Deploying to Kubernetes

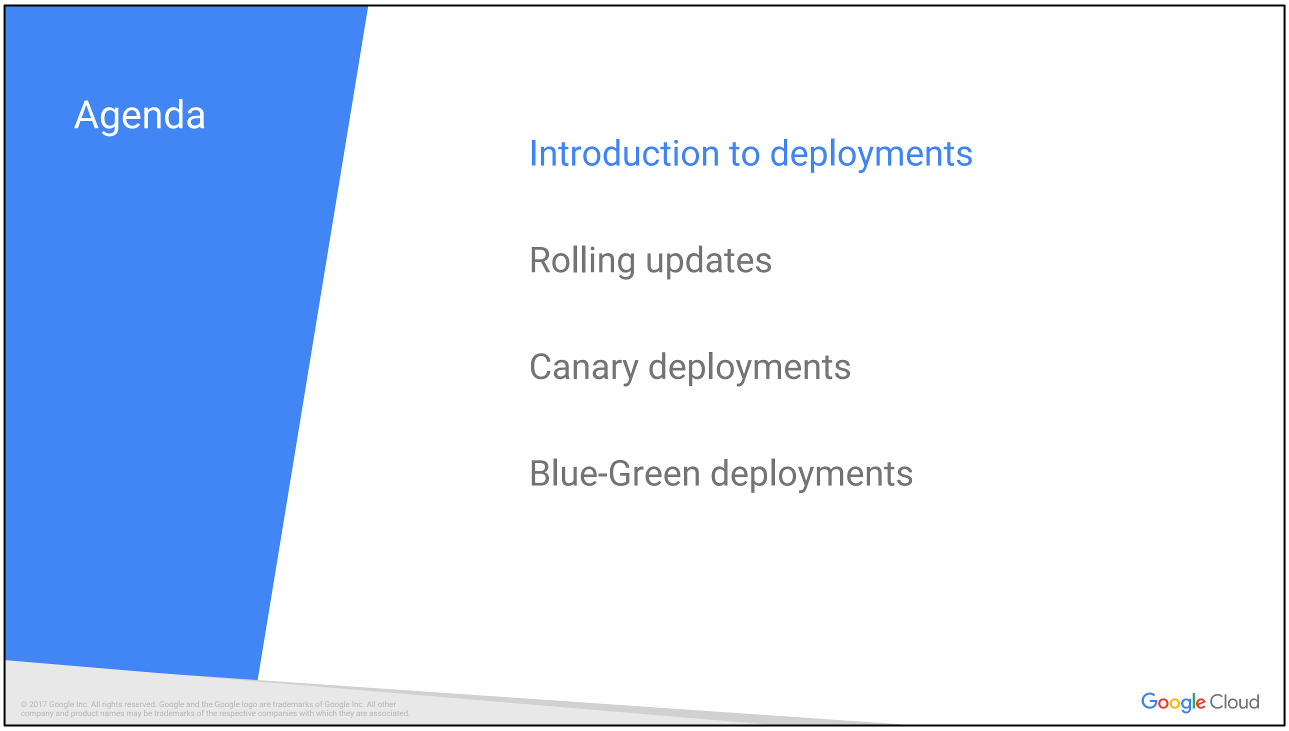
Getting Started With Google Kubernetes Engine

Version 1.5

Google Cloud

[Duration: 20 mins]

Now we're going to go through how you can deploy pods and orchestrate pods in Kubernetes.



Agenda

Introduction to deployments

Rolling updates

Canary deployments

Blue-Green deployments

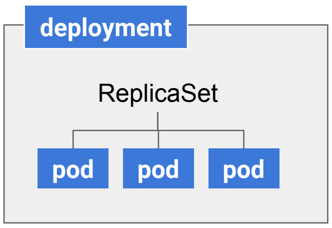
Deployments allow you to name a set of pods and ensure that the number and state of pods running is equal to the desired number and state of pods specified by the user.

Deployments rely on ReplicaSets to manage and run pods

Deployment
- name: hello

ReplicaSet
- replicas: 3
- selector:
  - app: hello

Pod
- containers:
  - image: hello1



Google Cloud

Behind the scenes, a deployment relies on a **ReplicaSet** to manage and run a given number of pods at a given time.

In this example, there is a Deployment named **hello**.

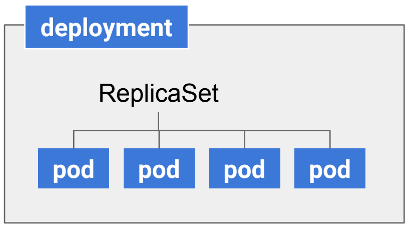
When you create that deployment, it's going to create a ReplicaSet of size 3.

You add the label selector of **app: hello**.

Inside of the pod, you have a single image called **hello1**.

Deployments monitor your cluster and make changes

ReplicaSet
- replicas: 4
- selector:
  - app: hello



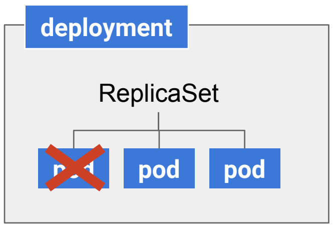
Google Cloud

A deployment monitors your cluster to see whether anything is different from what you defined. If there's anything different the deployment tries to rectify it.

So for example, if in the API you created a deployment with replicas set to 4, and only 3 replicas are running, the deployment detects that difference between what you defined in the API and what's running in the cluster and tries to rectify it by running another pod somewhere else in the cluster.

