Understand Deployments and How to Perform Rolling Updates

Dan Wahlin
Wahlin Consulting

@danwahlin codewithdan.com

Agenda

Understanding Rolling Update Deployments

Working with Rolling Update Deployments

Rolling Update Deployments in Action

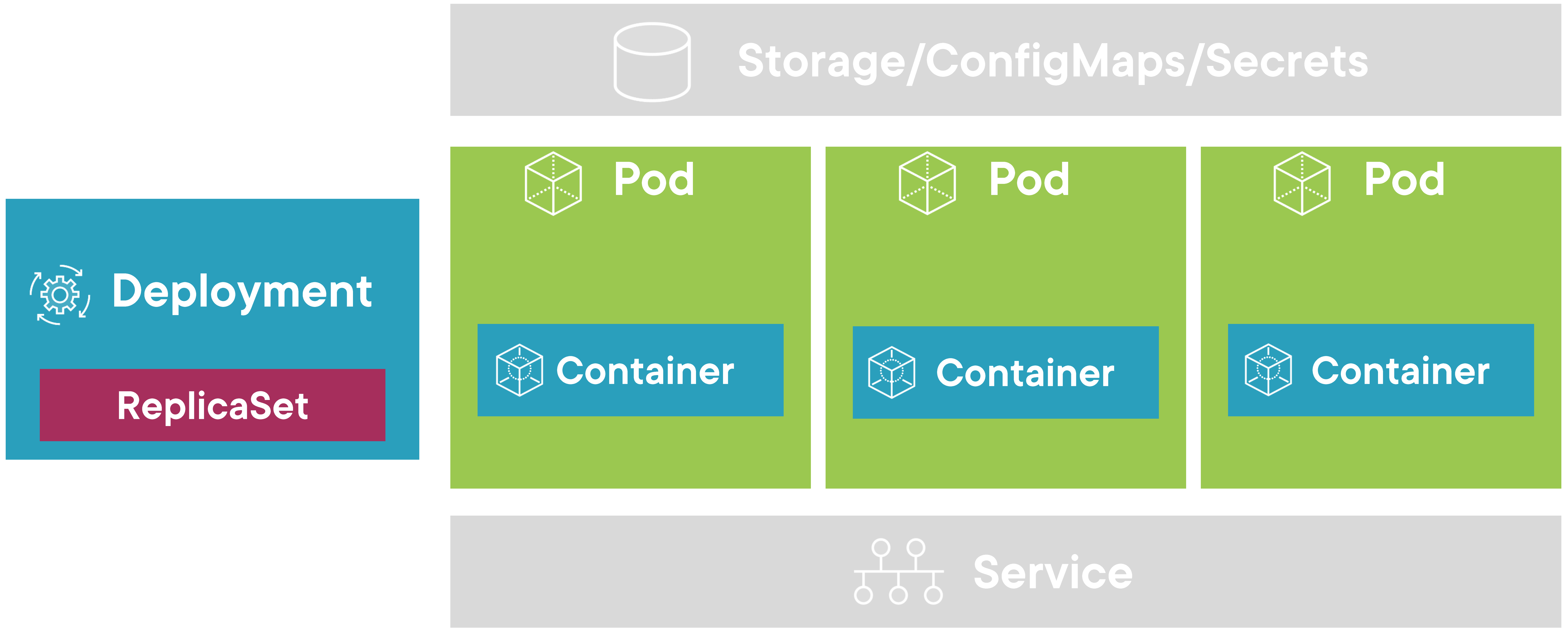
Rolling Back Deployments

Exam Scenarios

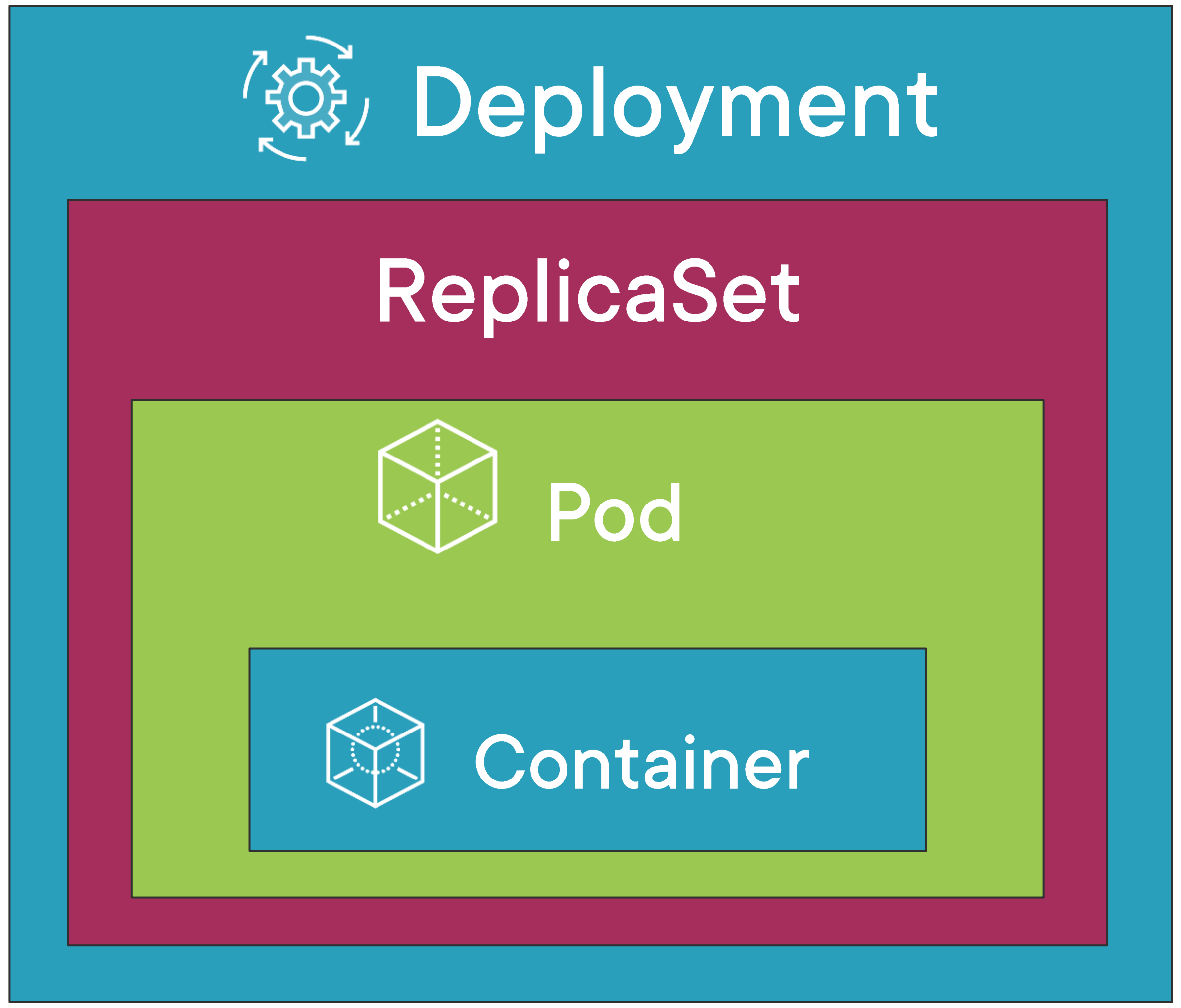
Recap and Test Yourself

