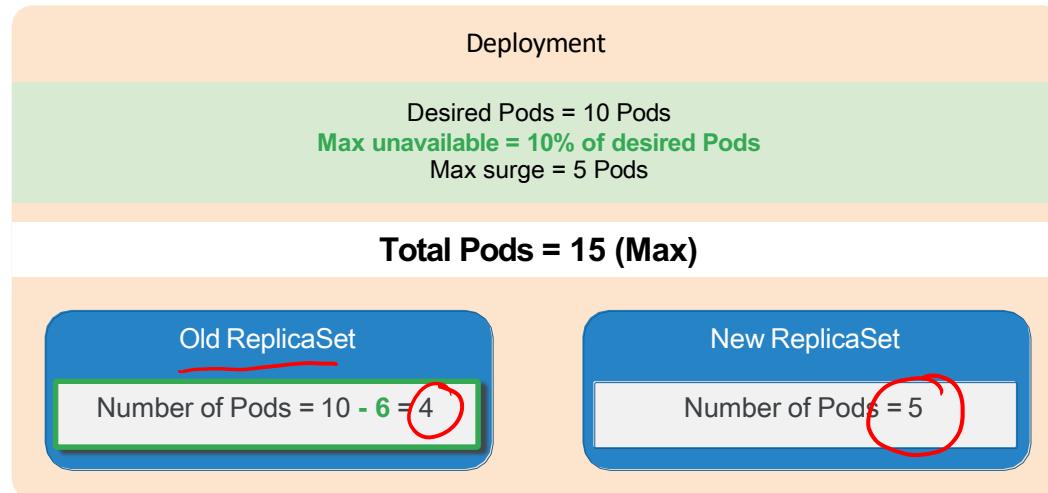
#### An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

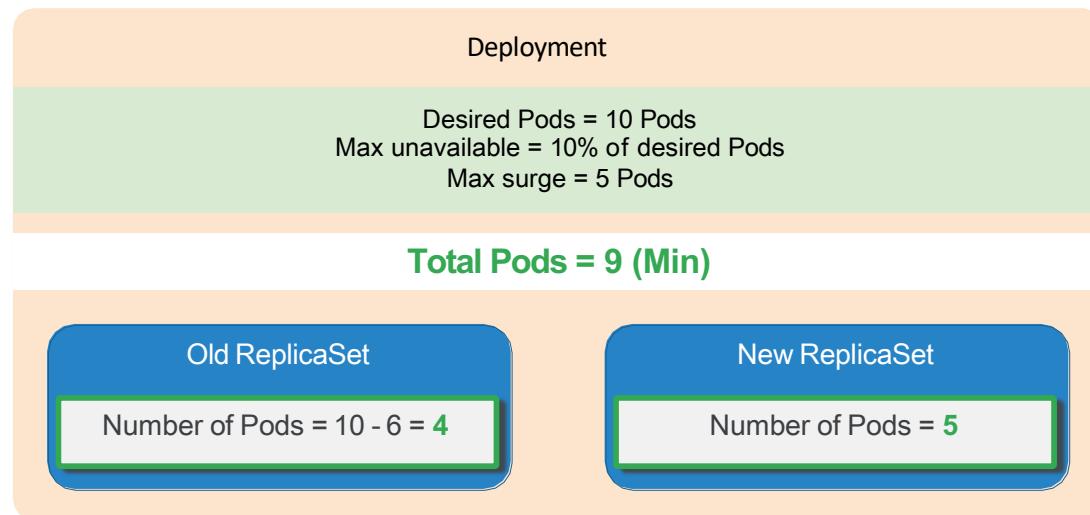
#### An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