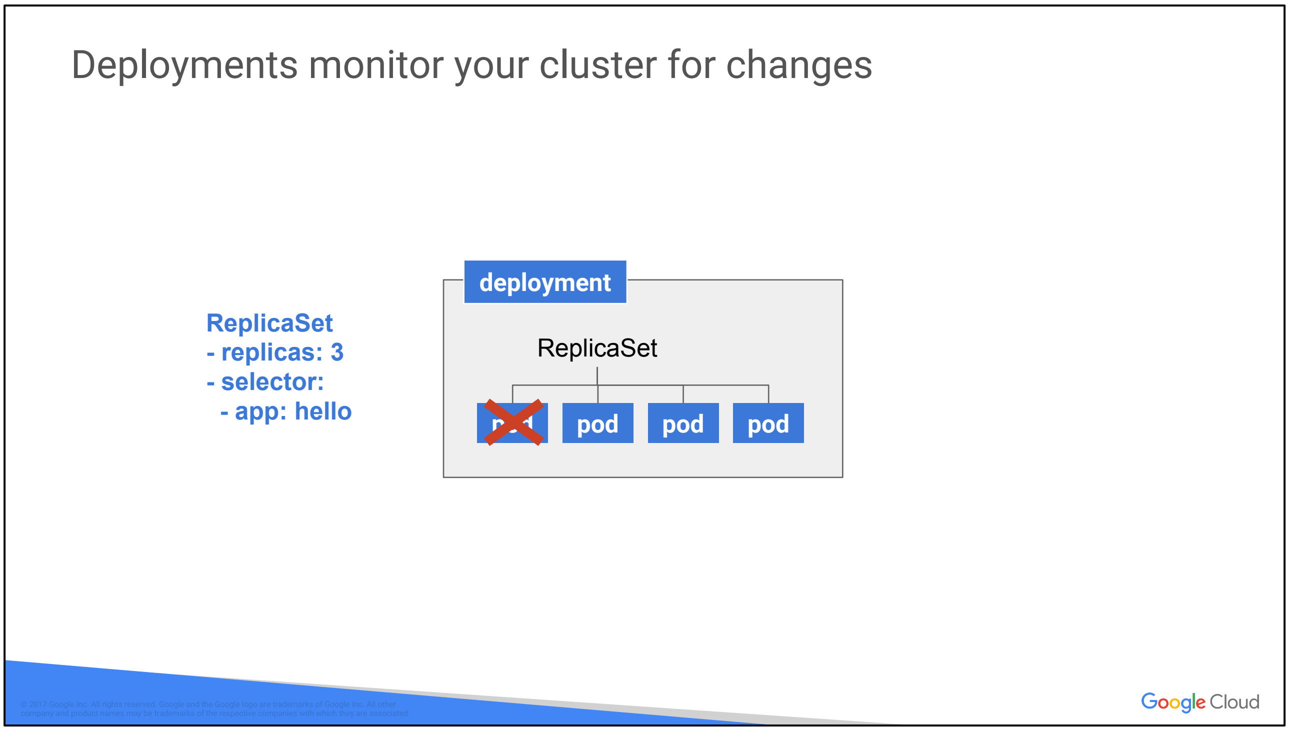
They also make corrections if pods stop running

ReplicaSet
- replicas: 3
- selector:
  - app: hello



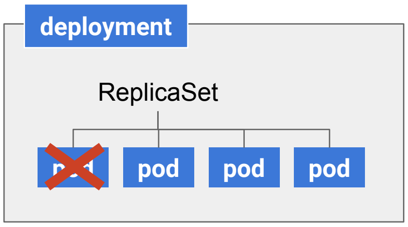
Google Cloud

If you had 3 pods, and one of them went away for some reason, for example if the node went down or there was a node upgrade and it was taken down by the system ...

