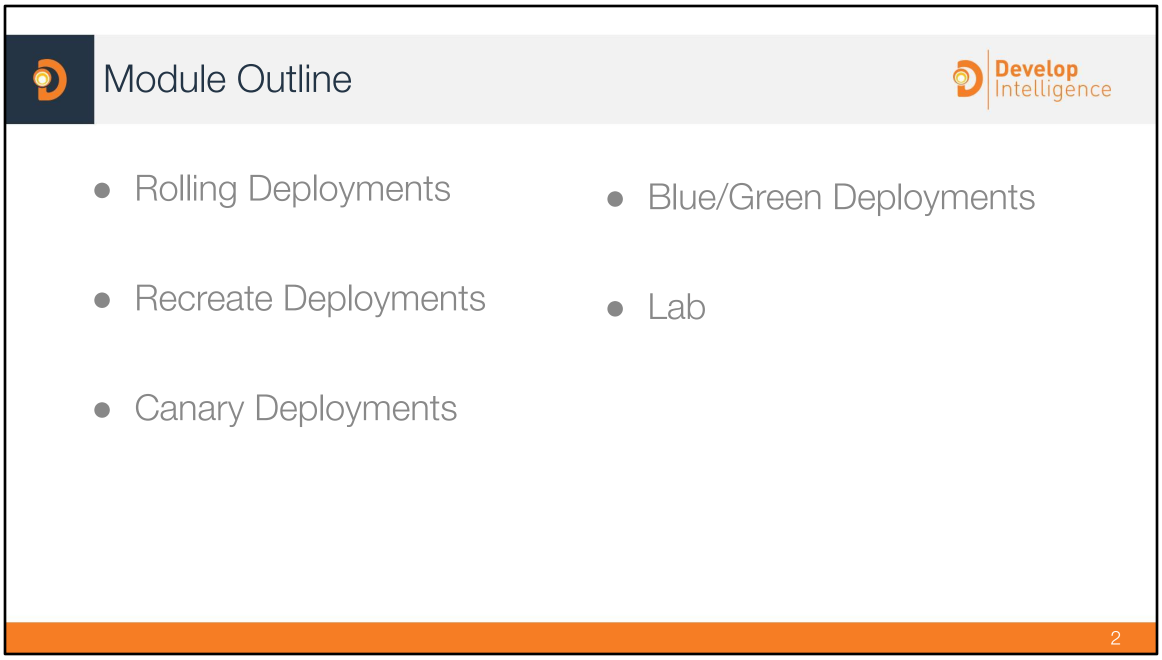
Deploying Apps on K8s

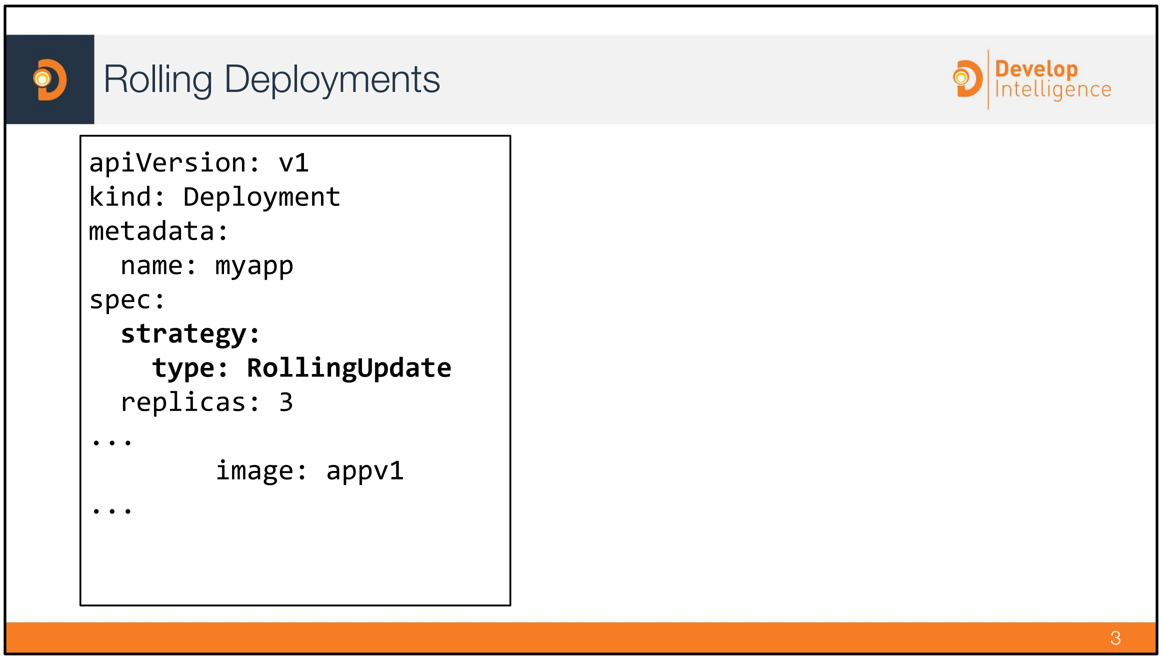


Module Outline

Develop Intelligence

- Rolling Deployments
- Recreate Deployments
- Canary Deployments
- Blue/Green Deployments
- Lab