Understanding Rolling Update Deployments

Understanding Deployments



Rolling Update Deployments



Increase new Pods while decreasing old Pods

Service handles load balancing traffic to available Pods

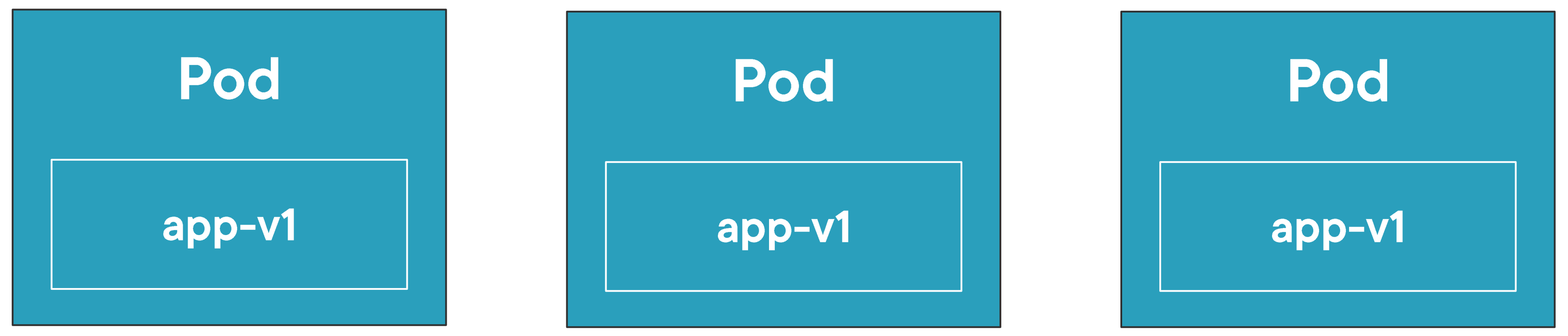
New Pods only scheduled on available Nodes

Deployments support two strategy options:

- Rolling Update (default)
- Recreate (can result in down-time)

