Deployment Strategies

A deployment strategy is <mark>a way to change or upgrade an application</mark>.

- **Rollout** is when you replace or update servers with new version of an application
- **Rollback** is when you replace or revert recently updated servers back to the previous version

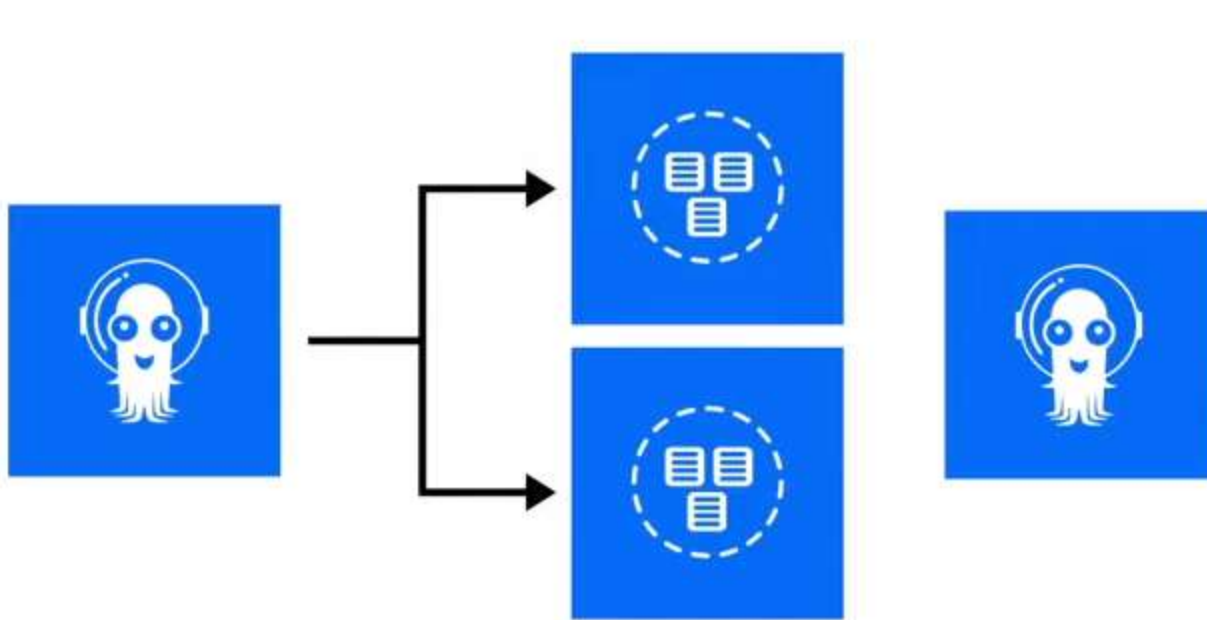There are several different deployment strategies that can be utilized with Kubernetes:

- **Recreate** – Terminate the current pods, create new pods all at once

- **RollingUpdate** – Replaces one or multiple pods at a time

- **Canary** – Add new pods and route a subset of your users to the new server, if no bugs or errors occur, roll out changes to all pods

- **Blue / Green** – Deploy an exact copy of your entire infrastructure, swap the traffic and then terminate the old environment or rollback to old environment

- **A/B Testing, Red / Black** – Similar to Canary and Blue/Green but continuously keep the new environments running to test features on a subset of users