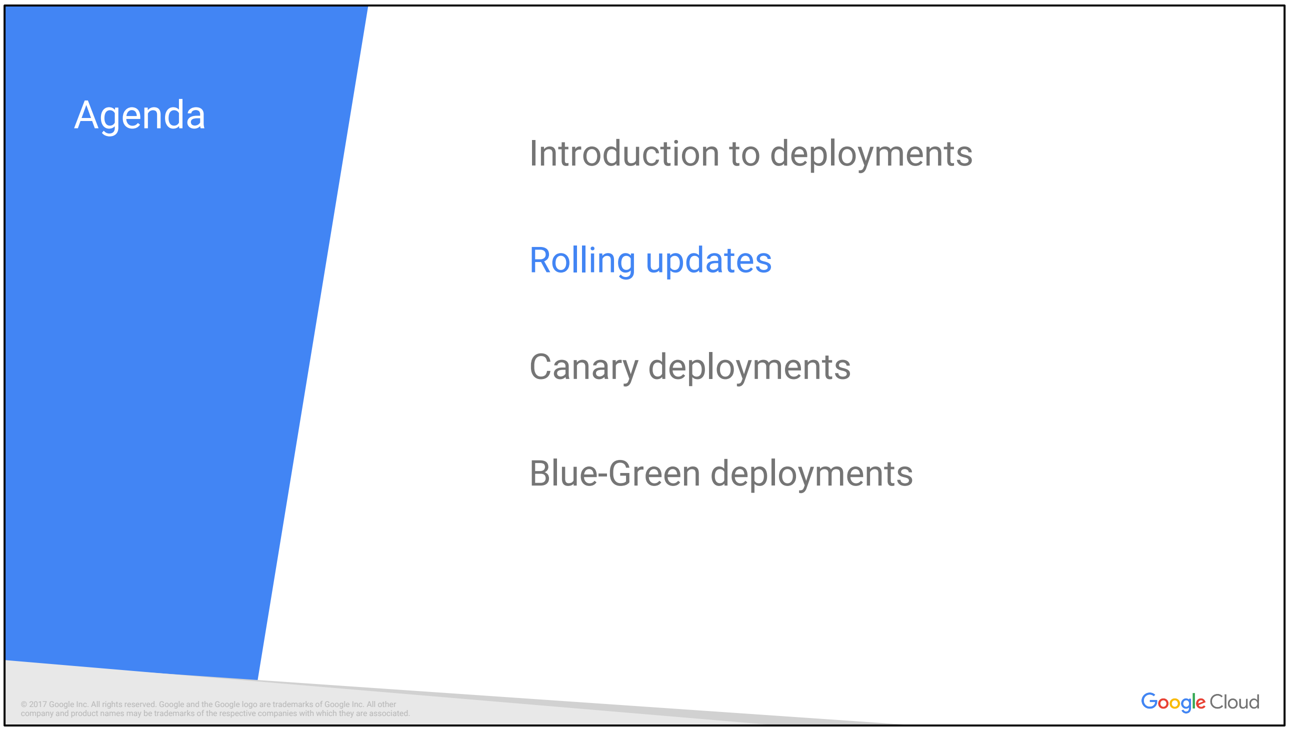


Deployments monitor your cluster for changes

ReplicaSet
- replicas: 3
- selector:
- app: hello



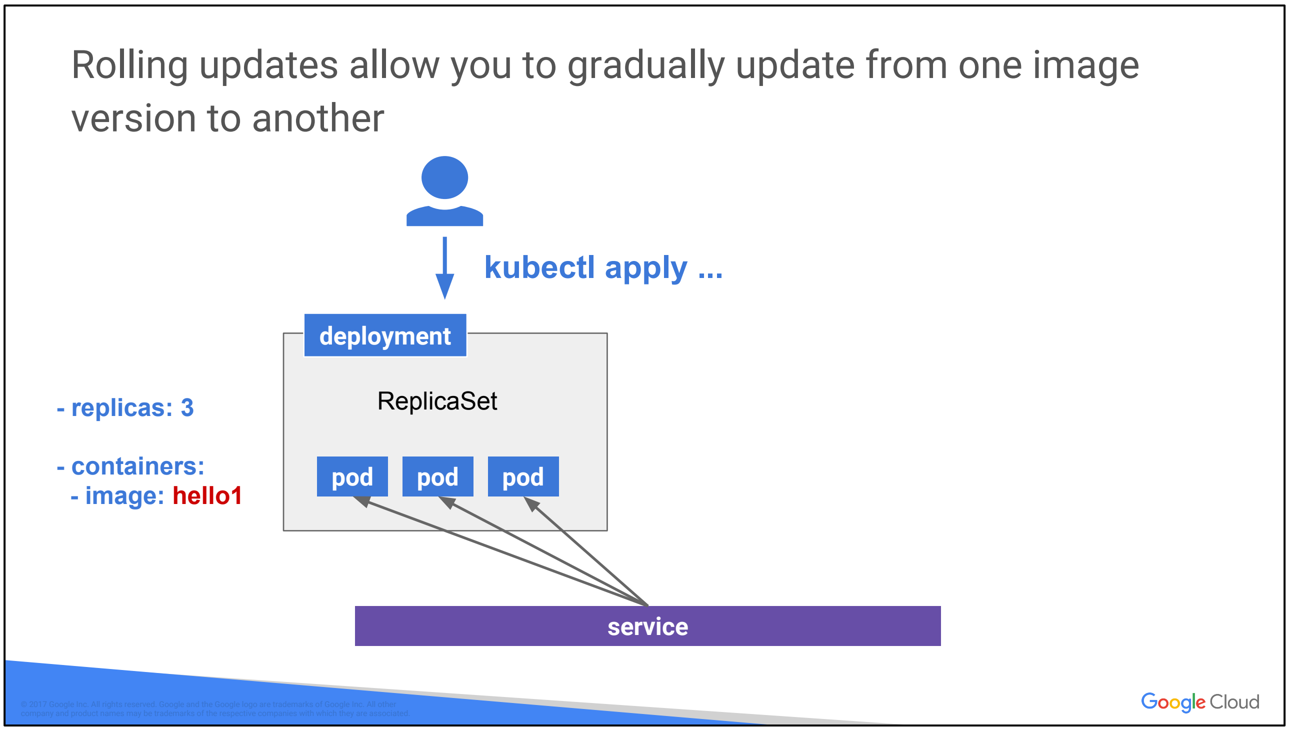
… then the pod would be rebuilt somewhere and your original state would be restored.



Agenda

- Introduction to deployments
- Rolling updates
- Canary deployments
- Blue-Green deployments

One of the things deployments support is the rolling update.

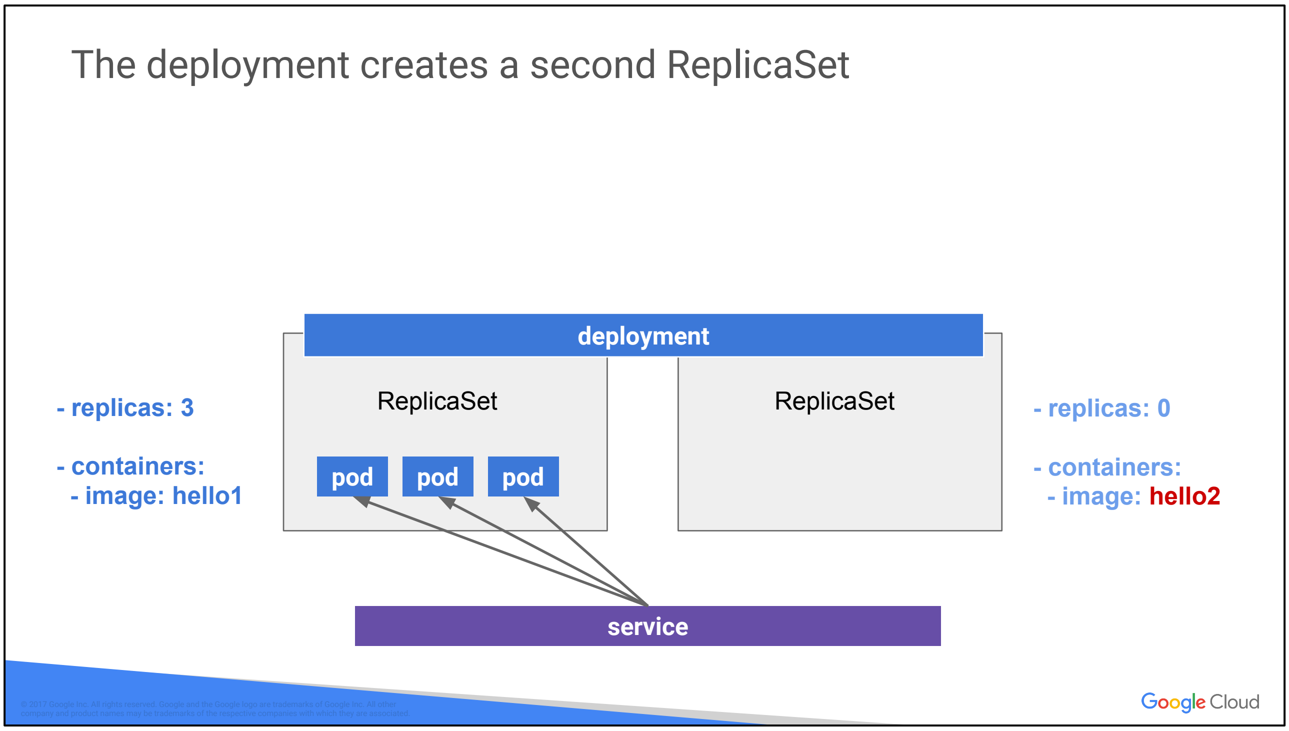


Rolling updates allow you to gradually update from one image version to another

Rolling updates allow you to gradually update from one image version to another, for example when you want to update from version 1 to version 2.