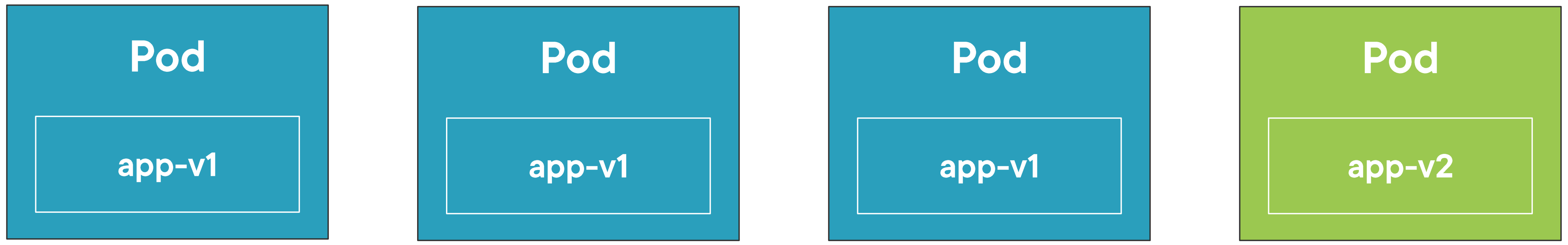
Rolling Update Deployments

Initial Pod State

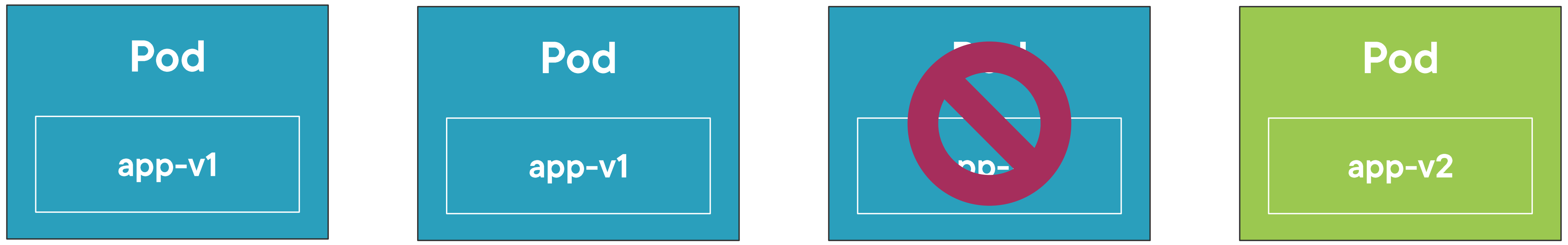


Rolling Update Deployments

Rollout New Pod

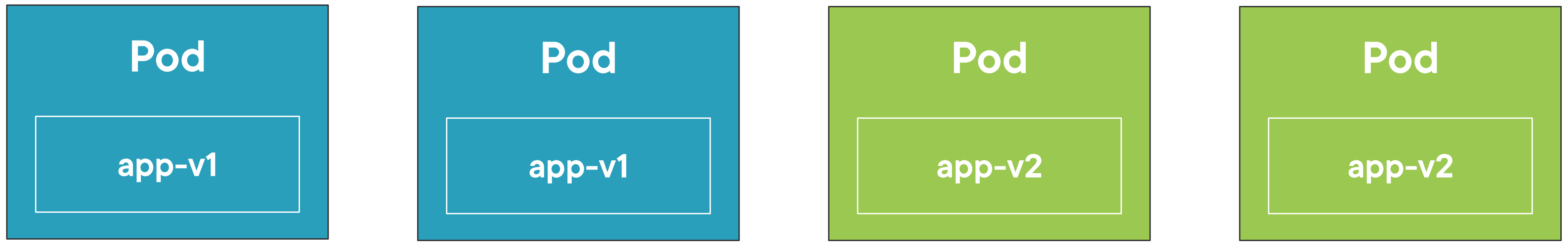


Rolling Update Deployments

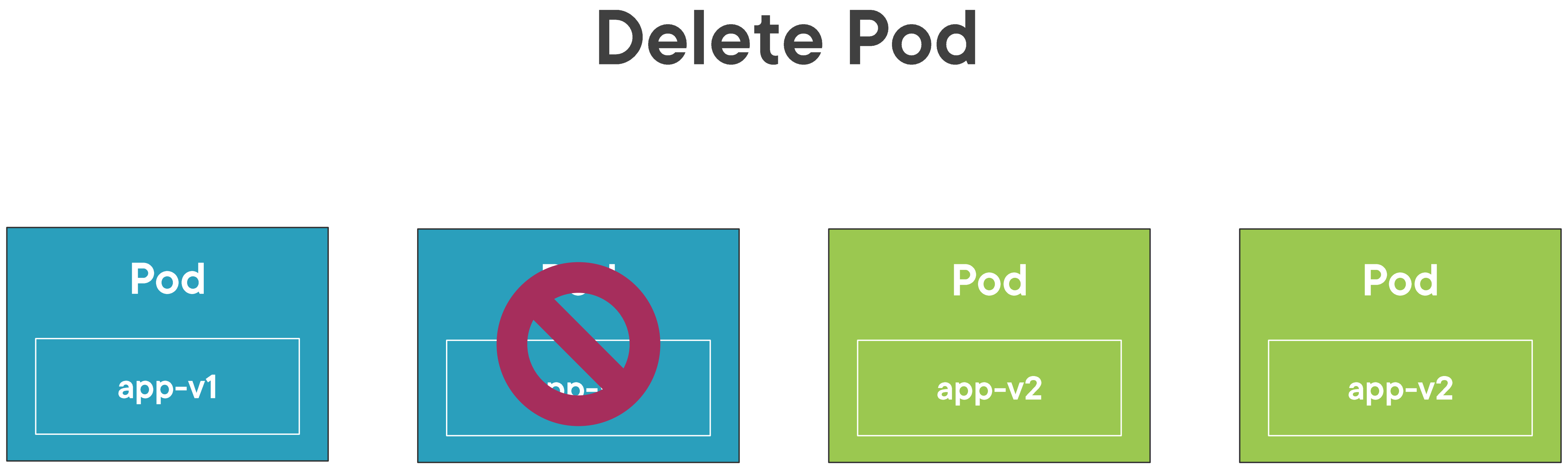