- **Dark Launches** – Similar to A/B Testing, except you use a feature flag to rollout new features, and rollback by turning off the feature in the software instead of reverting infrastructure changes
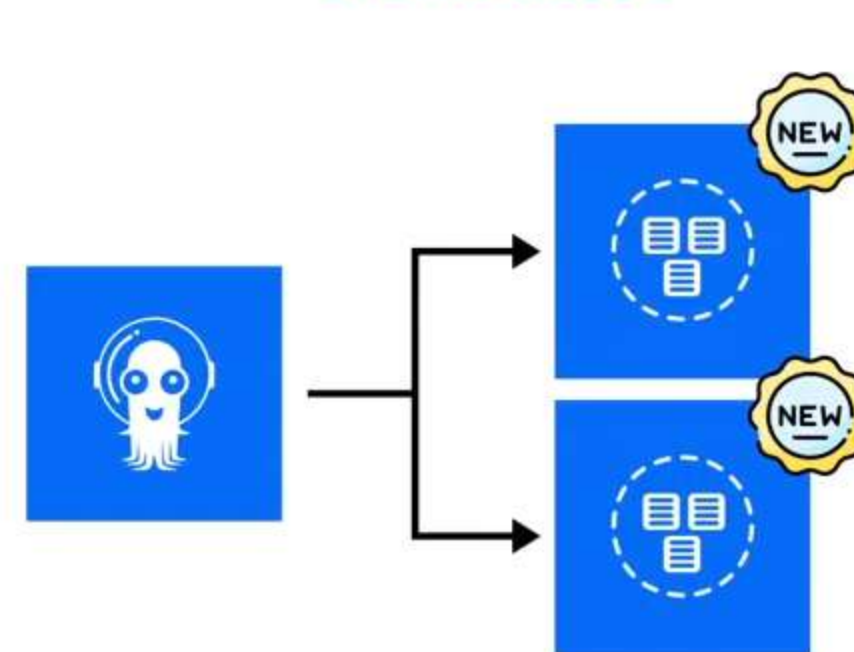
Recreate

Terminate all running instances and recreate with a new version.
Also known as an In-Place deploy

```
spec:
  replicas: 3
  strategy:
    type: Recreate
```

This is a **snippet**

**Terminate all servers.
Traffic is interrupted**

**New servers
are created**
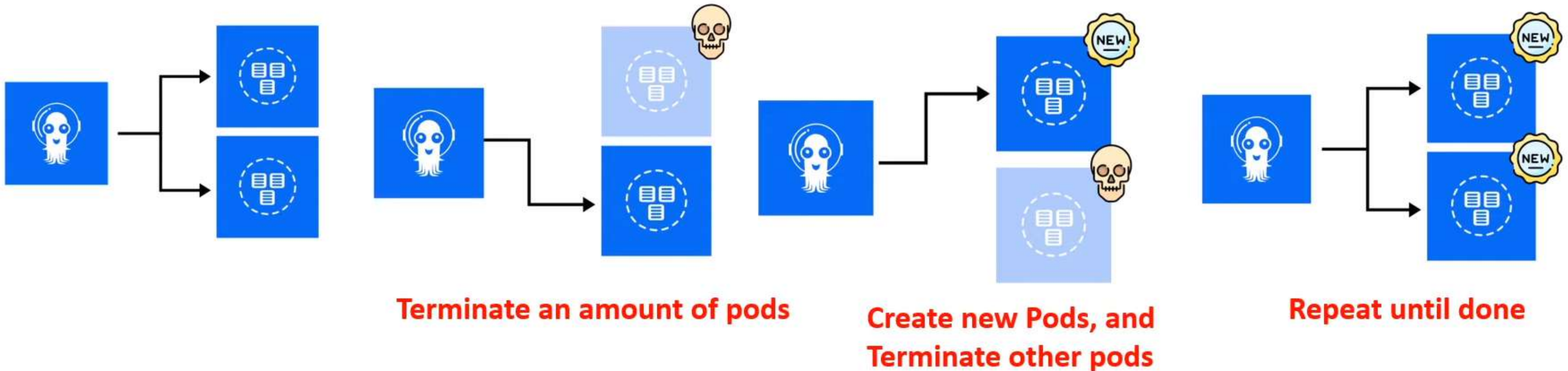
- Users will experience downtime.
- Can be very fast
- Very simple
- Rollback is not possible
- Ideal for non-production workloads or where interruptions can be tolerated

Rolling Updates

Rolling Updates **slowly replaces pods one by one**.
This is the **default strategy** of Kubernetes

Terminate an amount of pods

**Create new Pods, and
Terminate other pods**

Repeat until done

- *Reduced availability might happen while each set of pods is taken terminated as new are created
- Rollbacks can be slow and hard
- Deploys will be slow

Rolling Updates

What is availability?

The quality of being able to be used or obtained.

If there is not enough capacity (memory, CPU, bandwidth) to meet the demand of traffic then users can experience degraded, delayed experience or no access to services at all.

