Performing Canary Deployments

Dan Wahlin
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

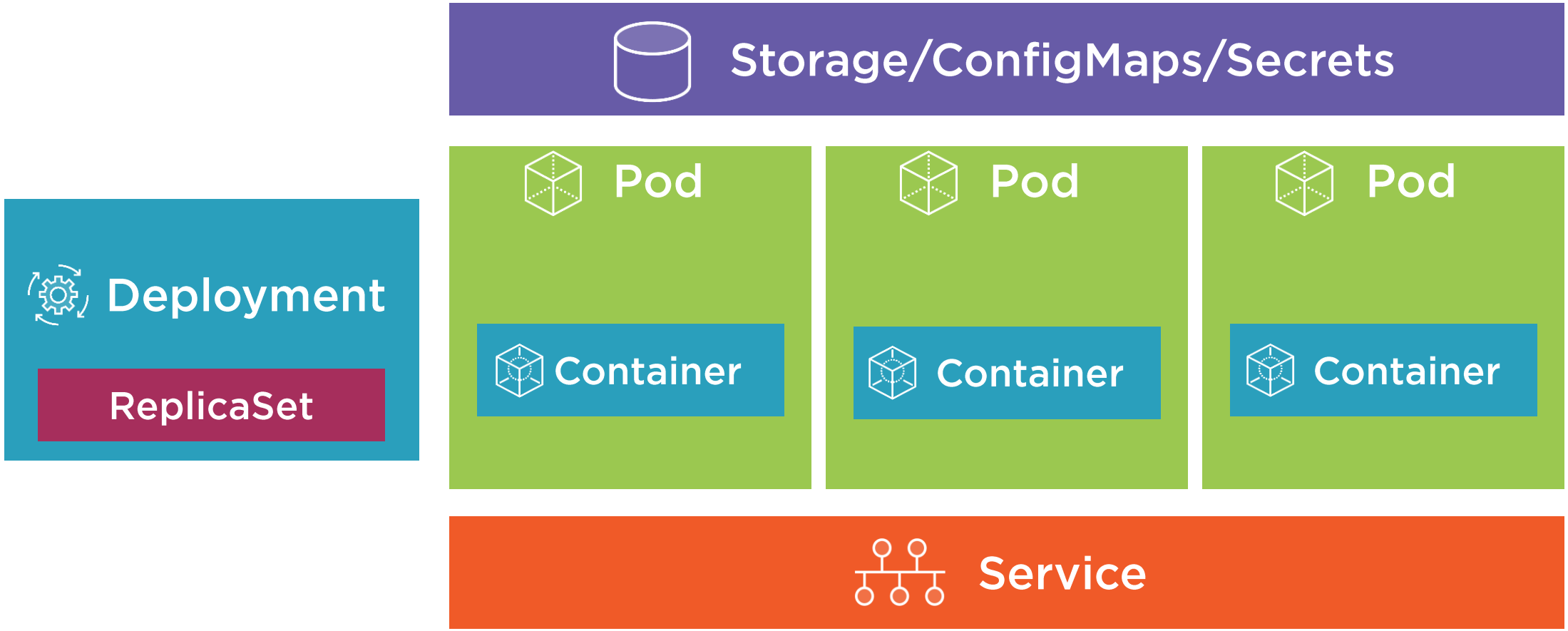
Module Overview

Understanding Canary Deployments

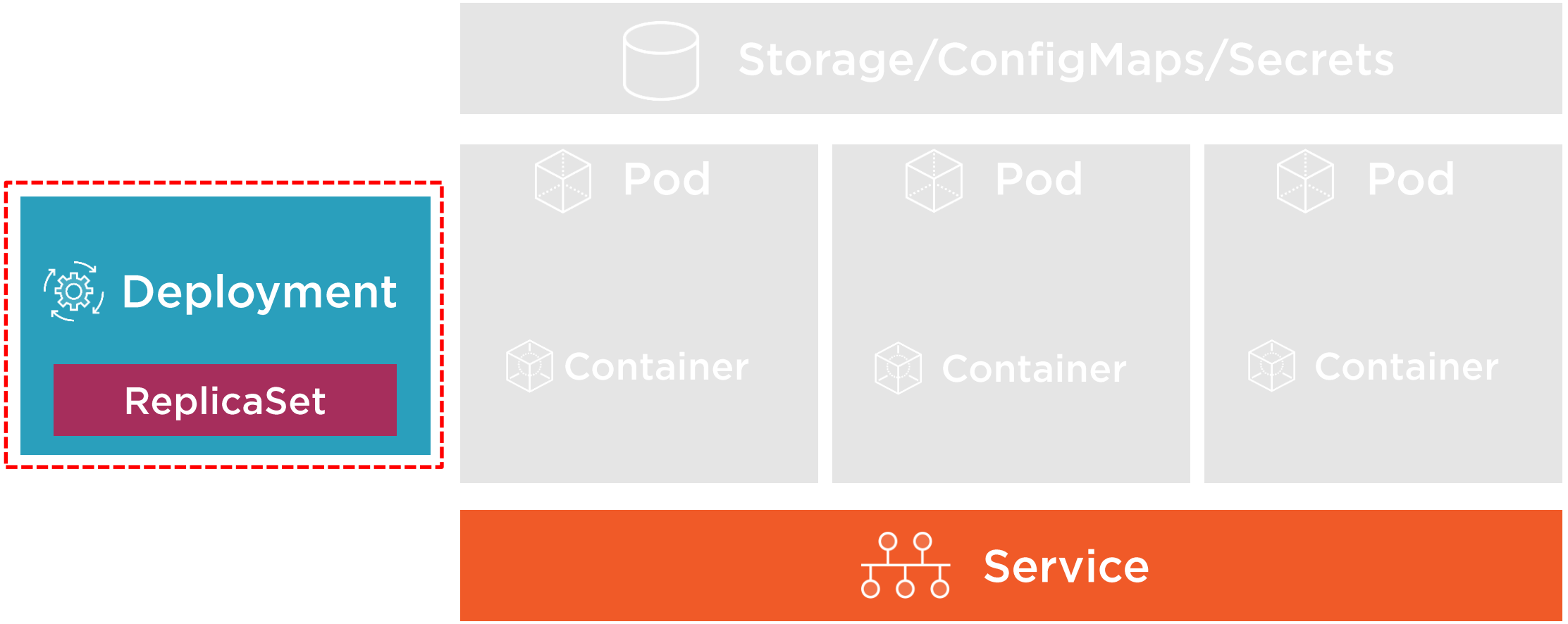
Creating a Canary Deployment

Canary Deployments in Action

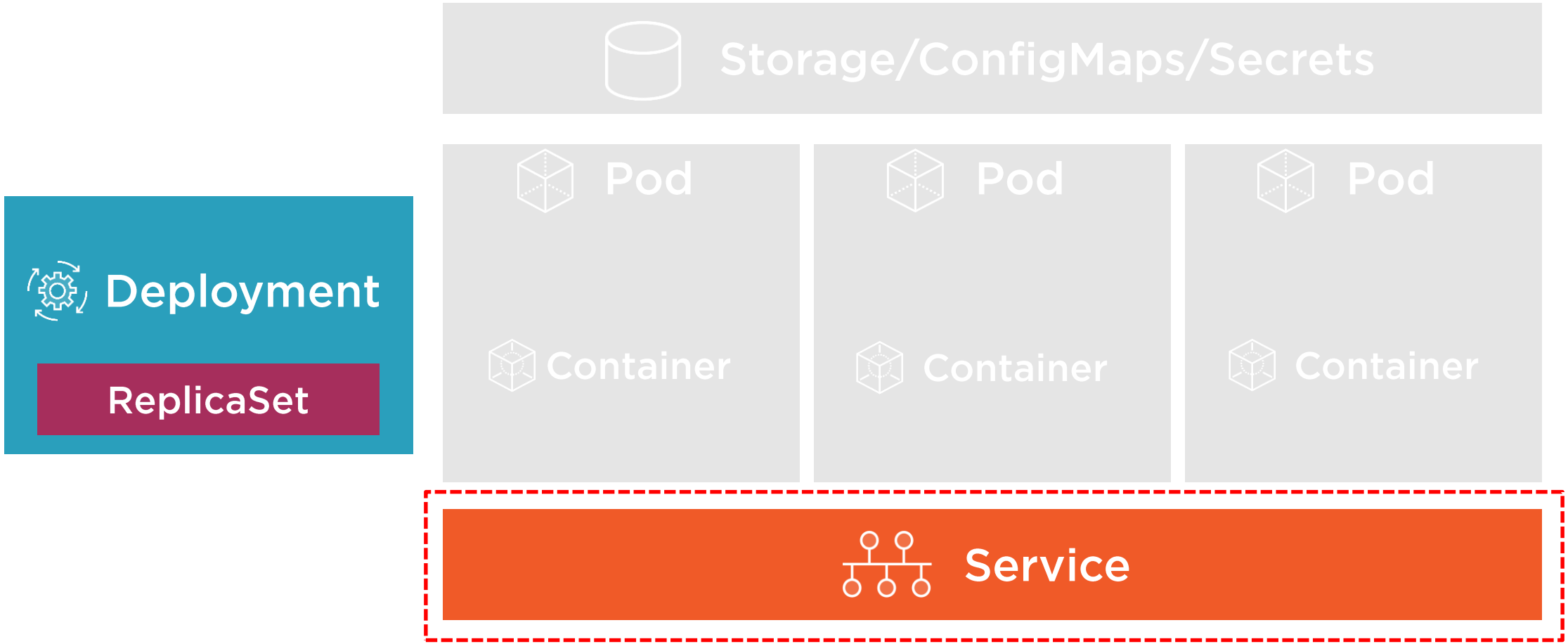
Kubernetes Resources