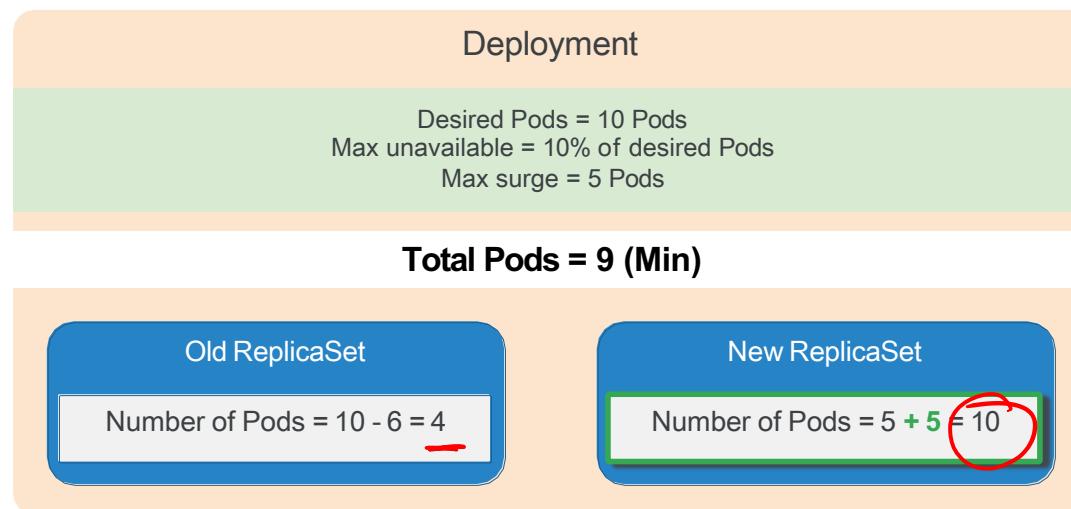
#### An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

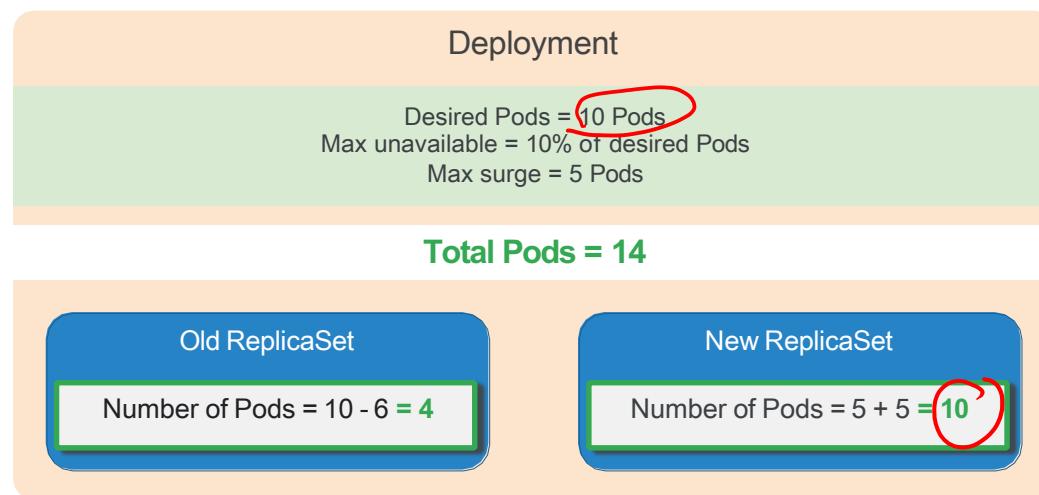
An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

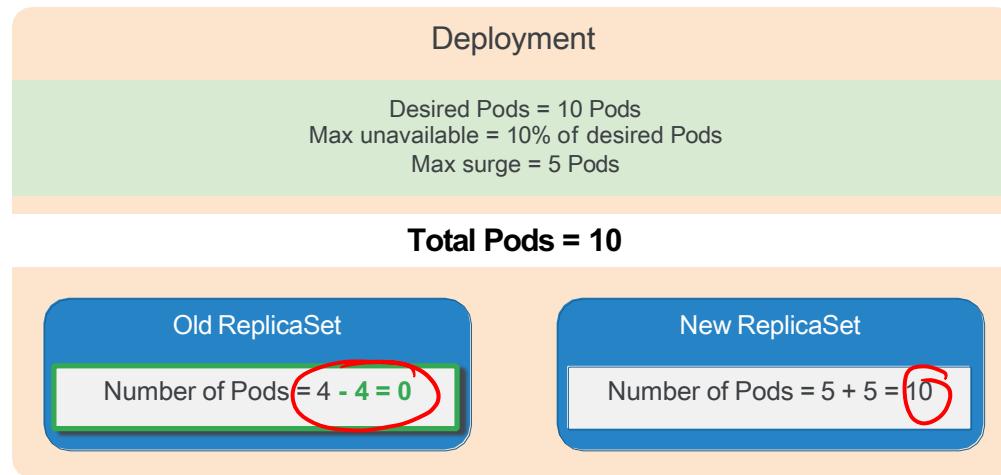
An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