Two important values:

maxSurge:

- The amount of pods that can be added

maxUnavailable :

- The amount of pods that can be unavailable

```
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2          # how many pods we can add at a time
      maxUnavailable: 0    # how many pods can be unavailable
```
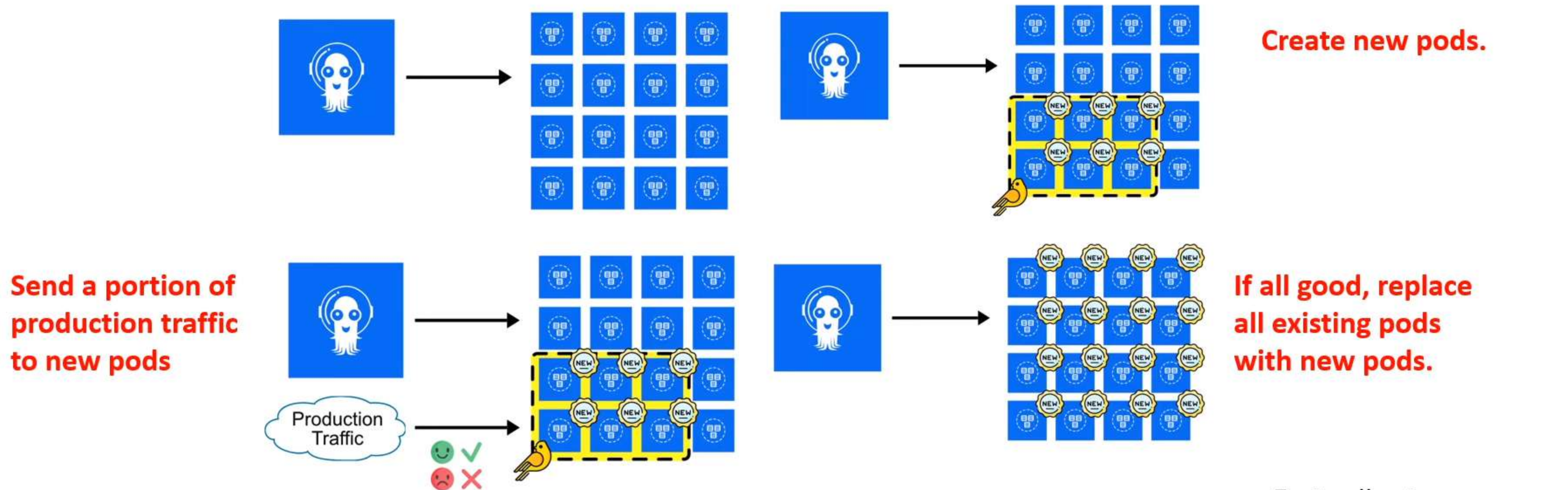
AVAILABLE ✓ A maxSurge of 2 and a maxUnavailable of 2 would <u>ensure no drop in availability</u>. The deployment would have to first create the new pods before tearing down the old.

Canaries

Deploy a new version of the app into a new pod and serve it to a subset of existing users.
If no error or bug has occurred then rollout changes to all users by replacing old pods with new pods

Create new pods.

Send a portion of production traffic to new pods

If all good, replace all existing pods with new pods.

Canary will use the load balancer weighted rules to only send an amount of traffic to the the canary pods and original pods.

- Fast rollout
- Slow rollback
- No drop in availability

Blue/Greens

Blue / Green Deployment Is when you **completely create a new environment of all components**, and you send all traffic to the new (Green) environment, if its all okay, you then terminate the old (blue) environment. If anything goes wrong you rollback to blue and teardown green.