Deployment rollouts are triggered if and only if the deployment's pod **template** (that is, **.spec.template**) is changed; for example, if the labels or container images of the template are updated. Other updates, such as scaling the deployment, do not trigger a rollout.

So we're going to use the **kubectl** command to apply that change.

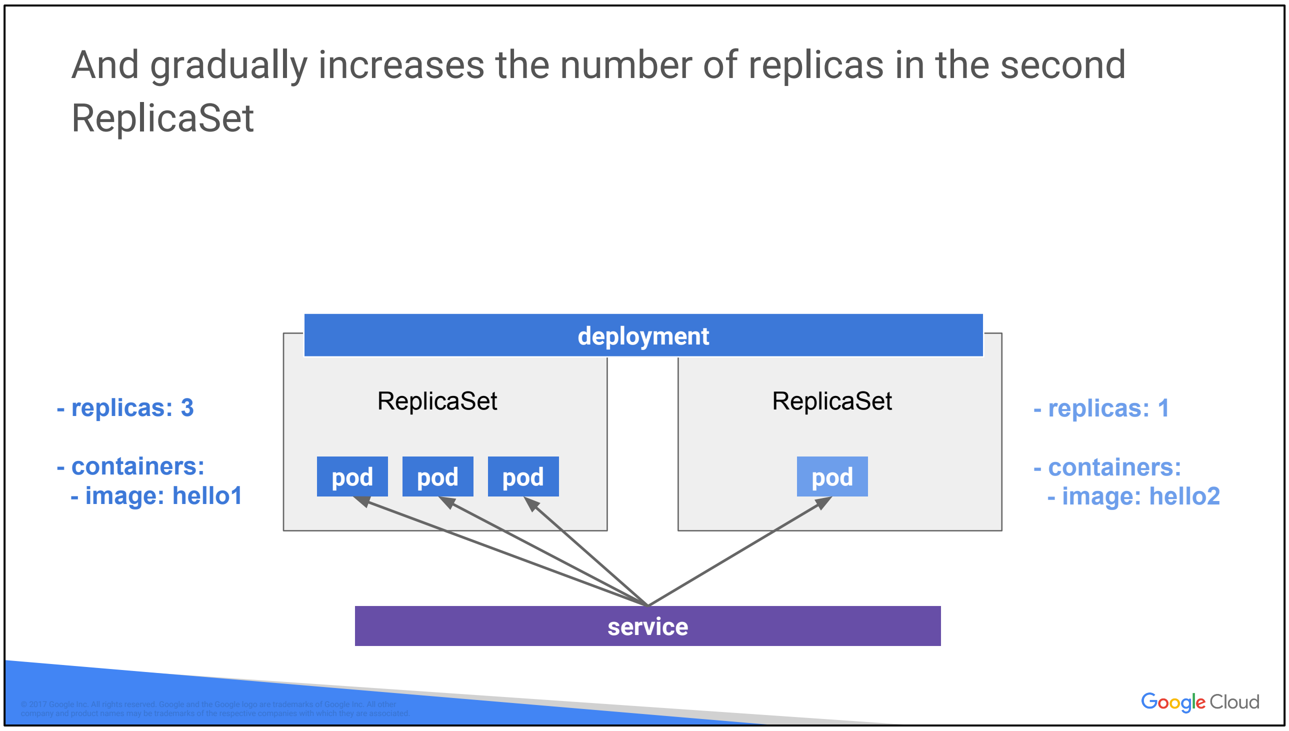


The deployment creates a second ReplicaSet

Google Cloud

And you're going to add a version two image for our pods.

And the deployment is going to create a second ReplicaSet



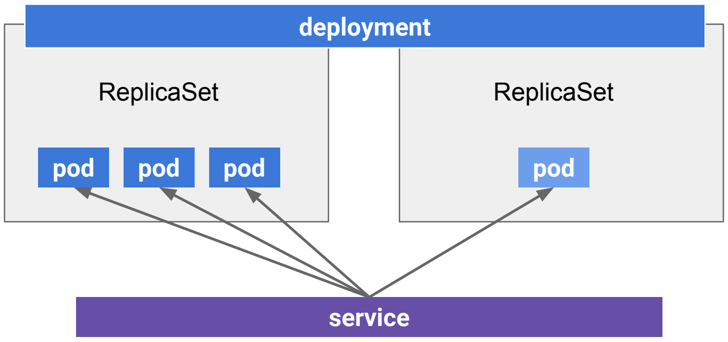
And gradually increases the number of replicas in the second ReplicaSet