Rolling Update Deployments

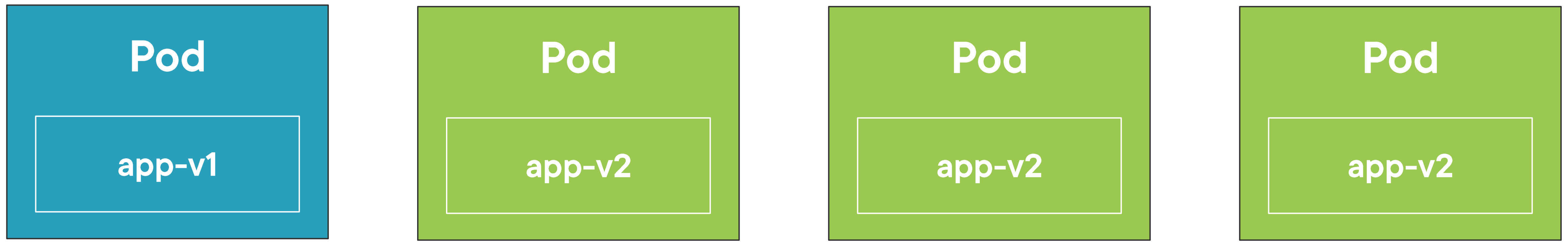
Rollout New Pod



Rolling Update Deployments

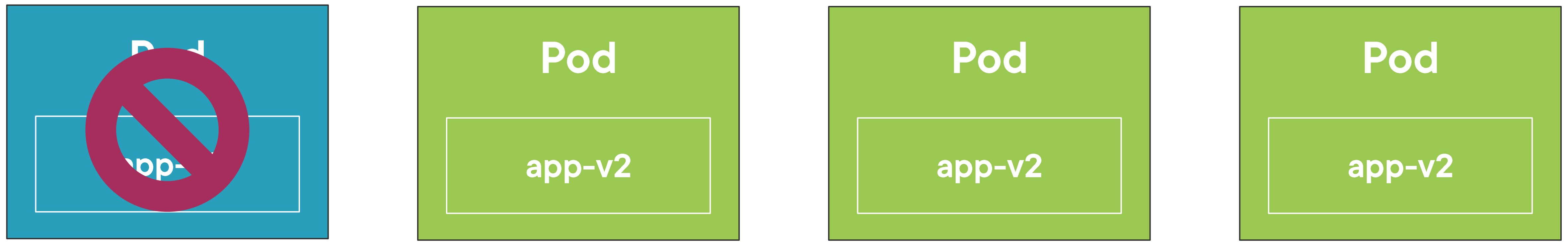


Rolling Update Deployments



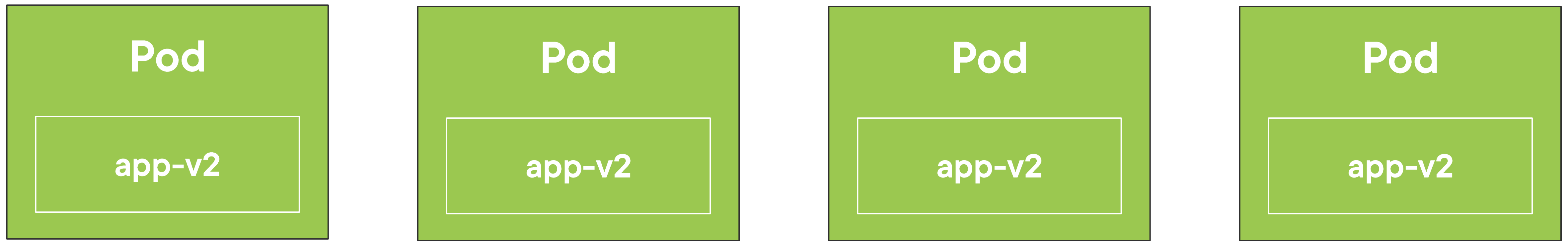
Rolling Update Deployments

Delete Pod



Rolling Update Deployments

Rollout New Pod



Working with Rolling Update Deployments

Defining a Rolling Update Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 4
  minReadySeconds: 1
  progressDeadlineSeconds: 60
  revisionHistoryLimit: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1