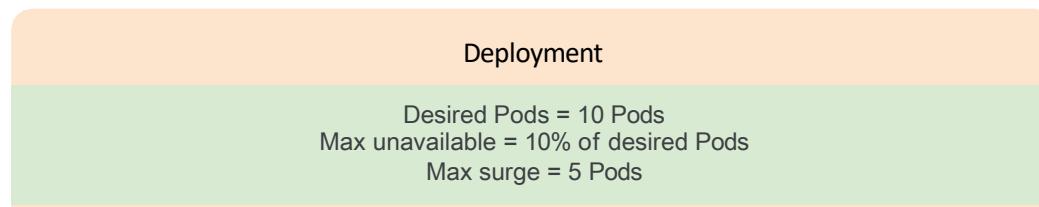
An example of a rolling update strategy



## Part 2 Deployments, jobs, and scaling

### Deployments

#### An example of a rolling update strategy



**Total Pods = 10**



## Part 2 Deployments, jobs, and scaling

### Deployments

#### Applying a Recreate strategy

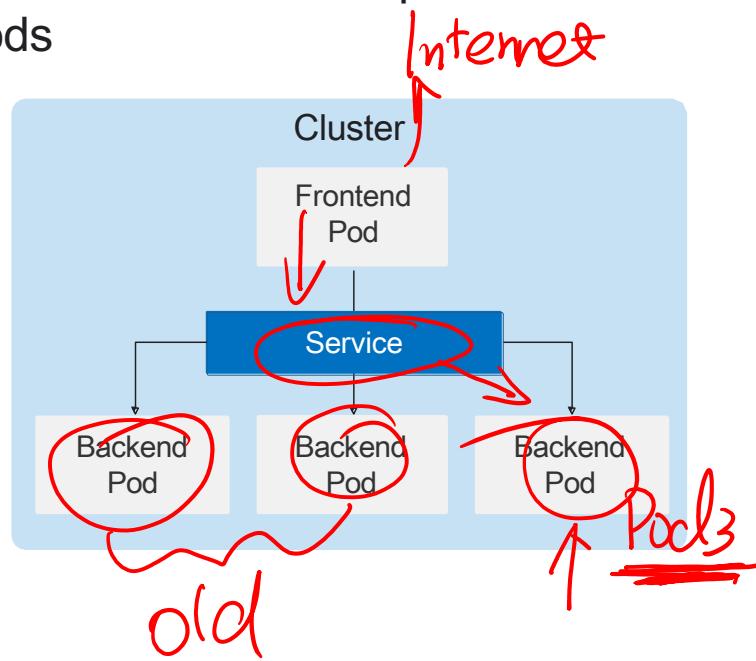
```
[...]  
kind: deployment  
spec:  
  replicas: 10  
  strategy:  
    type: Recreate  
[...]
```

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.

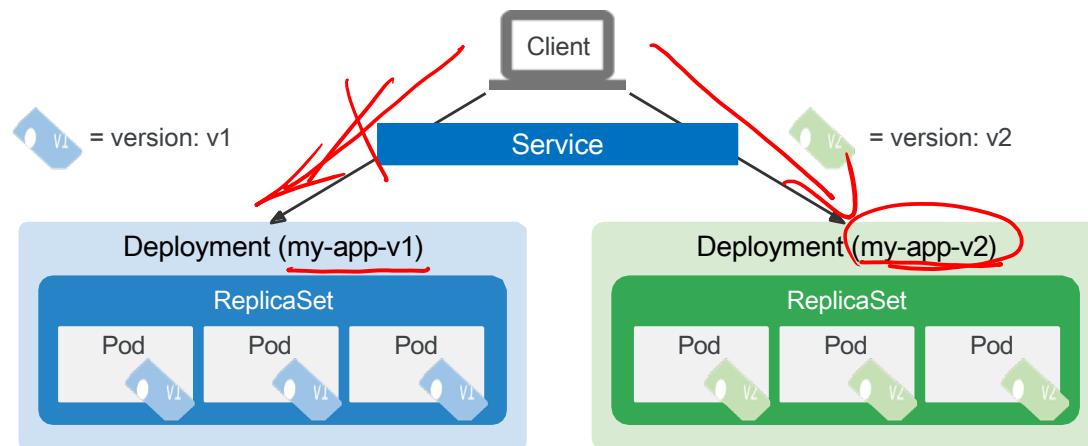
On the other hand, a service is responsible for exposing an interface to those pods, which enables network access from either within the cluster or between external processes and the service.