- Zero downtime
- No reduce availability
- Slow to deploy but faster than Canary
- If something goes wrong larger impact to users immediately
- Instantly rollback to previous infrastructure
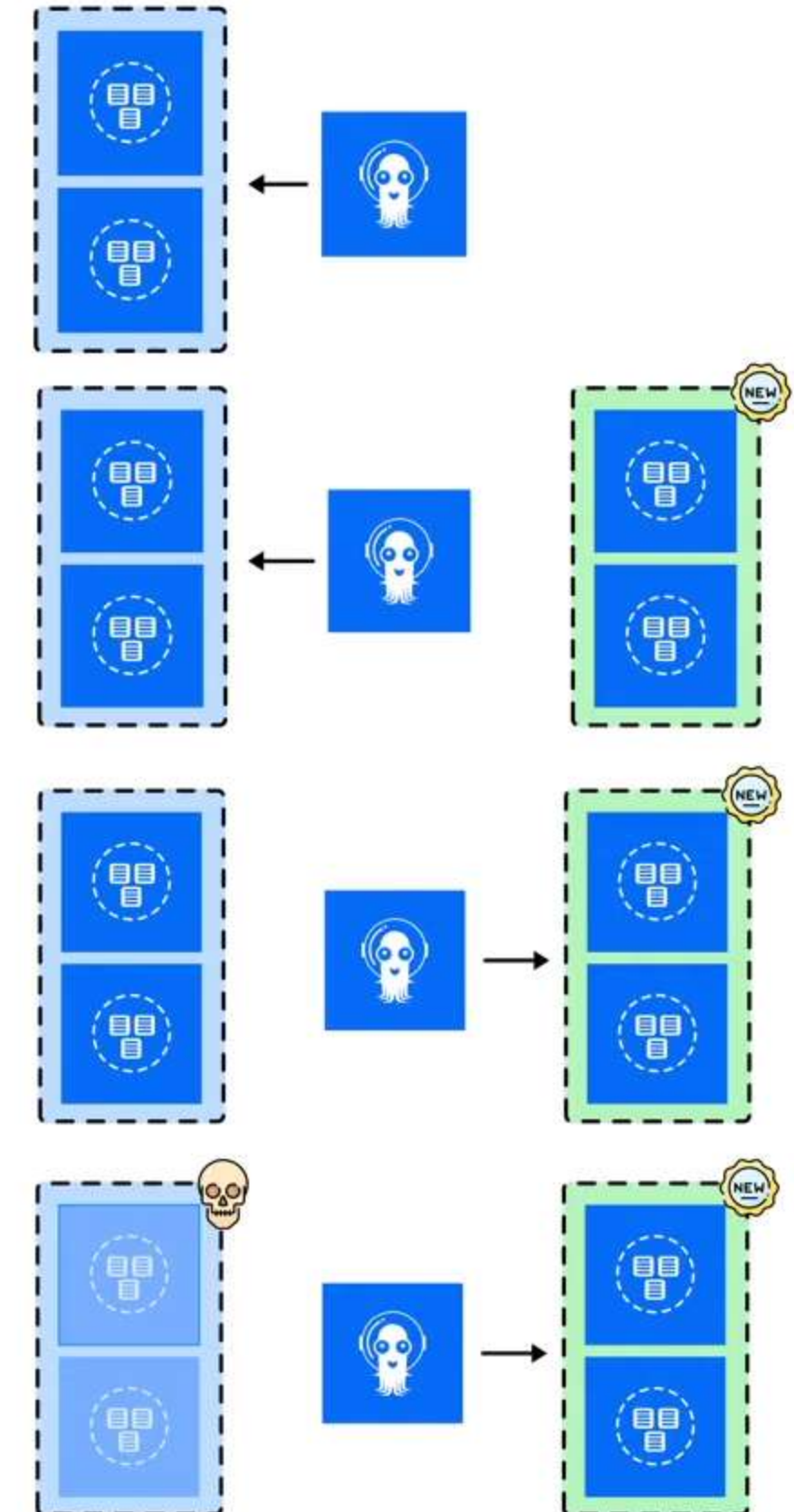
A to B Testing or Red/Black

Uses a Canary or Blue Green method of deployment but serves the new app (experimental features) to subset of users based on a set of load balancing rules.