2



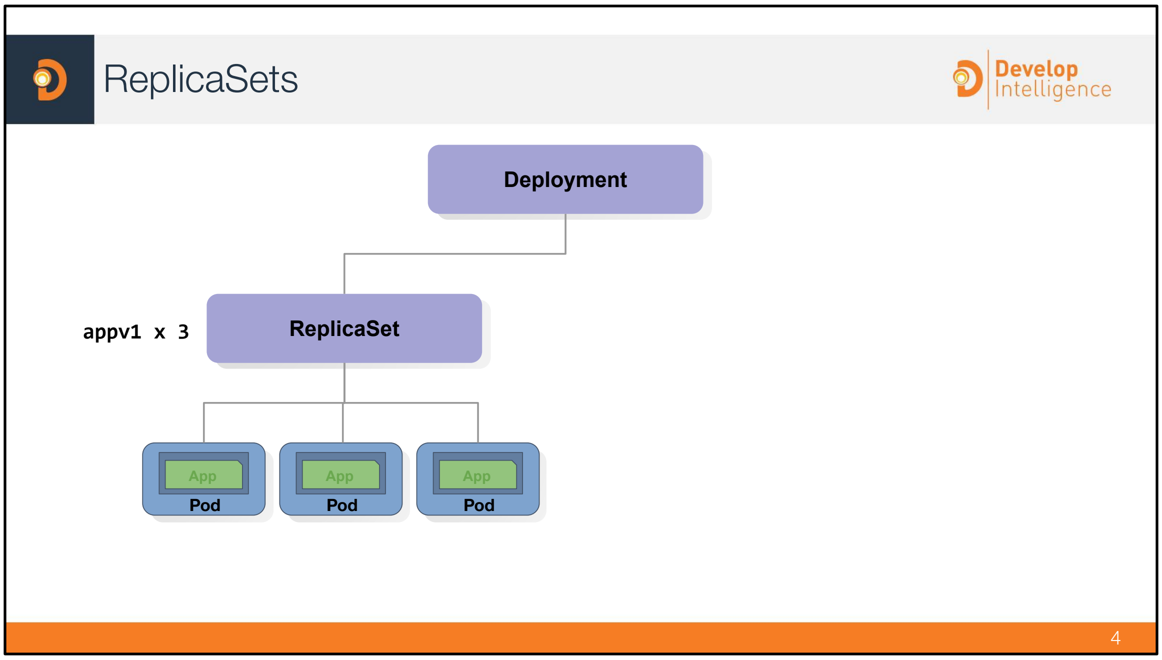
Rolling Deployments

Develop Intelligence

```
apiVersion: v1
kind: Deployment
metadata:
  name: myapp
spec:
  strategy:
    type: RollingUpdate
  replicas: 3
...
        image: appv1
...
```

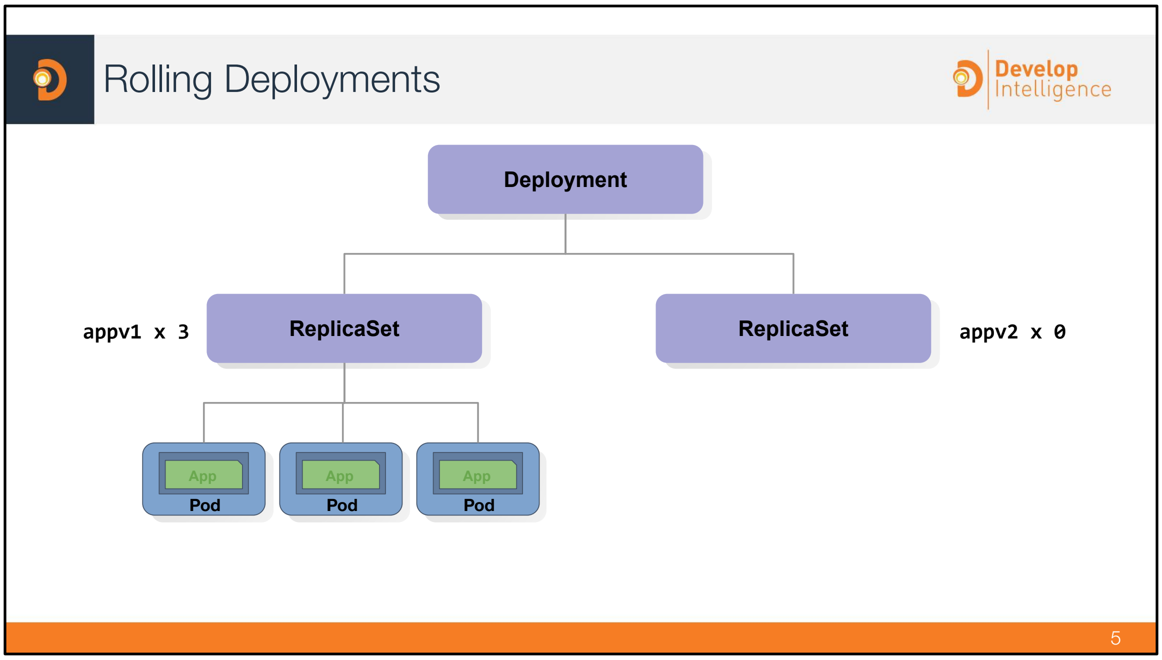
3

Rolling updates allow you to gradually update your application to the new version alongside the old version running. This means we can avoid downtime for our overall application and not impact users. Deployment objects use rolling updates by default. Whenever we change something about our container, like having a new image, the deployment will one by one, bring up a Pod of our new version and bring down a pod of the old version until everything is at the new version.