Kubernetes Resources



Kubernetes Resources



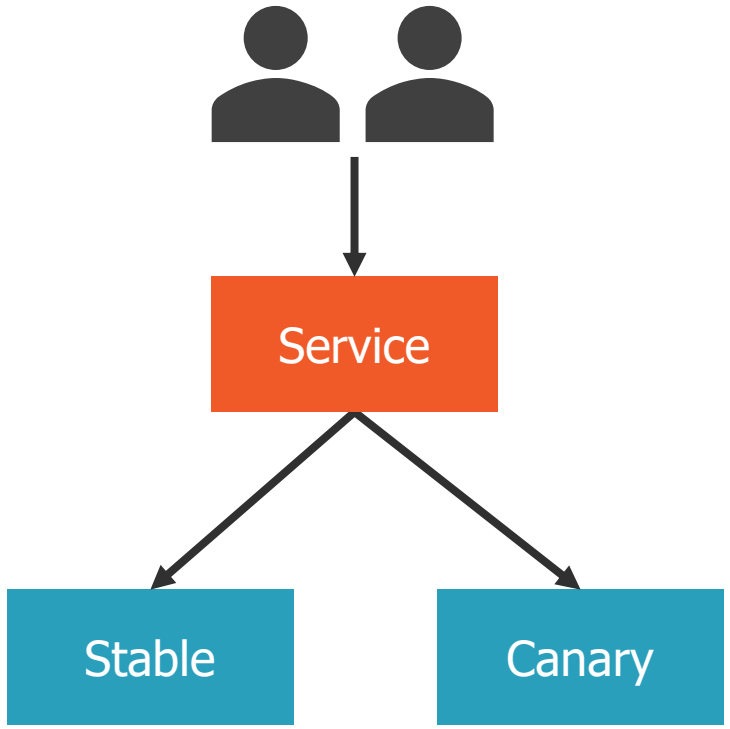
Understanding Canary Deployments

Wouldn't it be nice to rollout a new Deployment but only route a small percentage of the overall traffic to it to ensure it's working properly?

"Canary deployment strategy involves deploying new versions of applications