- replicas: 3

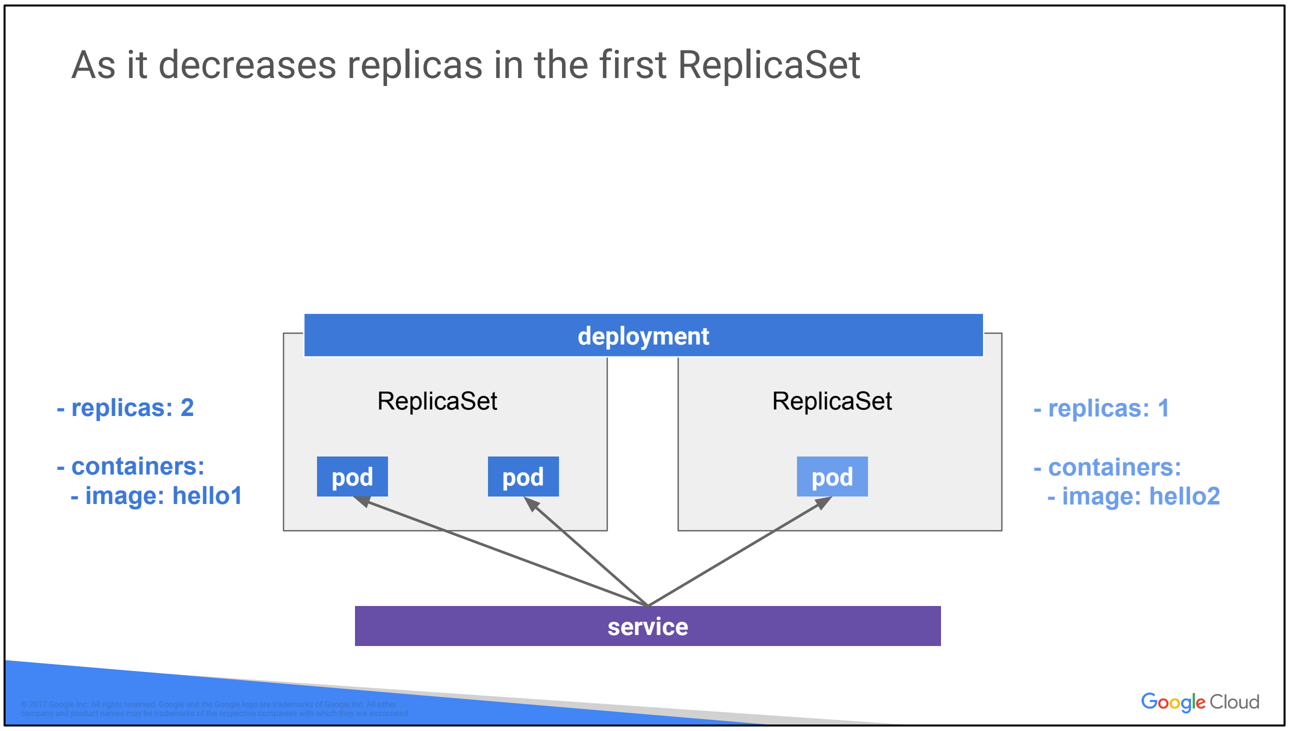
- containers:
  - image: hello1

- replicas: 1

- containers:
  - image: hello2



… and create pods in the new ReplicaSet ...



As it decreases replicas in the first ReplicaSet

- replicas: 2

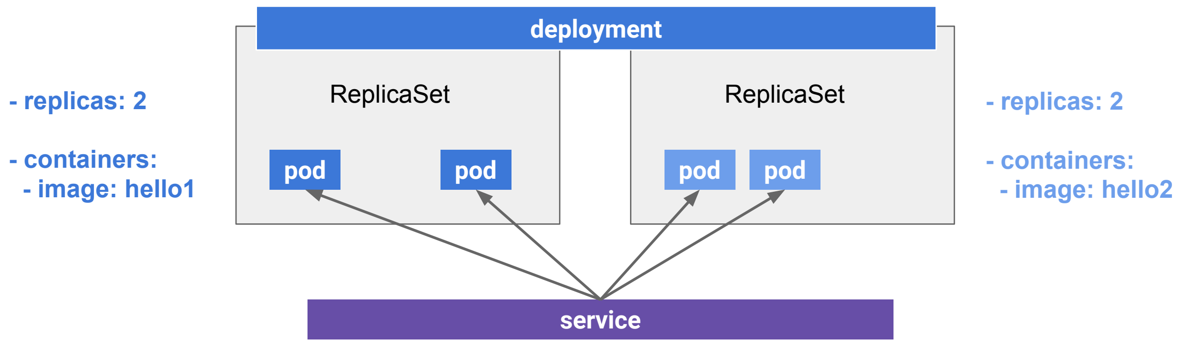
- containers:
  - image: hello1

- replicas: 1

- containers:
  - image: hello2

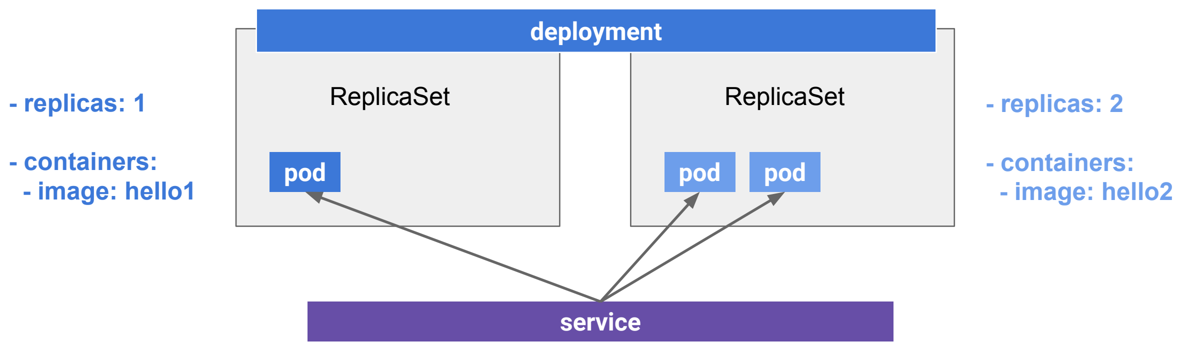
… as it shuts down pods from the old ReplicaSet so we keep the same number of pods that we asked for in the deployment.

So at any time you have at most 4 pods



So at any point, we have at most 4 pods.

And at least 3 pods

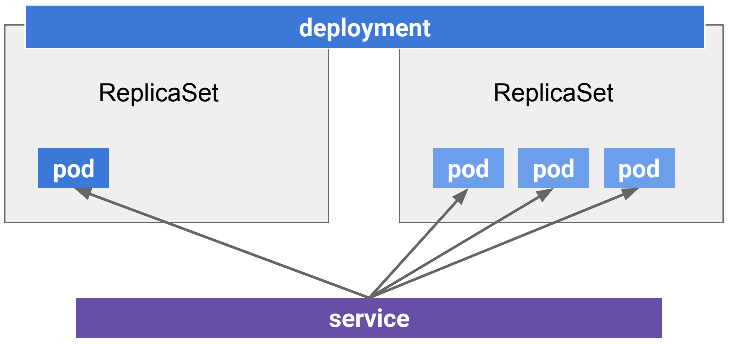


And at least 3 pods.