Cluster IP  
Node Port  
Load Balancer

## Part 2 Deployments, jobs, and scaling

### Deployments

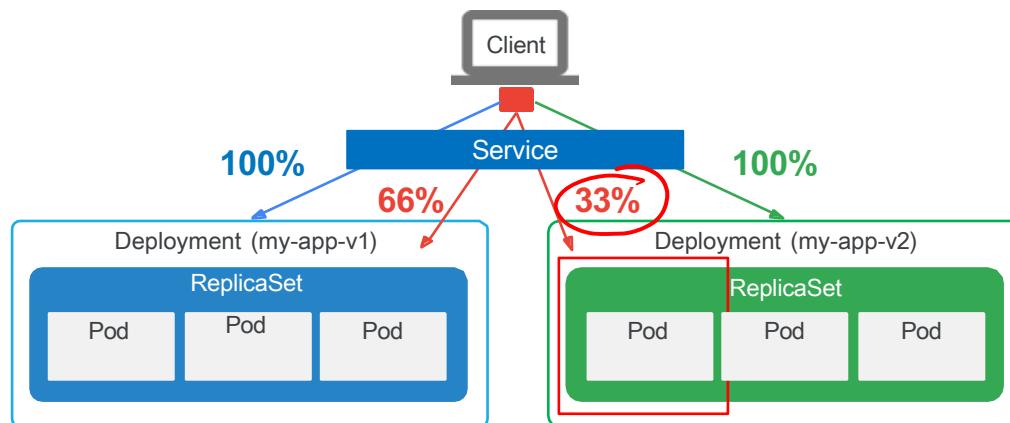
A blue/green deployment strategy ensures app services remain available



## Part 2 Deployments, jobs, and scaling

### Deployments

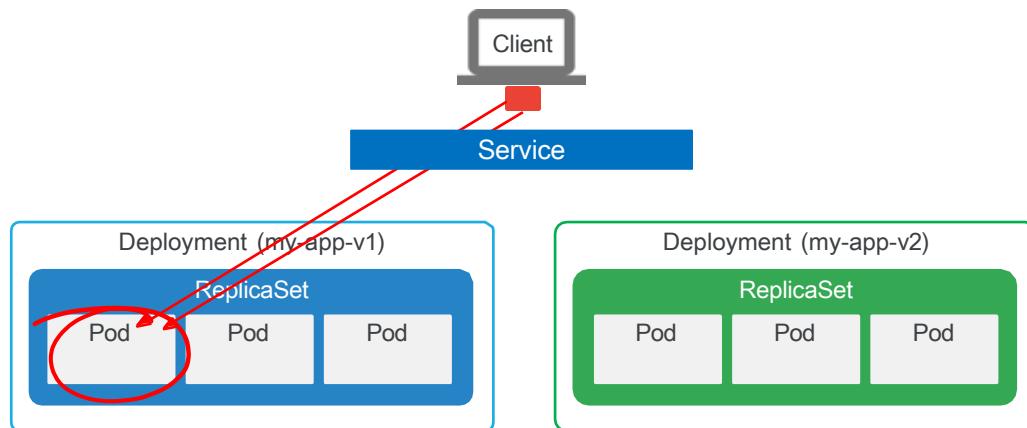
Canary deployment is an update strategy where traffic is gradually shifted to the new version