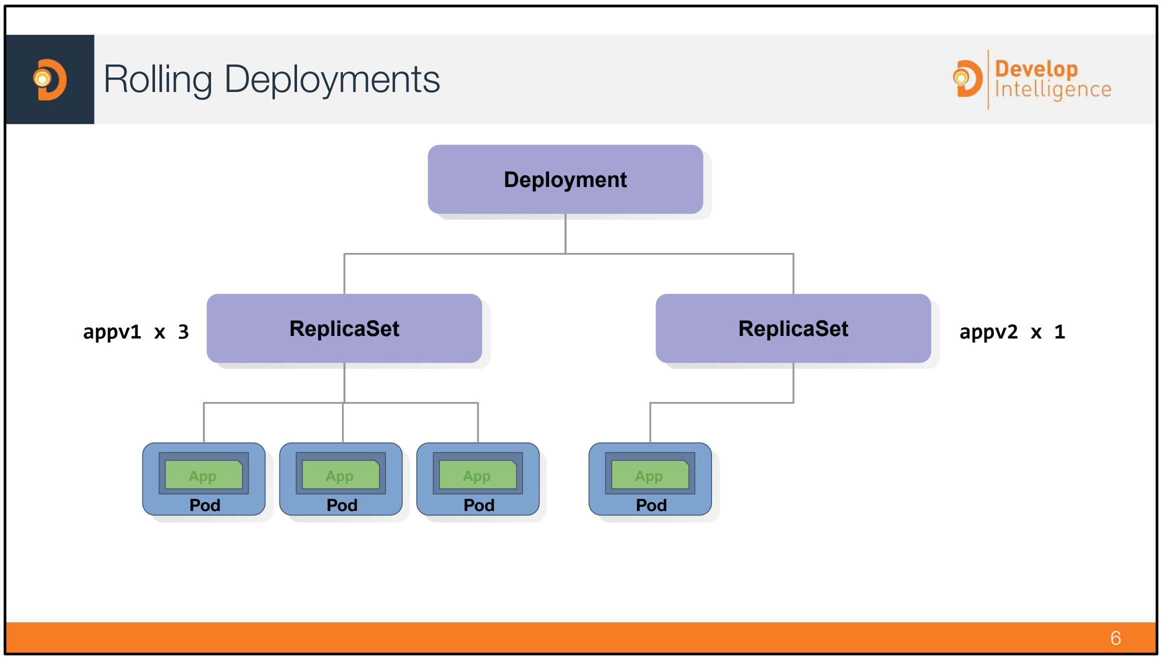


ReplicaSets

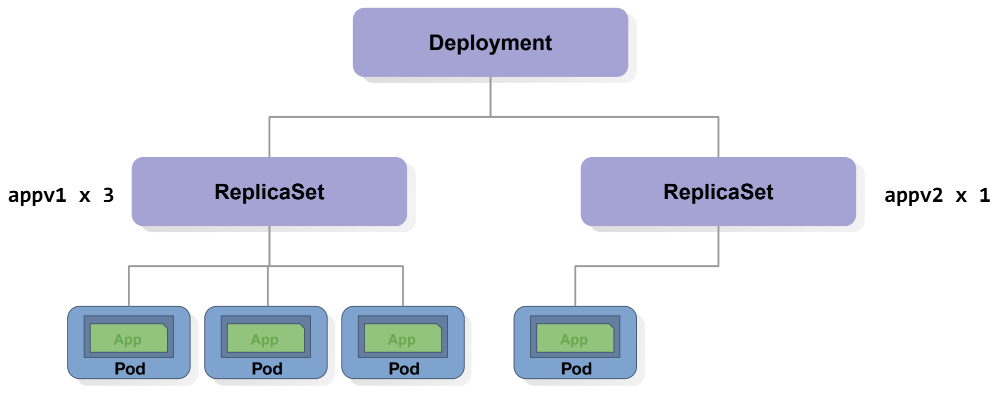
Deployments handle rolling updates by creating and deleting ReplicaSets. ReplicaSets are the underlying objects that control the number of Pod replicas, it's not actually the deployment object itself. This allows a Deployment object to create 2 ReplicaSets, one of the new version and one of the old. It will increment the number of replicas on the new ReplicaSet and decrement the number on the old.