next to stable production versions to see how the canary version compares against the baseline before promoting or rejecting the deployment."

~ https://docs.microsoft.com

Canary Deployments



Strategy for checking the viability of a deployment

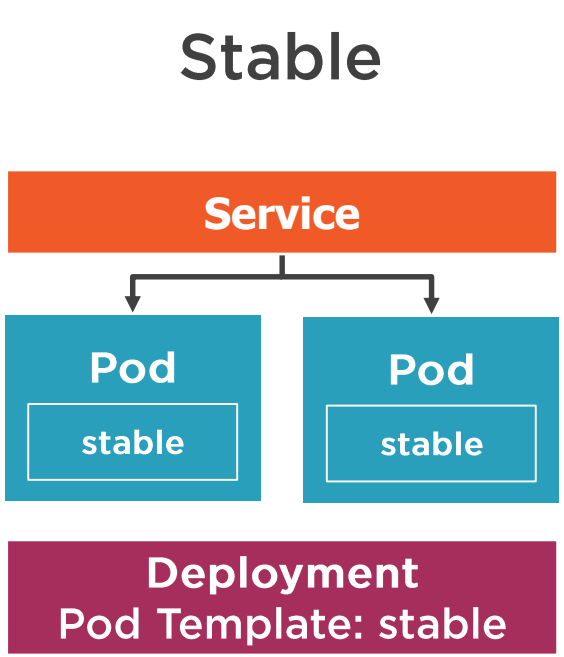
Run two identical production environments at the same time