And this continues

- replicas: 1

- containers:
  - image: hello1



- replicas: 3

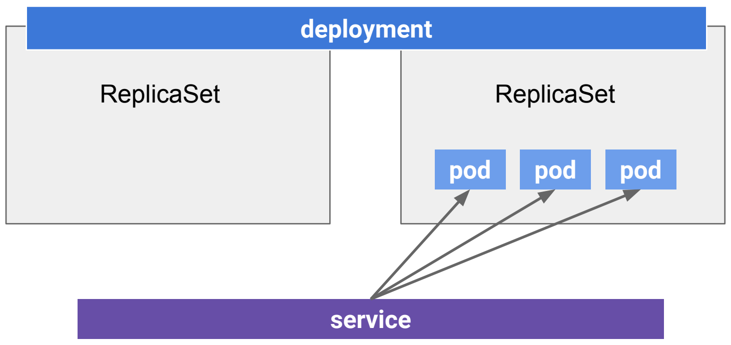
- containers:
  - image: hello2

Google Cloud

Until the new image version is rolled out

- replicas: 0

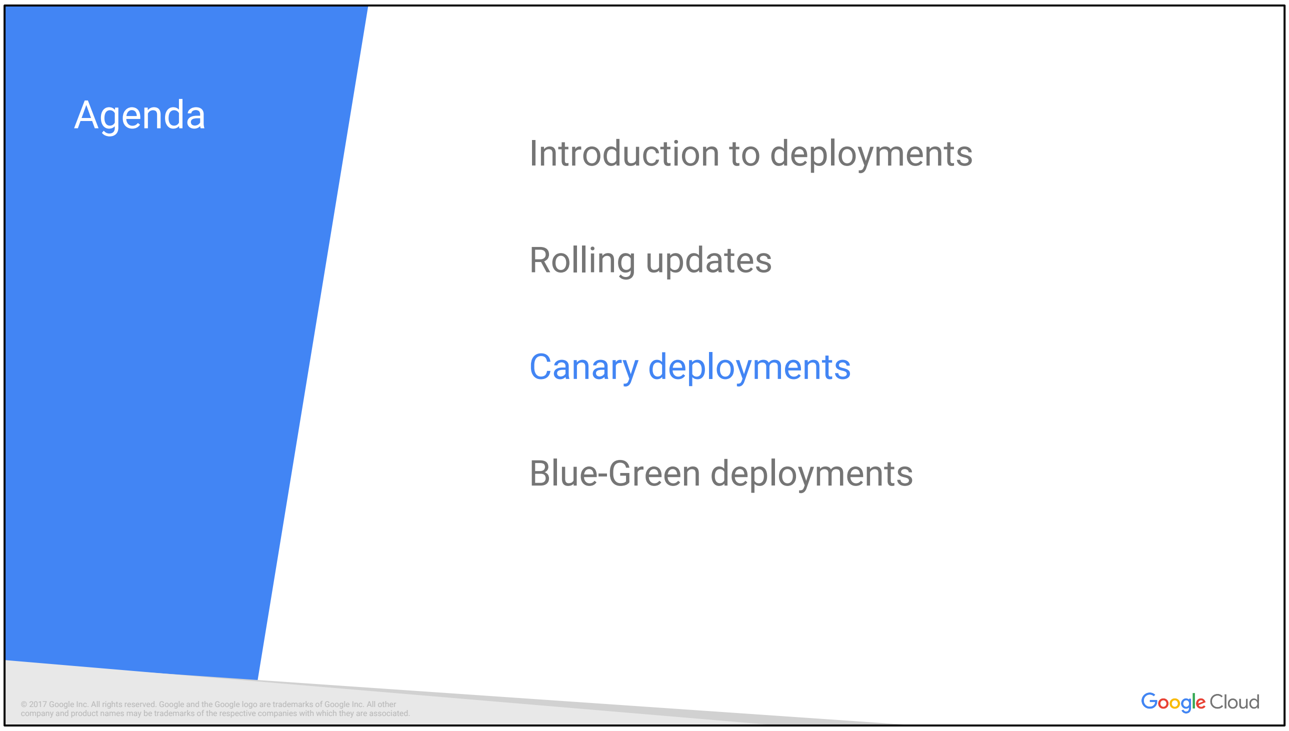
- containers:
- image: hello1



- replicas: 3

- containers:
- image: hello2

And now we can see that version 2 has rolled out. And that was all done through one API request that just changed the deployment to use version 2 instead of version 1.

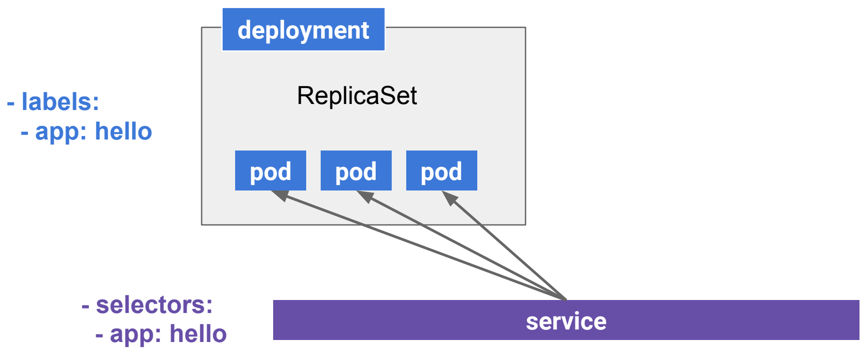


Agenda

Google Cloud

One other way you can deploy is by doing Canary deployments.