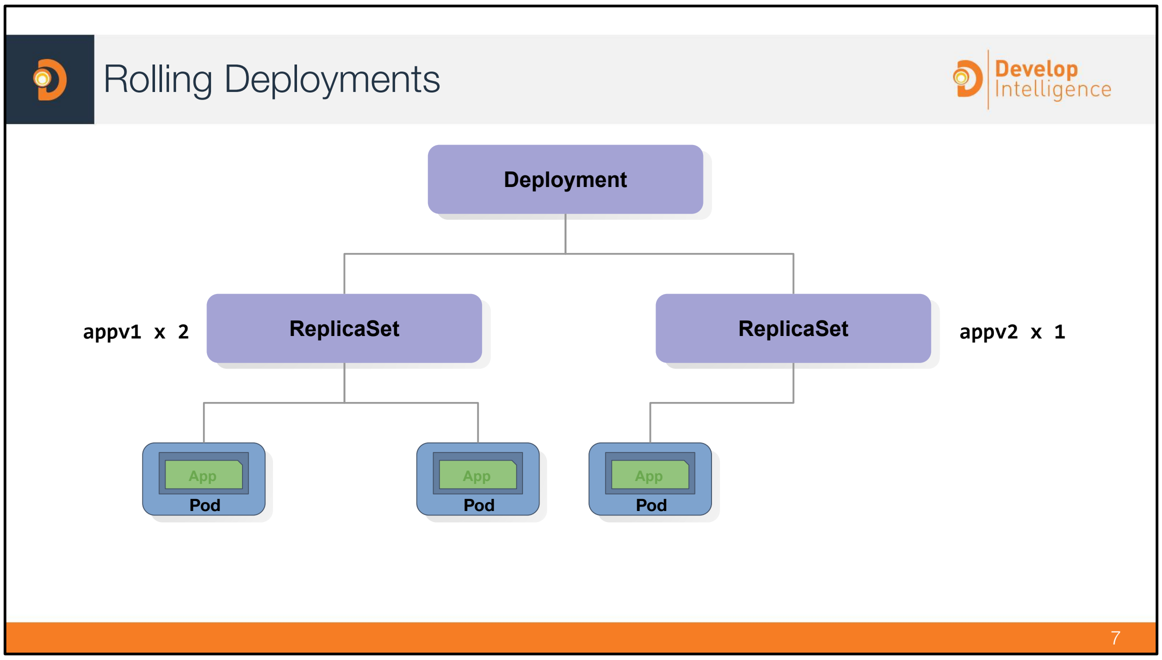


Rolling Deployments

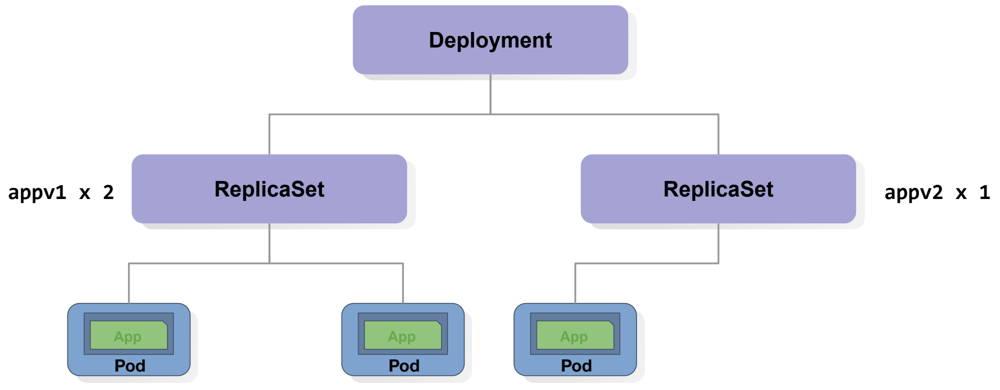


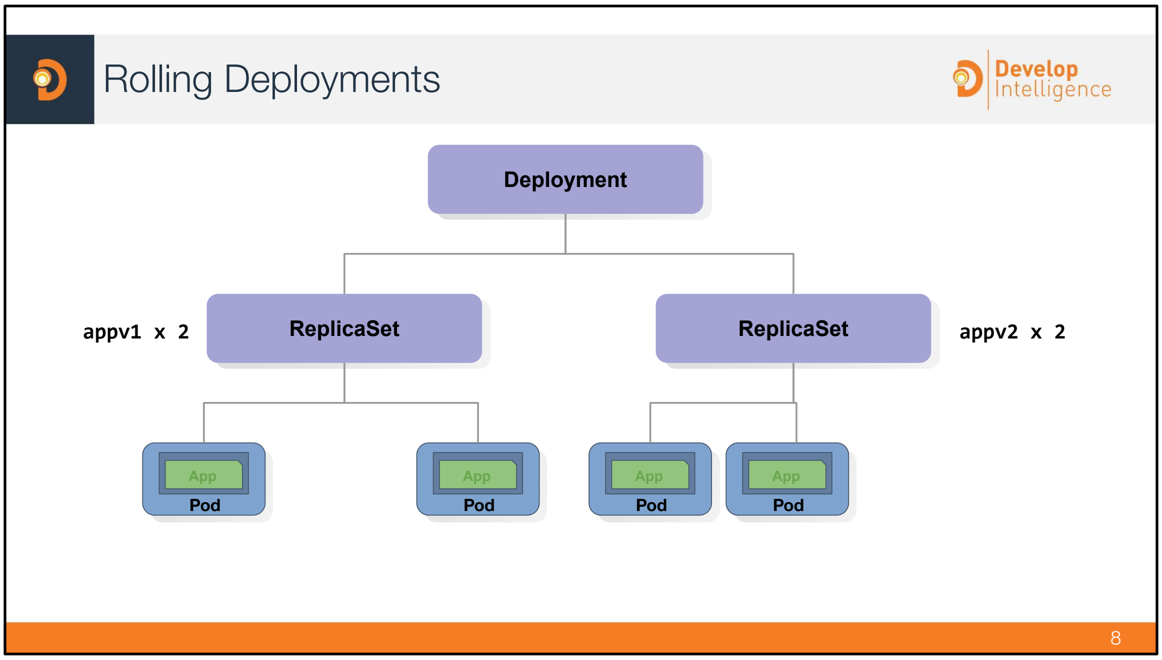
Rolling Deployments





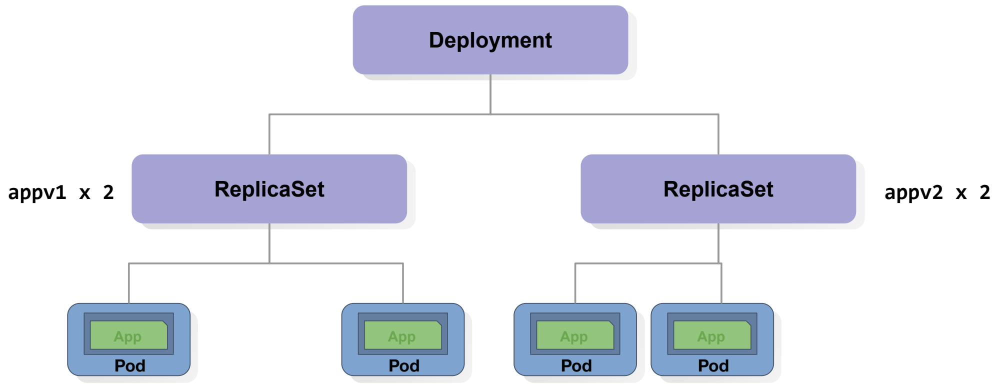
Rolling Deployments





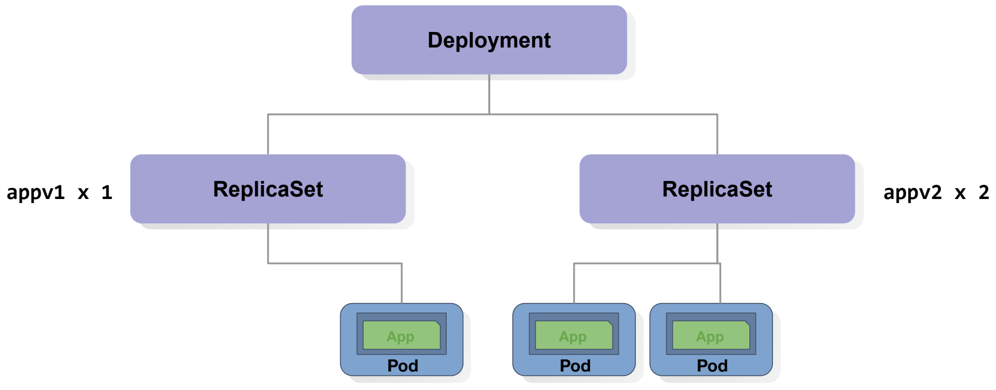
Rolling Deployments

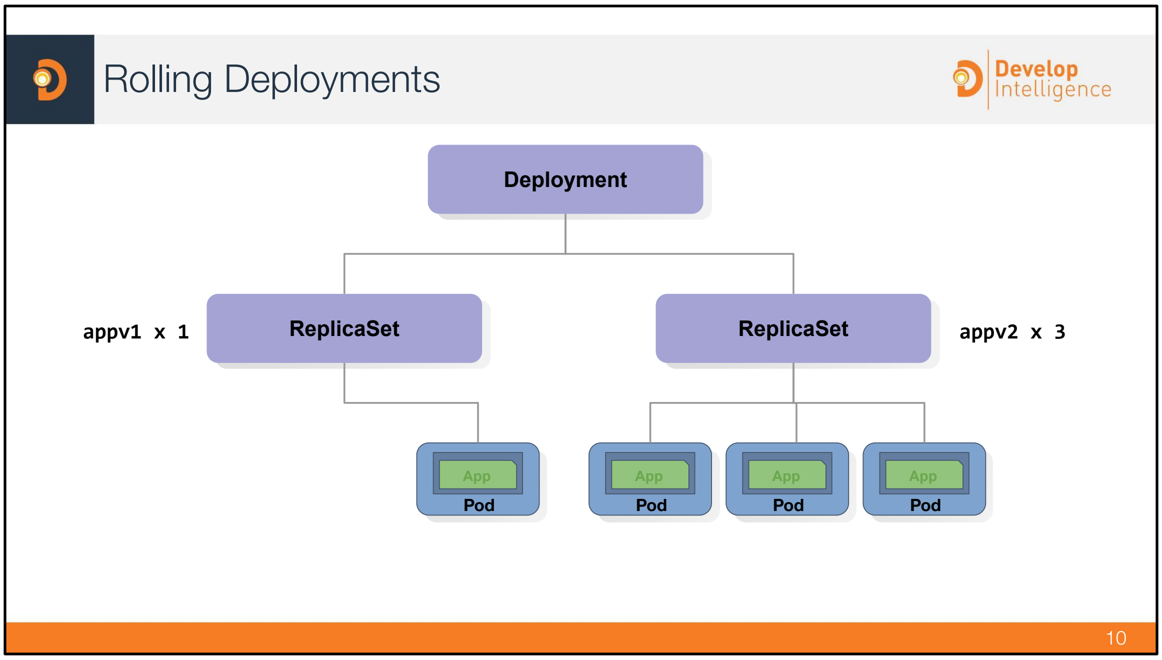
Develop Intelligence



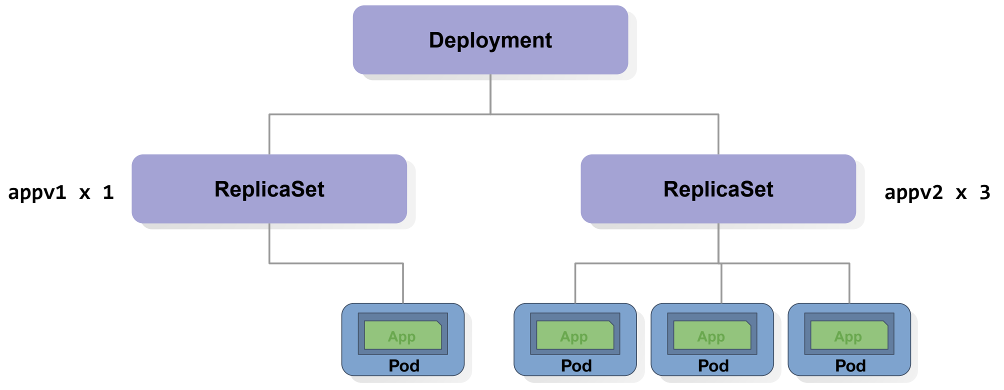
8

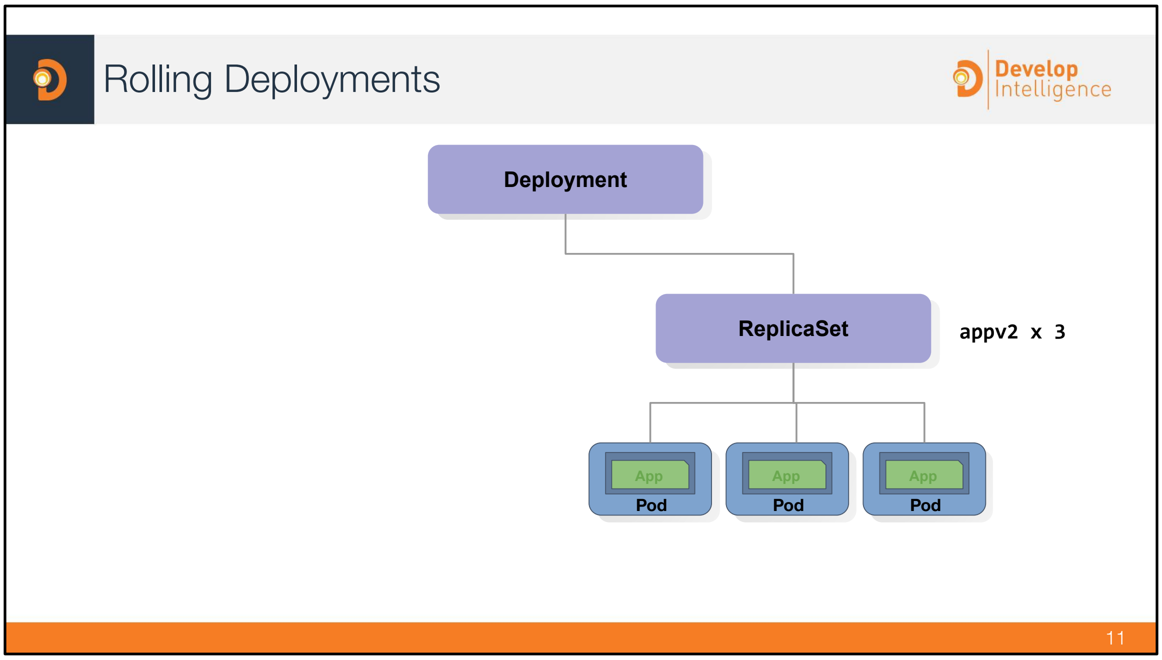
Rolling Deployments





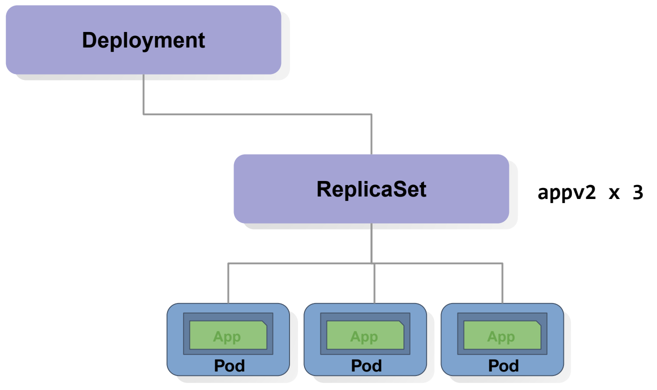
Rolling Deployments



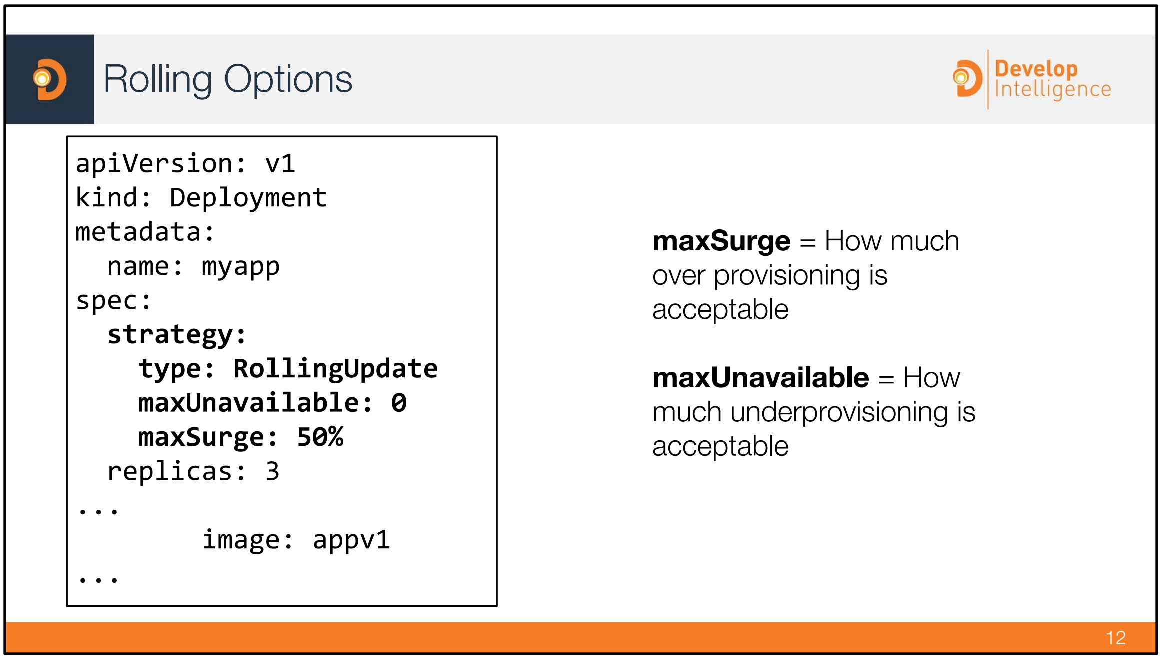


Rolling Deployments

Develop Intelligence



11



Rolling Options

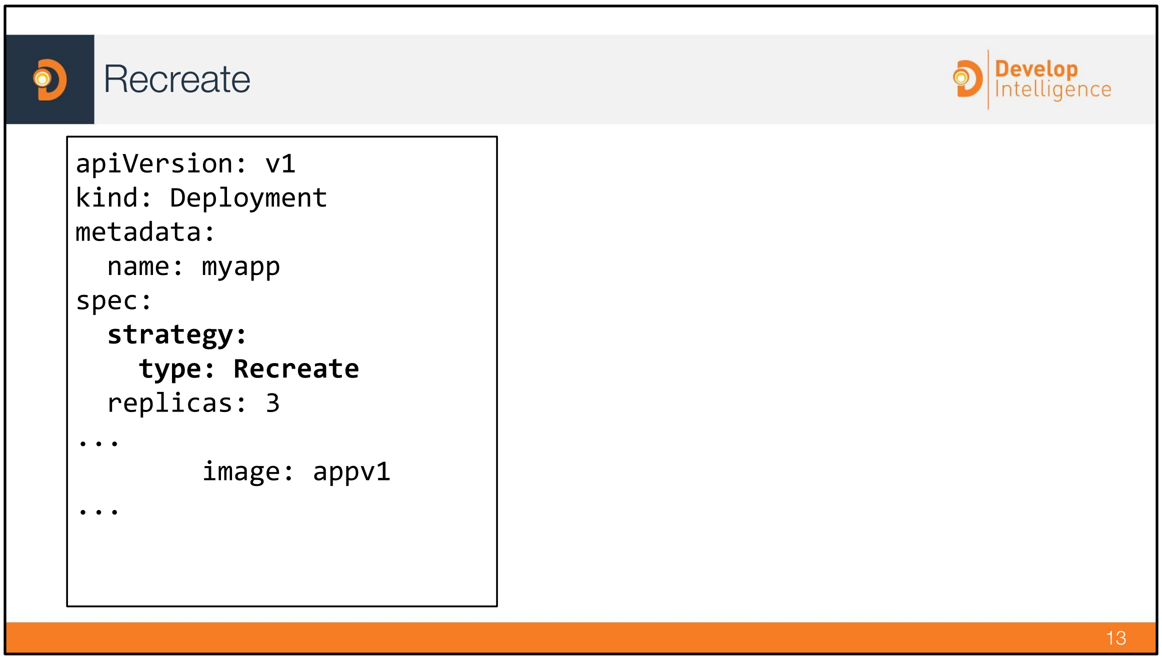