Canary Deployment runs alongside the existing stable Deployment

Canary Deployment is setup to receive minimal traffic

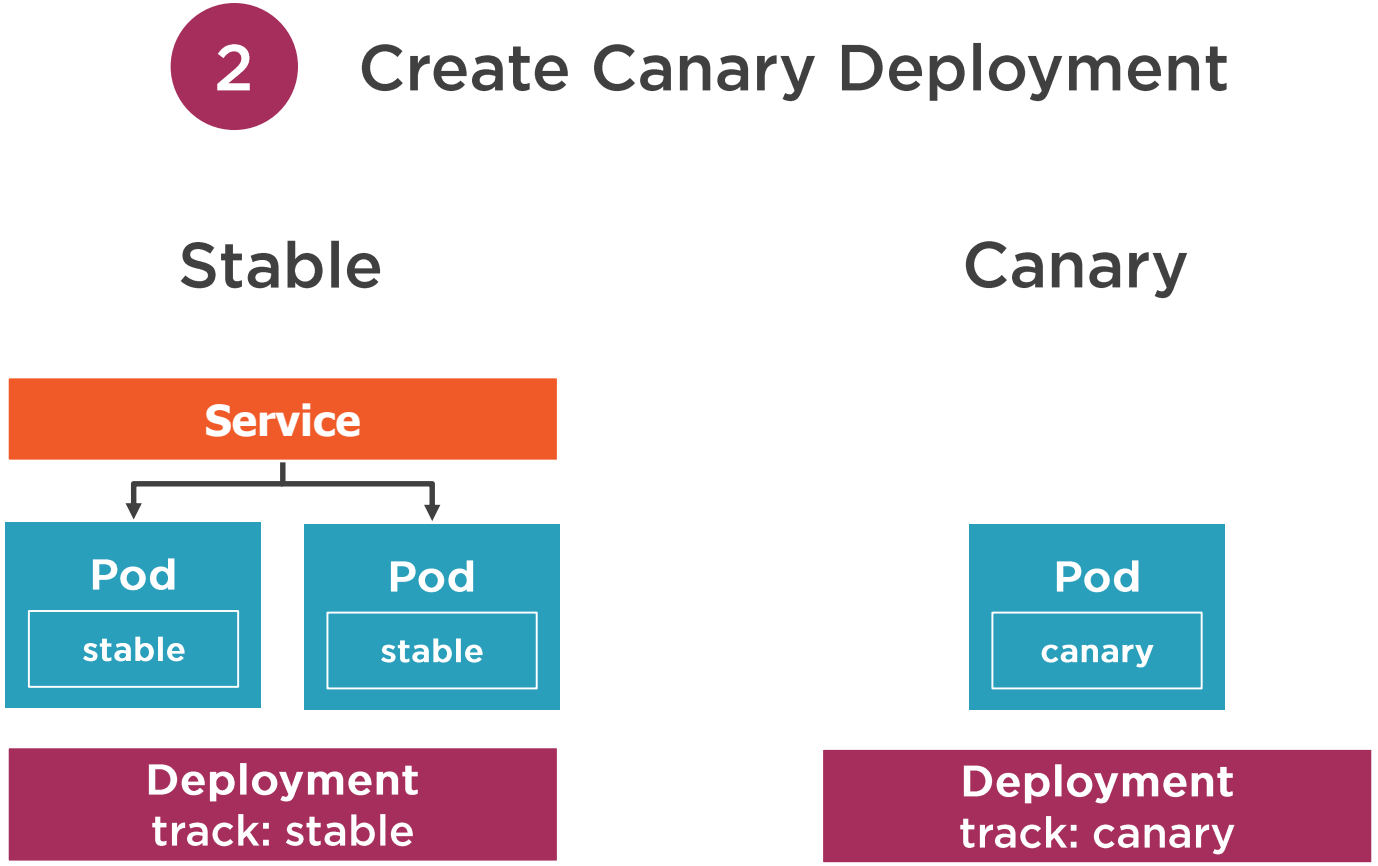
Canary Deployments

1 **Create Stable Deployment and Service**



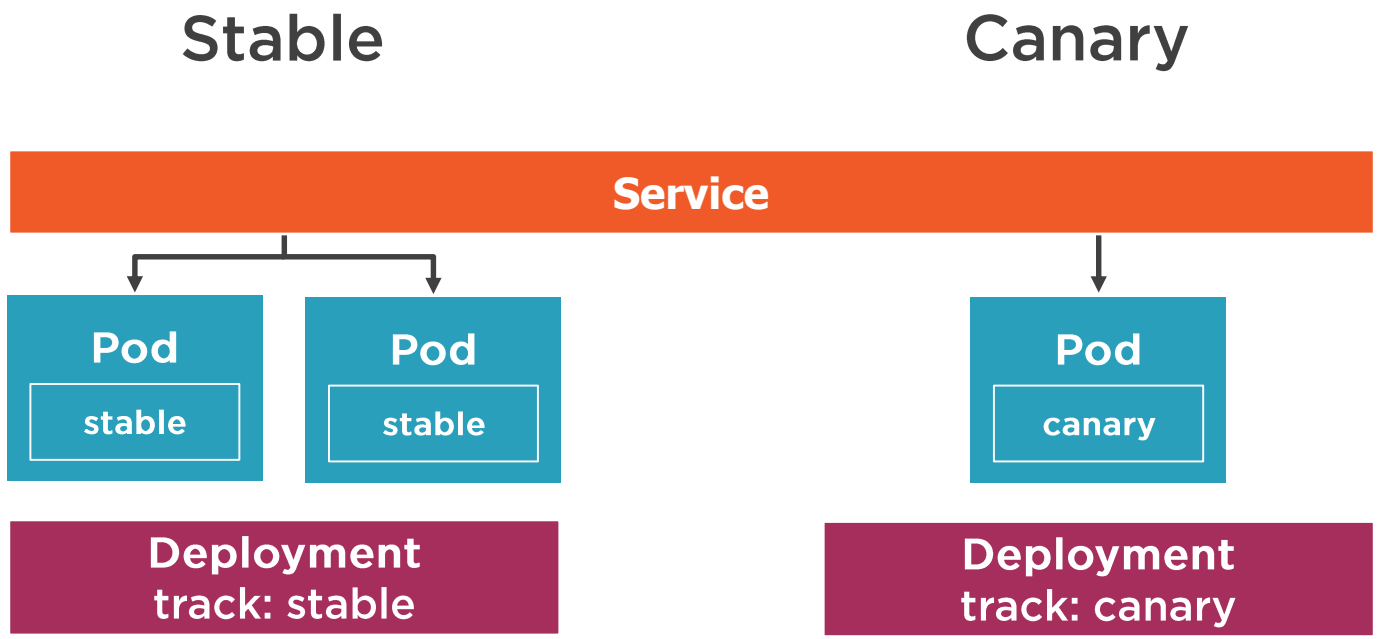
Canary Deployments

2 Create Canary Deployment



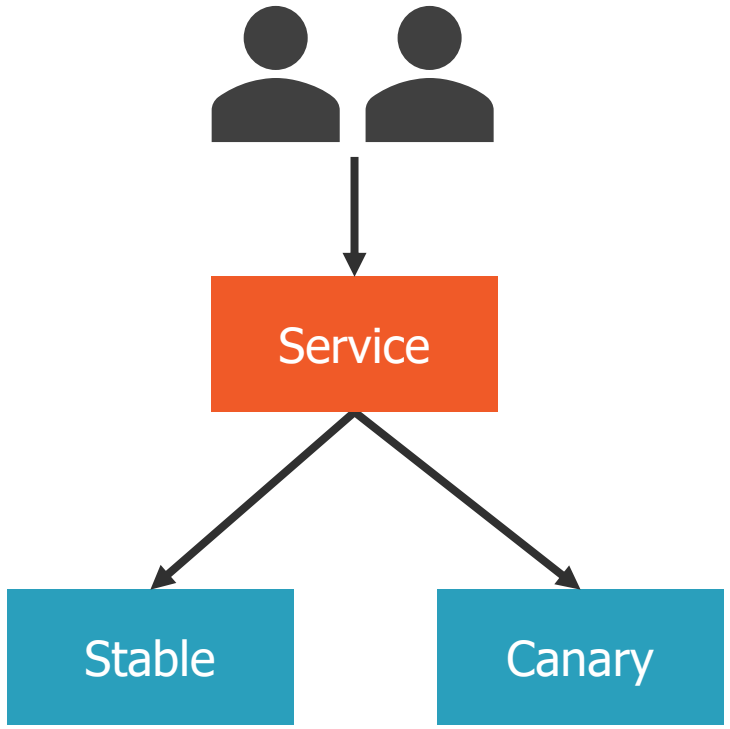
Canary Deployments

3 Service adds Canary Pod(s) and traffic is routed



Creating a Canary Deployment

Canary Deployment Resources

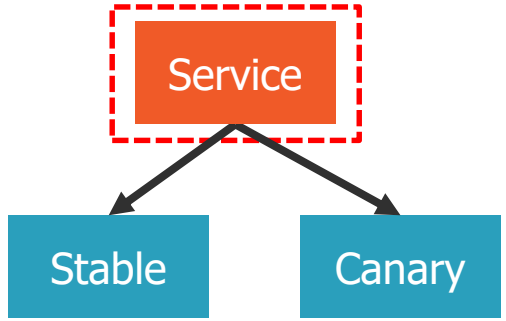


A Canary Deployment involves 3 main Kubernetes resources:

- Service
- Stable Deployment
- Canary Deployment

Defining a Service

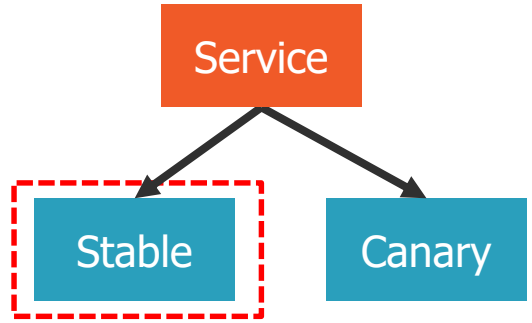
```
kind: Service
apiVersion: v1
metadata:
  name: stable-service
  labels:
    app: aspnetcore
spec:
  type: LoadBalancer