## Part 2 Deployments, jobs, and scaling

### Deployments

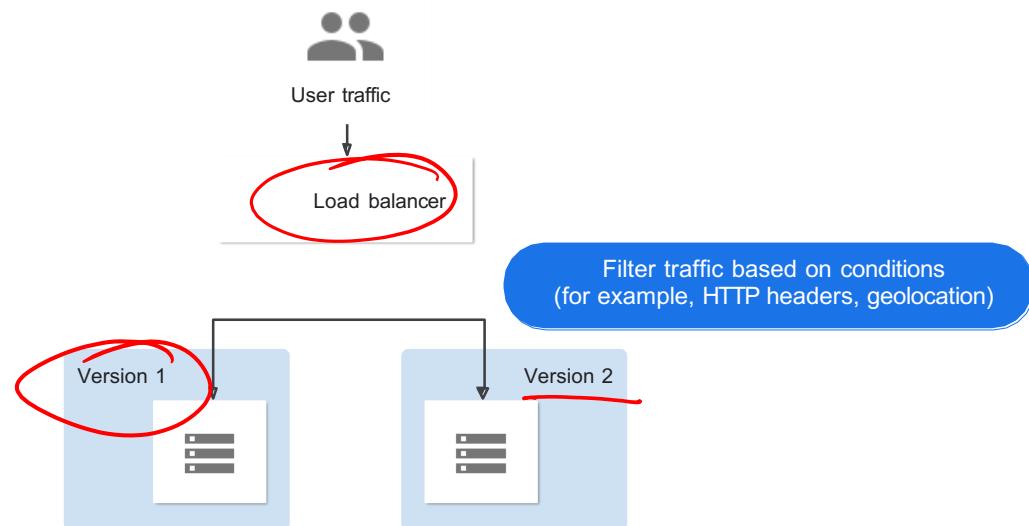
Session affinity ensures that all client requests are sent to the same Pod



## Part 2 Deployments, jobs, and scaling

### Deployments

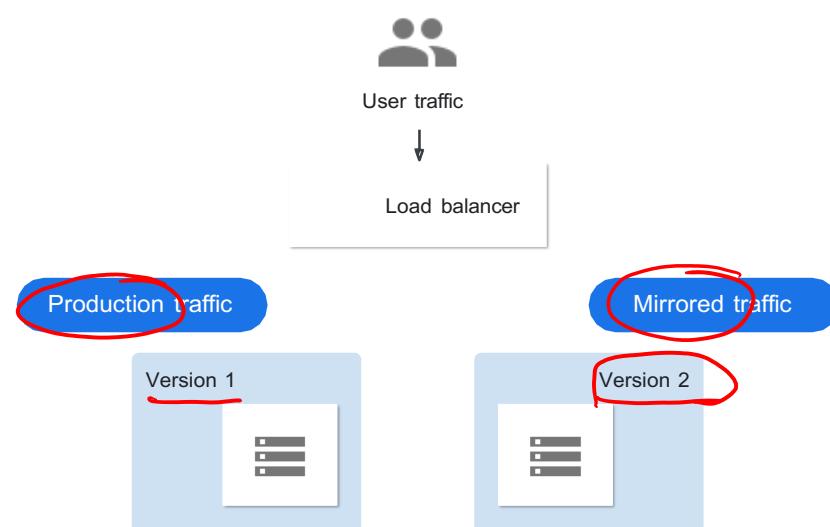
A/B testing is used to measure the effectiveness of functionality in an application



## Part 2 Deployments, jobs, and scaling

### Deployments

Shadow testing allows you to run a new, hidden version



## Part 2 Deployments, jobs, and scaling

### Deployments

# Choosing the right strategy

Deployment or testing pattern	<u>Zero downtime</u>	<u>Real production traffic testing</u>	<u>Releasing to users based on conditions</u>	<u>Rollback duration</u>	<u>Impact on hardware and cloud costs</u>
<b>Recreate</b> Version 1 is terminated, and Version 2 is rolled out.	✗	✗	✗	Fast but disruptive due to downtime	No extra setup required
<b>Rolling update</b> Version 2 is gradually rolled out and replaces Version 1.	✓	✗	✗	Slow	Can require extra setup for surge upgrades
<b>Blue/green</b> 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
<b>Canary</b> Version 2 is released to a subset of users, followed by a full rollout.	✓	✓	✗	Fast	No extra setup required
<b>A/B</b> Version 2 is released, under specific conditions, to a subset of users.	✓	✓	✓	Fast	No extra setup required
<b>Shadow</b> 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

## Part 2 Deployments, jobs, and scaling

### Deployments

#### 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

## Part 2 Deployments, jobs, and scaling

### Deployments

#### 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>