  ...
```

◀ Number of Pod replicas

◀ Seconds new Pod should be ready to be considered healthy (0)

◀ Seconds to wait before reporting stalled Deployment

◀ Number of ReplicaSets that can be rolled back (10)

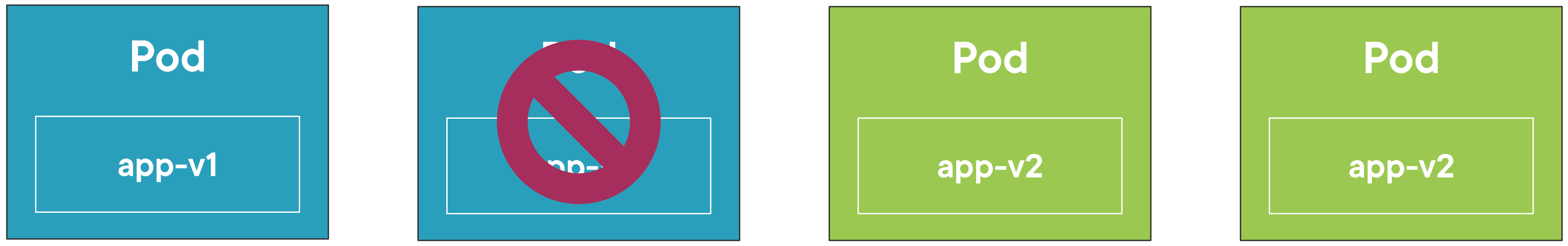
◀ RollingUpdate (default) or Recreate strategy

◀ Max Pods that can exceed the replicas count (25%)

◀ Max Pods that are not operational (25%)

Understanding maxSurge

How many Pods can be added above the replicas count during the rolling update?

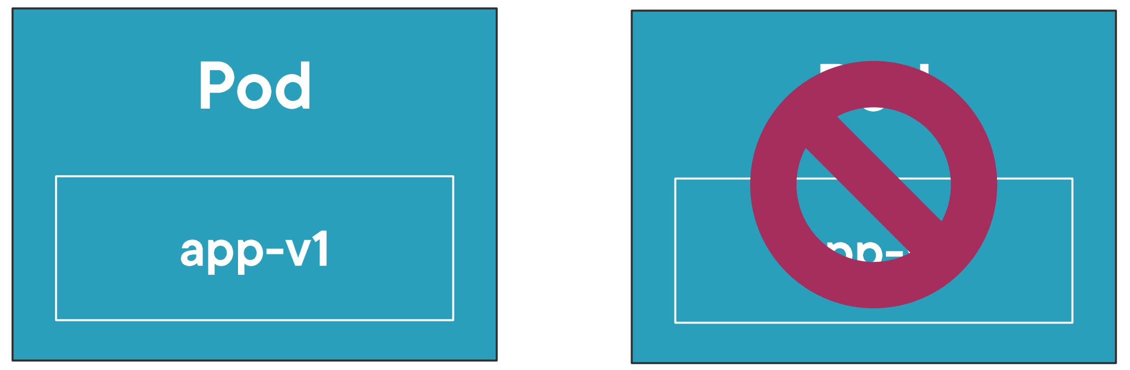


replicas	= 2
maxSurge	= 1
current	= 3

1 above replicas count which is allowed

Understanding maxUnavailable

How many of the existing Pods can be made unavailable during a rolling update?



maxUnavailable = 1

It's OK for 1 of the 2 replicas to be unavailable

Creating the Deployment

Use the kubectl create command along with the --filename or -f switch

Save configuration in resource's annotations

```
# Create initial deployment
kubectl create –f file.deployment.yml --save-config
```

Creating or Modifying a Deployment

Use the kubectl apply command along with the --filename or -f switch

Record the command in the Deployment revision history

Note: May be removed in future

```
# Apply changes to a deployment
kubectl apply -f file.deployment.yml --record=true

# Update deployment annotation
kubectl annotate deployment [name]
  kubernetes.io/change-cause="Change details" --overwrite=true
```

Checking the Deployment Status

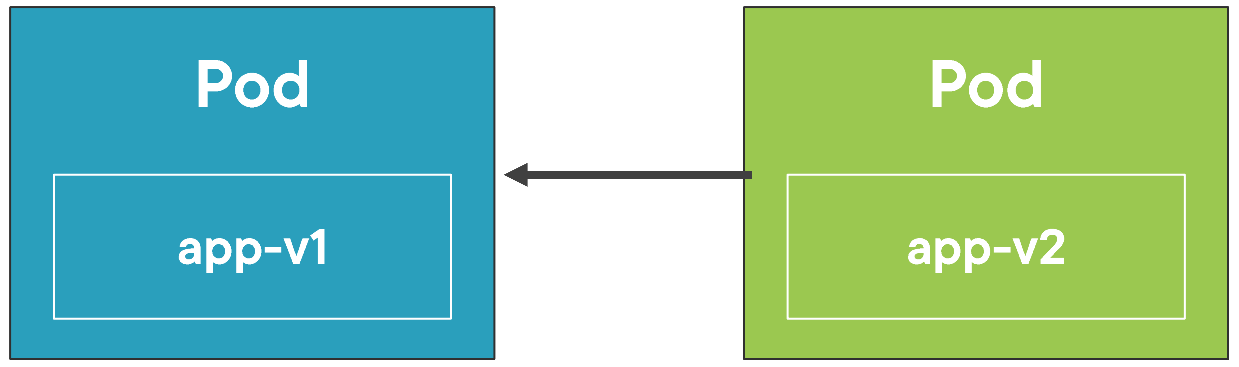
The kubectl rollout status command can be used to get information about a specific Deployment