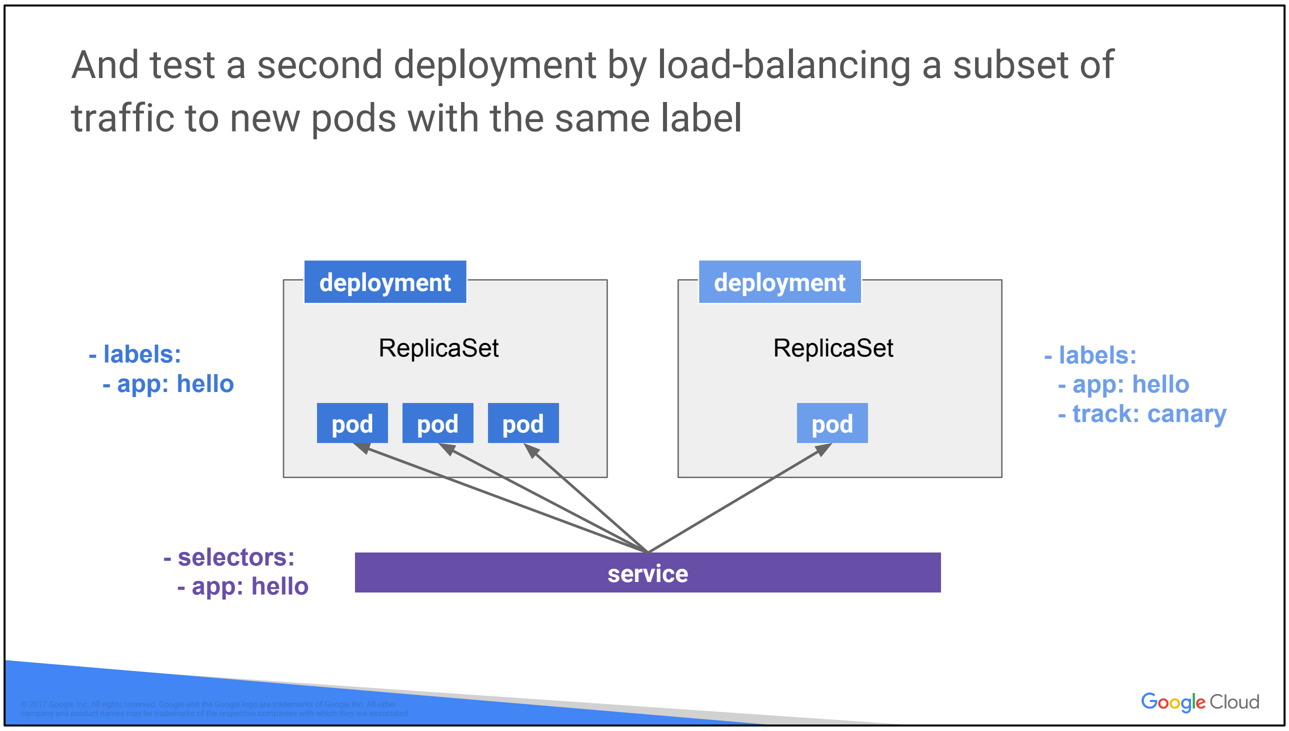
A Canary deployment relies on a service to load-balance traffic to primary pods based on label selectors



Google Cloud

So in this case, we have a service pointing at our deployment and the service is using the label selector **app: hello**. So it's going to load-balance to any pods that have the **app: hello** labels.

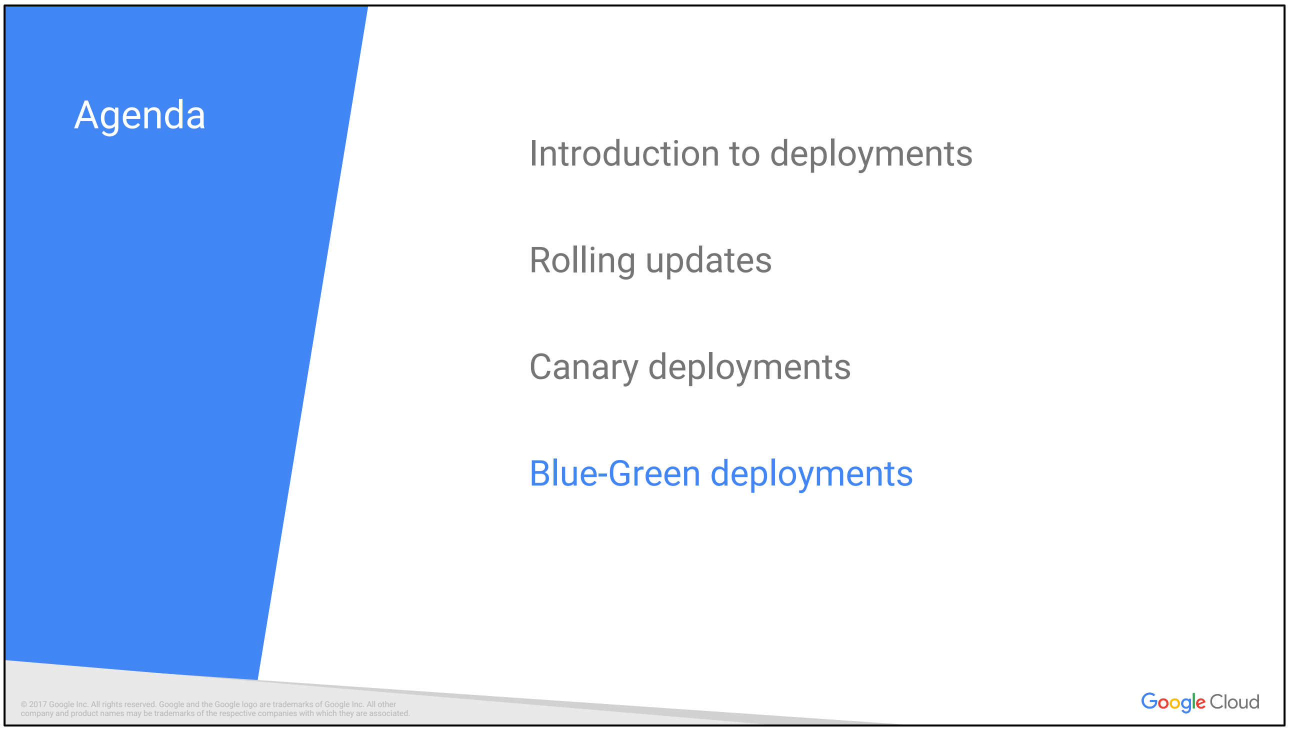
In this case we only have one deployment doing that.



And test a second deployment by load-balancing a subset of traffic to new pods with the same label

But you may have a second deployment that also adds a second label called **track: canary**. In this case, since the second deployment also has the **app: hello** label, the service also load-balances against those pods.

So you can try out a new version of your application against a small subset of your live requests, and then when you're satisfied, you can roll it out to the new deployment.