```
apiVersion: v1
kind: Deployment
metadata:
  name: myapp
spec:
  strategy:
    type: RollingUpdate
    maxUnavailable: 0
    maxSurge: 50%
  replicas: 3
...
        image: appv1
...
```

maxSurge = How much over provisioning is acceptable

maxUnavailable = How much underprovisioning is acceptable

12

We just saw a max surge of 1. We could instead have it move faster through this process by spinning up 2 or 3 or N pods of the new version at a time. The MaxSurge and maxUnavailable options on the deployment give us this control.

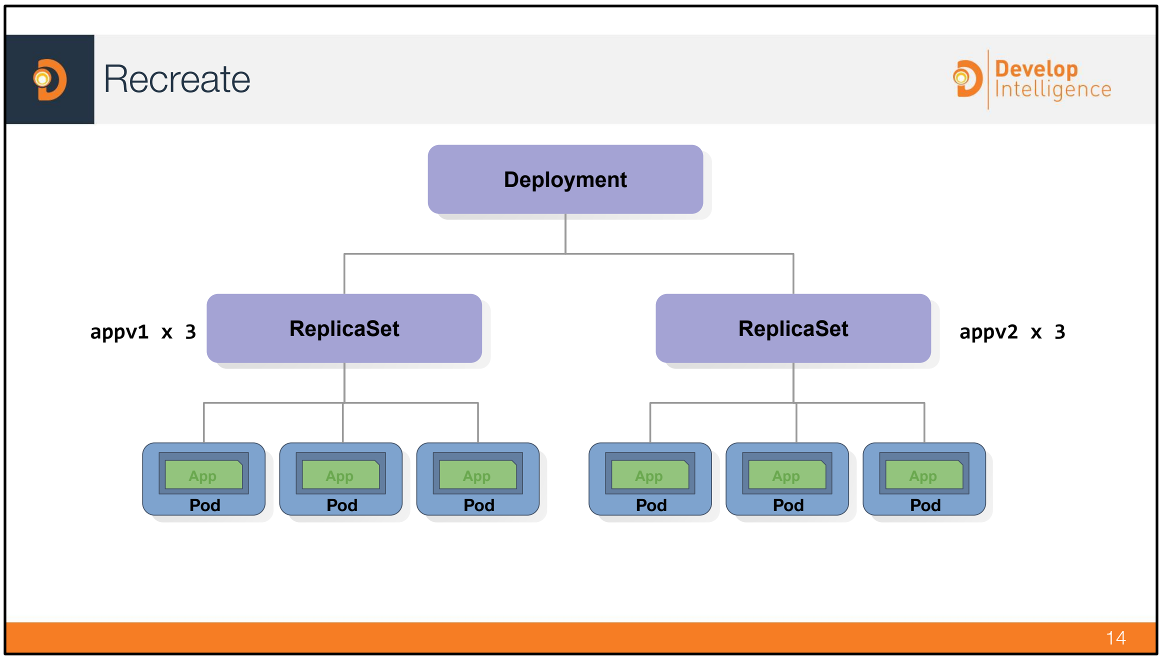


Recreate

```
apiVersion: v1
kind: Deployment
metadata:
  name: myapp
spec:
  strategy:
    type: Recreate
  replicas: 3
...
        image: appv1
...
```

13

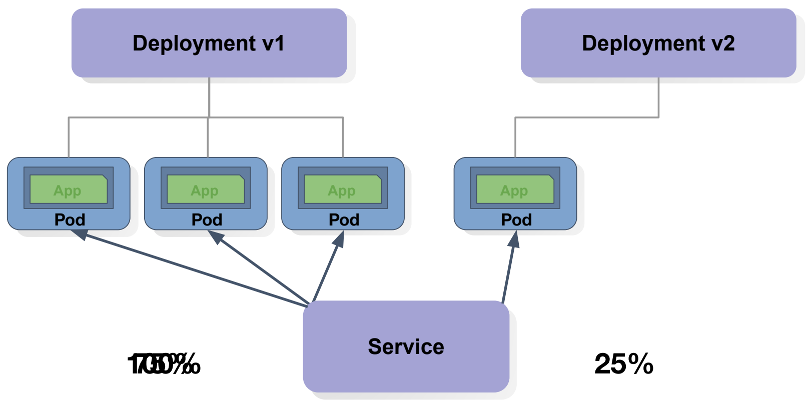
If our application cannot support both version operating simultaneously, or if we want a consistent experience by our users at the cost of downtime, we can instead use the Recreate deployment strategy.



Recreate

This will simply delete all existing Pods and then create all of the Pods of the new version. Your overall approach may be a combination of deployment strategies. Where you try to do rolling whenever possible but in the case of, say, a database schema update, you opt for recreate as the old version of the app may not understand the new database.