  selector:
    app: aspnetcore
  ports:
    - port: 80
      targetPort: 80
```



◂ Pod Label to select for Service

Defining a Stable Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: stable-deployment
spec:
  replicas: 4
  selector:
    matchLabels:
      app: aspnetcore
      track: stable
  template:
    metadata:
      labels:
        app: aspnetcore
        track: stable
    spec:
      containers:
      - name: stable-app
        image: stable-app
```

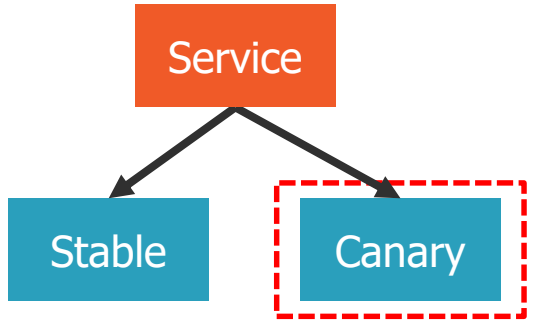


- Create stable replicas

- Pod labels (recall that **app:aspnetcore** is targeted by the Service)

Defining a Canary Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: canary-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aspnetcore
      track: canary
  template:
    metadata:
      labels:
        app: aspnetcore
        track: canary
    spec:
      containers:
      - name: canary-app
        image: canary-app
```



- Create canary replicas (25% of stable in this example)

- Pod labels (recall that **app:aspnetcore** is targeted by the Service)

Creating the Stable and Canary Resources

Use kubectl create or kubectl apply commands to create the Service, Stable Deployment, and Canary Deployment

```
# Create Service, Stable Deployment, and Canary Deployment
kubectl create –f [folder-name] --save-config --record
```

Canary Deployments in Action

Summary

Canary Deployments allows a new version to be deployed next to a stable version

Configured to only handle a small percentage of the traffic initially

Once the Canary Deployment is verified it can be scaled up and the existing stable Deployment can be scaled down