Agenda

Introduction to deployments

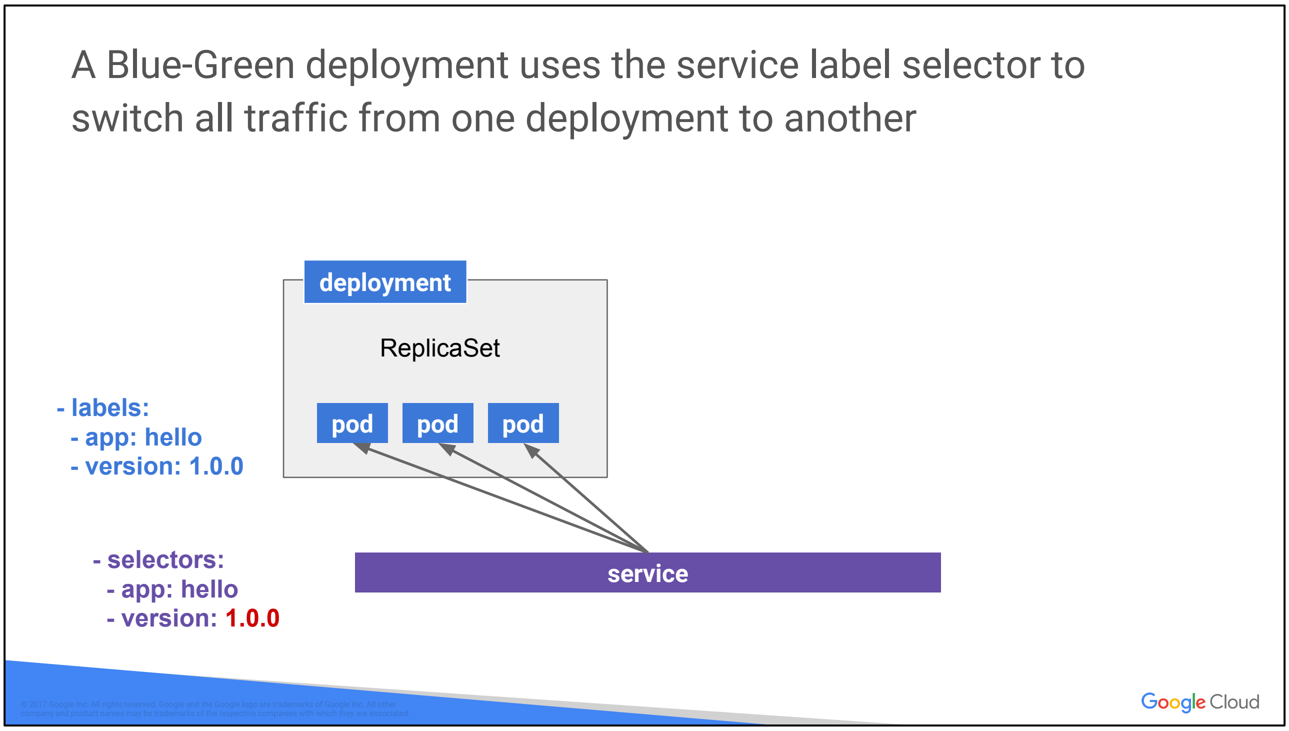
Rolling updates

Canary deployments

Blue-Green deployments

Google Cloud

One other way you deploy is using Blue-Green.



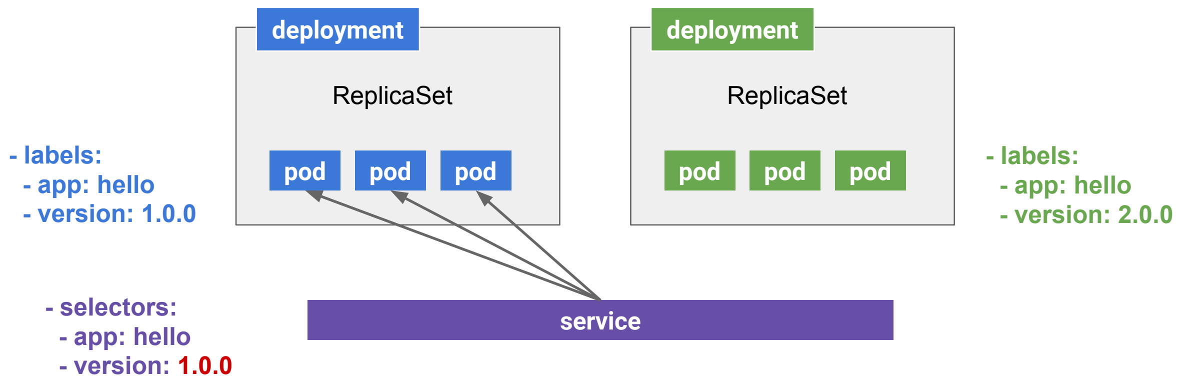
A Blue-Green deployment uses the service label selector to switch all traffic from one deployment to another

A Blue-Green deployment switches all traffic from one deployment to another at once.

So in this case you have one deployment with pod labels **app: hello** and **version: 1.0.0**.

The service maps traffic to those pods because they match both service label selectors.

First you bring up and test your new deployment without live traffic

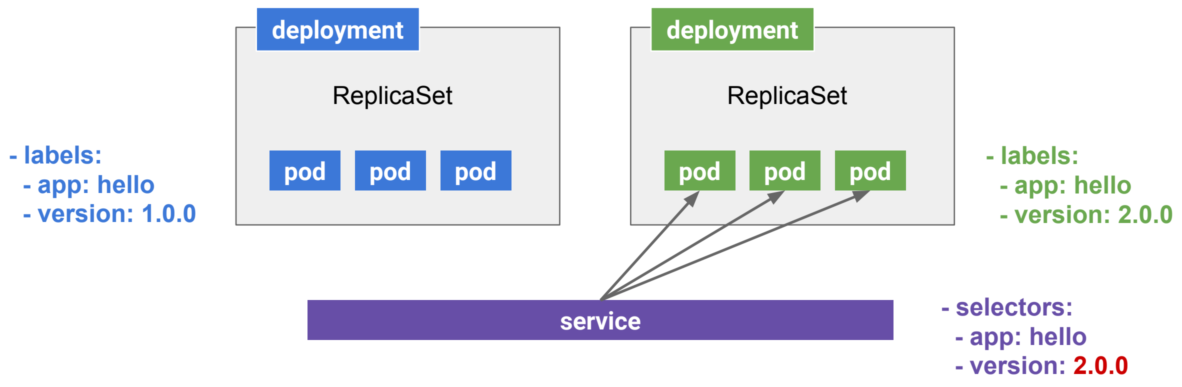


Google Cloud

We're going to roll out a **FULL** second deployment with label selectors **app: hello** and **version 2.0.0**.

As we're verifying the new deployment, we can ensure that things are running the way we want to without routing any live traffic to our new pods.

When you want to make that version live, change the service label selector, which switches all traffic



And as soon as you want to make that version live, you just change the service label selectors from **1.0.0** to **2.0.0** and that switches the traffic over to the different pods.

Lab

So, you understand how you can deploy and orchestrate pods in Kubernetes.

Let's check your knowledge and perform lab exercises to set up and test configurations.