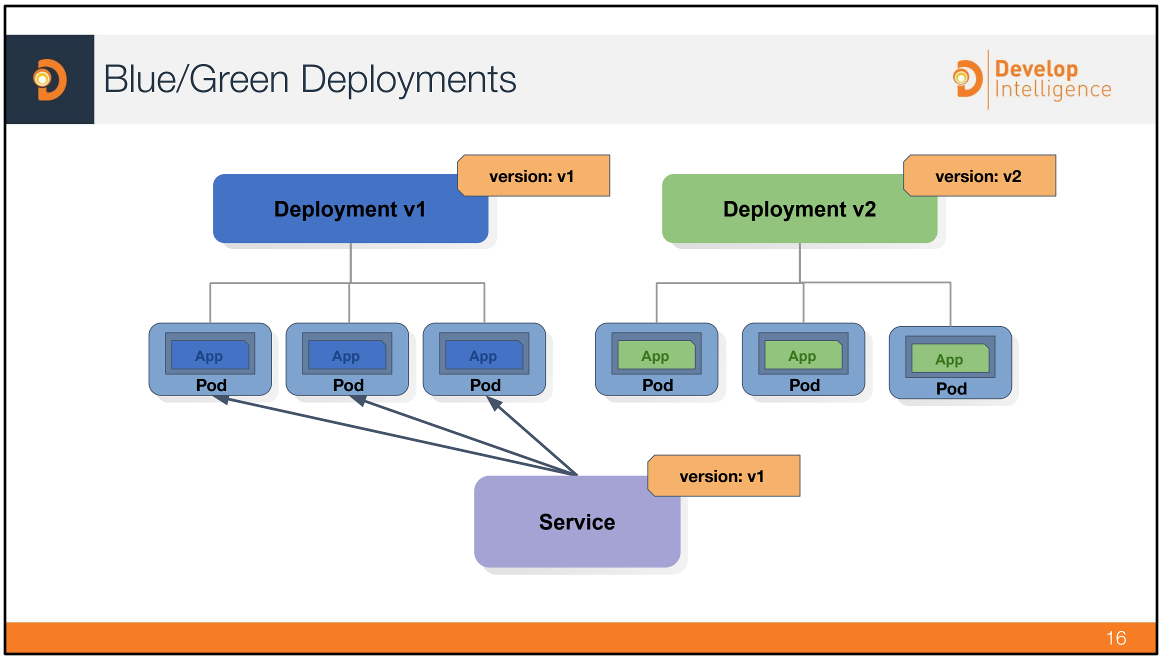
Canary Deployments



15

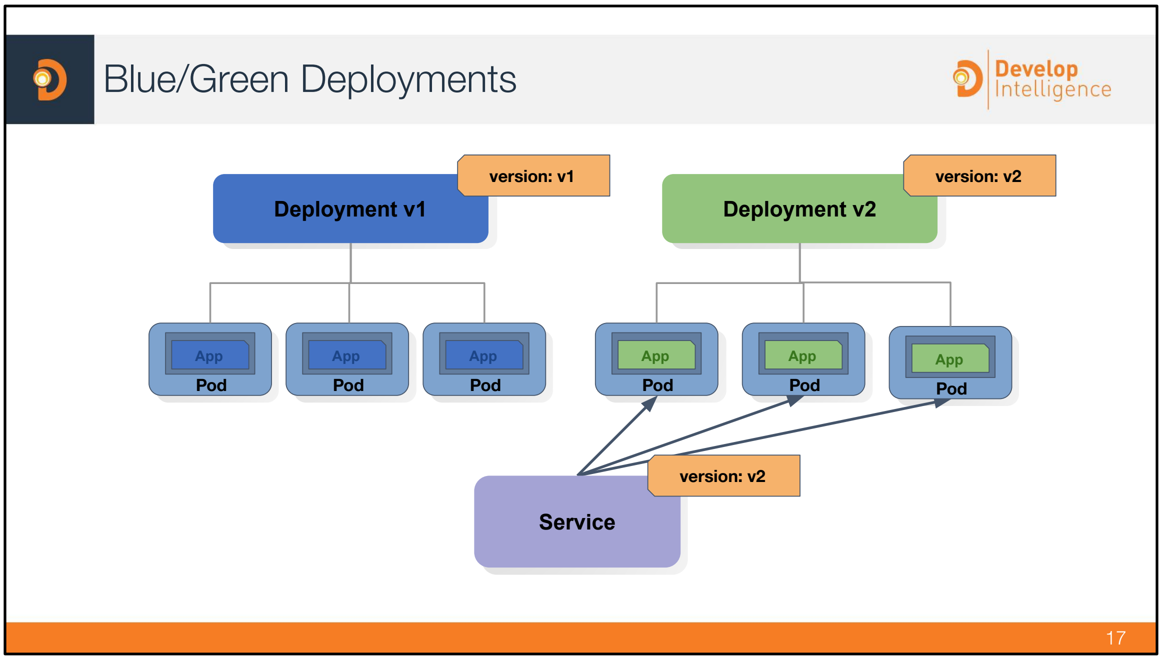
You may find value in testing your new version against real user traffic before updating everything. Canary deployments allow you to only have a percentage of user traffic go to the new version. (like a canary in a coal mine) This will allow you to vet the new version with scenarios you may not have been able to predict, and if there is a problem, it only affects a small subset of your users.

One way to do canary deployments is to create a new Deployment object for the new version of the app, say with just 1 pod. Have the service point to Pods of both deployments and when you are happy, just scale up the new version then delete the old Deployment. There are additional tools you can use on top of kubernetes to have finer grained control over splitting traffic for canary deployments, say for example you only want users named "jason" to hit the new version before you move everyone onto it.Or or you want exactly 5% of your users to hit the new version until you decide it's good.