```
# Get information about a Deployment
kubectl rollout status deployment [deployment-name]
```

Rolling Update Deployments in Action

Rolling Back Deployments

Rolling Back Deployments



Have Deployment issues?

- **Apply a new Deployment**
- **Revert to a previous revision**

Several kubectl commands can be used for rollbacks:

- **kubectl rollout status**
- **kubectl rollout history**
- **kubectl rollout undo**

Checking Deployment History

The kubectl rollout history command can be used to view history of a Deployment

```
# Get information about a Deployment
kubectl rollout history deployment [deployment-name]

# Get information about a specific Deployment revision
kubectl rollout history deployment [deployment-name] --revision=2
```

Rolling Back a Deployment

Use the kubectl rollout undo command to rollback to a specific Deployment revision

```
# Check status
kubectl rollout status -f file.deployment.yml

# Rollback a Deployment
kubectl rollout undo -f file.deployment.yml

# Rollback to a specific revision
kubectl rollout undo -f file.deployment.yml --to-revision=2
```

Exam Scenarios – Task 1

Prev. | Task 1 of X | Next | Task weight: 8%

```
Cluster: ckad0018
Namespace: dev
Doc links: Deployments
```

Task

1. Create a namespace named **dev**.

2. Create a Deployment named **web-app** that uses the **nginx:1.17.8-alpine** image and creates **6** Pods in the **dev** namespace. Add the **--save-config** flag.

3. Verify all the Pods are running.

4. Edit the Deployment and change the image to **nginx:1.23.1-alpine**.

5. Add an annotation named **kubernetes.io/change-cause** with a value of **"Changed to nginx:1.23.1"** to the Deployment.

6. List the value of the **kubernetes.io/change-cause** annotation (display this specific annotation).

7. Verify all Pods are running and that the previous Pods are terminated using **kubectl describe** to view events.

8. List the Deployment's rollout status.

Exam Scenarios – Task 2

Prev. | Task 2 of X | Next | Task weight: 10%