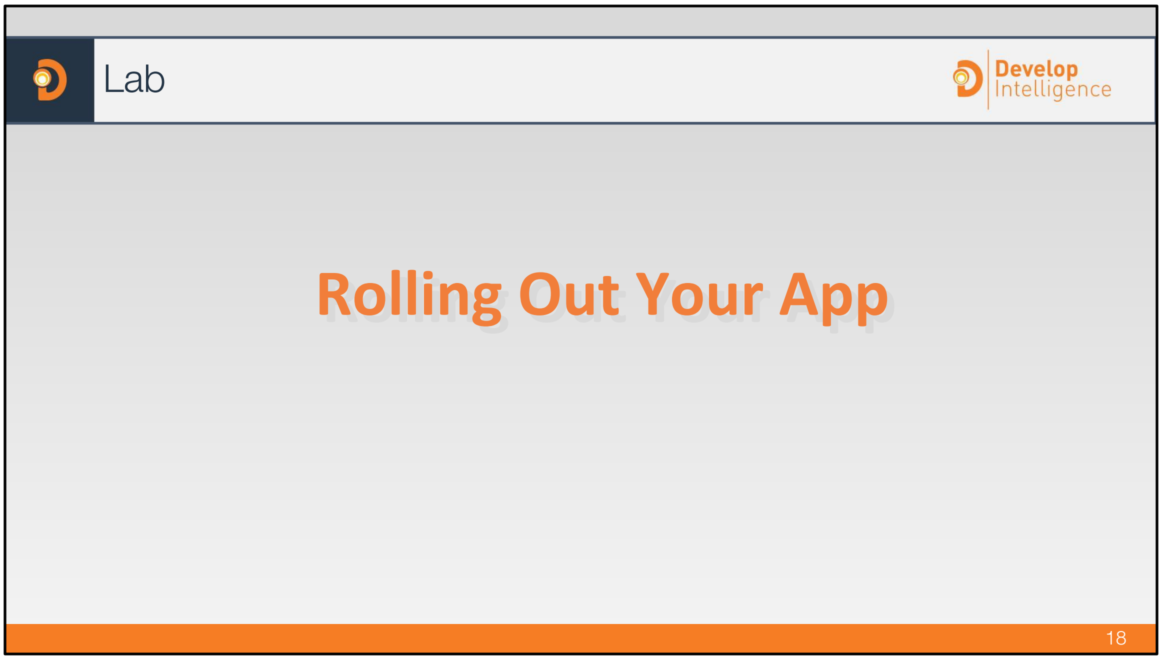
Blue/Green Deployments

Blue/Green deployments are an approach where you complete roll out the new version separately, then when you are ready, you flip over to sending all users to the new version. THis allows you to control when the deployment occurs but minimize downtime. This is done by creating a new deployment for the new version and updating a Service label selector to point to the new version instead of the old.



Blue/Green Deployments

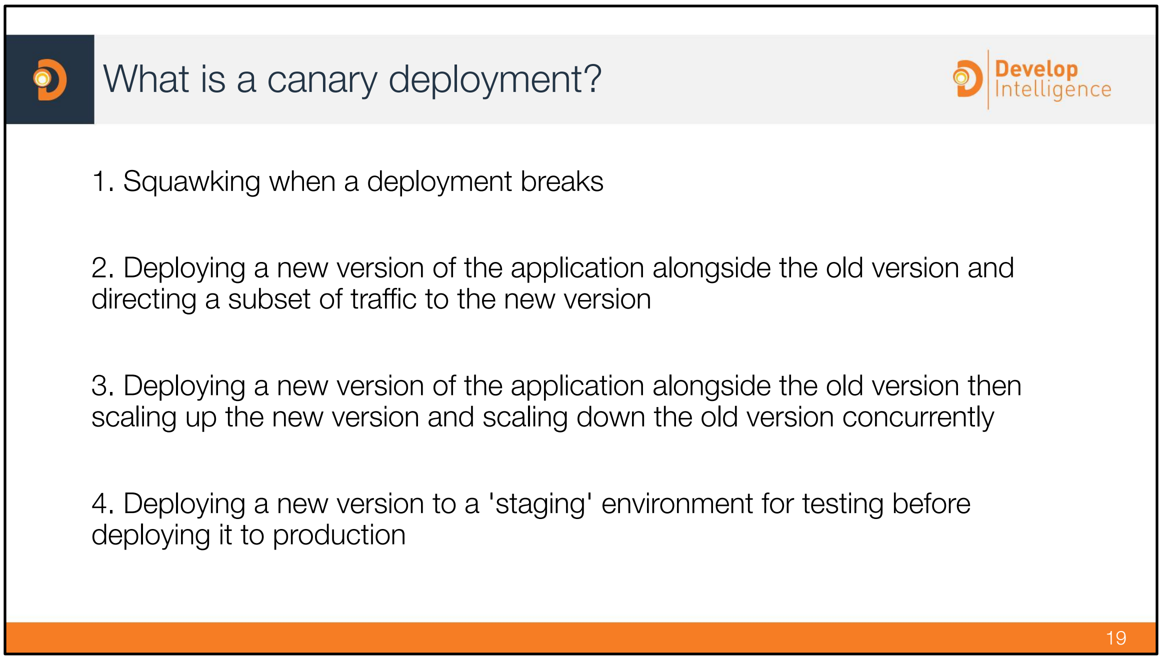
Blue/Green deployments are an approach where you complete roll out the new version separately, then when you are ready, you flip over to sending all users to the new version. THis allows you to control when the deployment occurs but minimize downtime. This is done by creating a new deployment for the new version and updating a Service label selector to point to the new version instead of the old.



Lab

Develop Intelligence

Rolling Out Your App



What is a canary deployment?

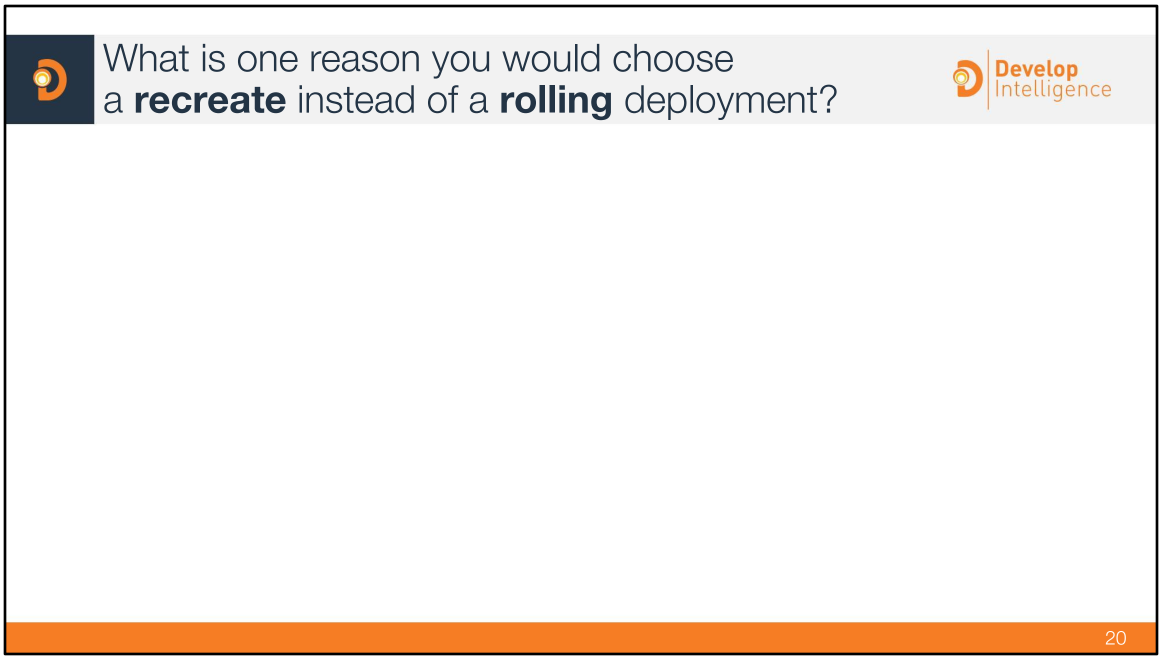
Develop Intelligence

1. Squawking when a deployment breaks

2. Deploying a new version of the application alongside the old version and directing a subset of traffic to the new version

3. Deploying a new version of the application alongside the old version then scaling up the new version and scaling down the old version concurrently

4. Deploying a new version to a 'staging' environment for testing before deploying it to production



What is one reason you would choose a **recreate** instead of a **rolling** deployment?

Develop Intelligence

20

Synchronized deployment with a dependent API (backward incompatible change)