```
Cluster: ckad0018
Namespace: dev
Doc links: Deployments
```

Task

1. Create a Deployment file named **biz-app.deploy.yml**. The Deployment should be named **biz-app**, use an image of **nginx:1.23.1-alpine**, and create **4** Pods.

2. Create the Deployment and add the **--save-config** flag when running the command.

3. Verify all the Pods are running.

4. Edit **biz-app.deploy.yml** and change the image to **nginx:7.85.1-alpine**. Apply the changes and use the **--record** flag.

5. List the Pods but notice there's a problem. View the **biz-app** Deployment's rollout status and rollout history.

6. Rollback to the previous Deployment version. List the Deployment history again.

7. Verify that the previous deployment is running by checking the Deployment's image value (it should be **nginx:1.23.1-alpine** now).

Recap and Test Yourself

Top Three Take-home Points

Rolling updates provide a zero-downtime solution.

When creating a Deployment, use --save-config to store metadata about the Deployment in the annotations.

Deployments support a rollout status and history. You can rollback to a previous Deployment.

Key kubectl Commands

Key kubectl commands to know for the exam.

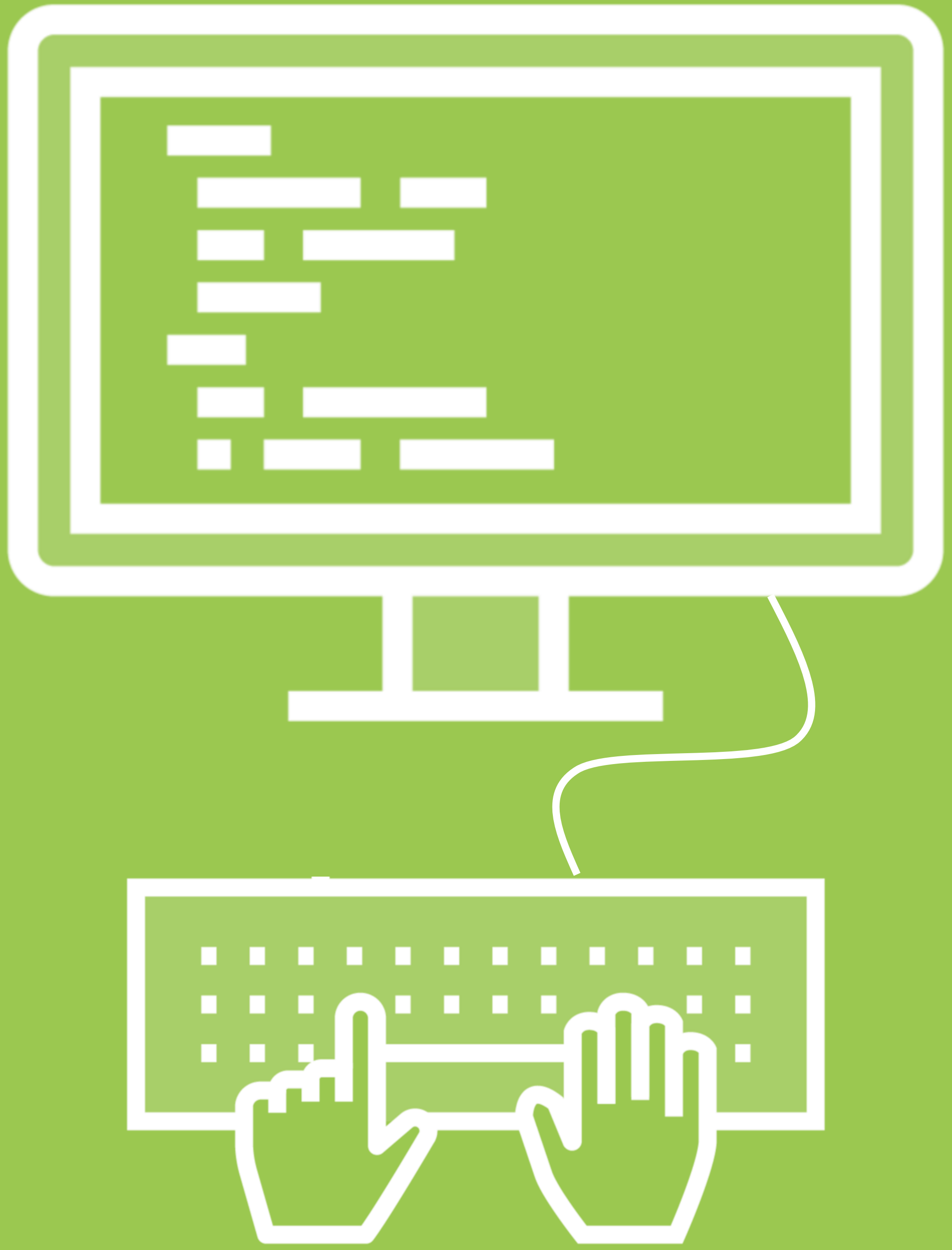
```
k create deploy [deployment-name] --image=[image-name]:[tag]
  --dry-run=client -o yaml > deploy.yaml

k annotate deploy [deployment-name]
kubernetes.io/change-cause="Change to nginx:1.23.1" --overwrite=true

k rollout history deploy [deployment-name]

k rollout undo deploy [deployment-name]

k get deploy web-app -o json | jq
'.metadata.annotations."kubernetes.io/change-cause"'
```



GitHub Repo

https://github.com/nigelpoulton/ckad