A: 60%

B: 40%
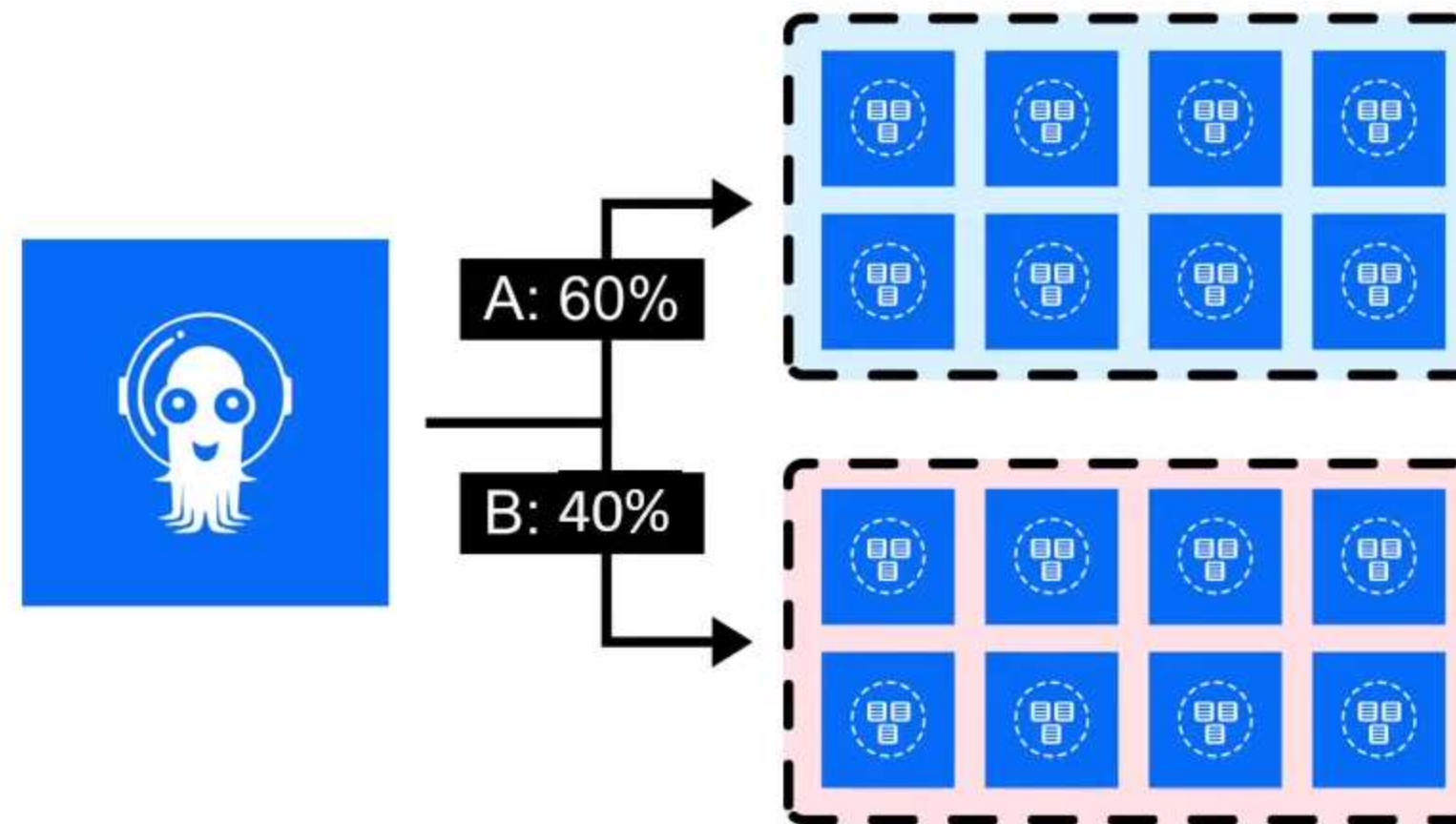
≠ The key difference is Canary and Blue/Green deployment is temporary as you intend to rollout changes to all pods. Where A-to-B Testing and Red/Black happen for a long period of time.

Dark Launches

Similar to A/B Testing, except A to B happens **at the application layer** (within the app code)

- You want to test a feature on a subset of users,
- You code a feature flag into your app to turn the new feature on or off
- If the users like the feature you leave it switched on, if they don't you turn it off.
- Doesn't require you to rollback at the infrastructure level. (fast rollback)
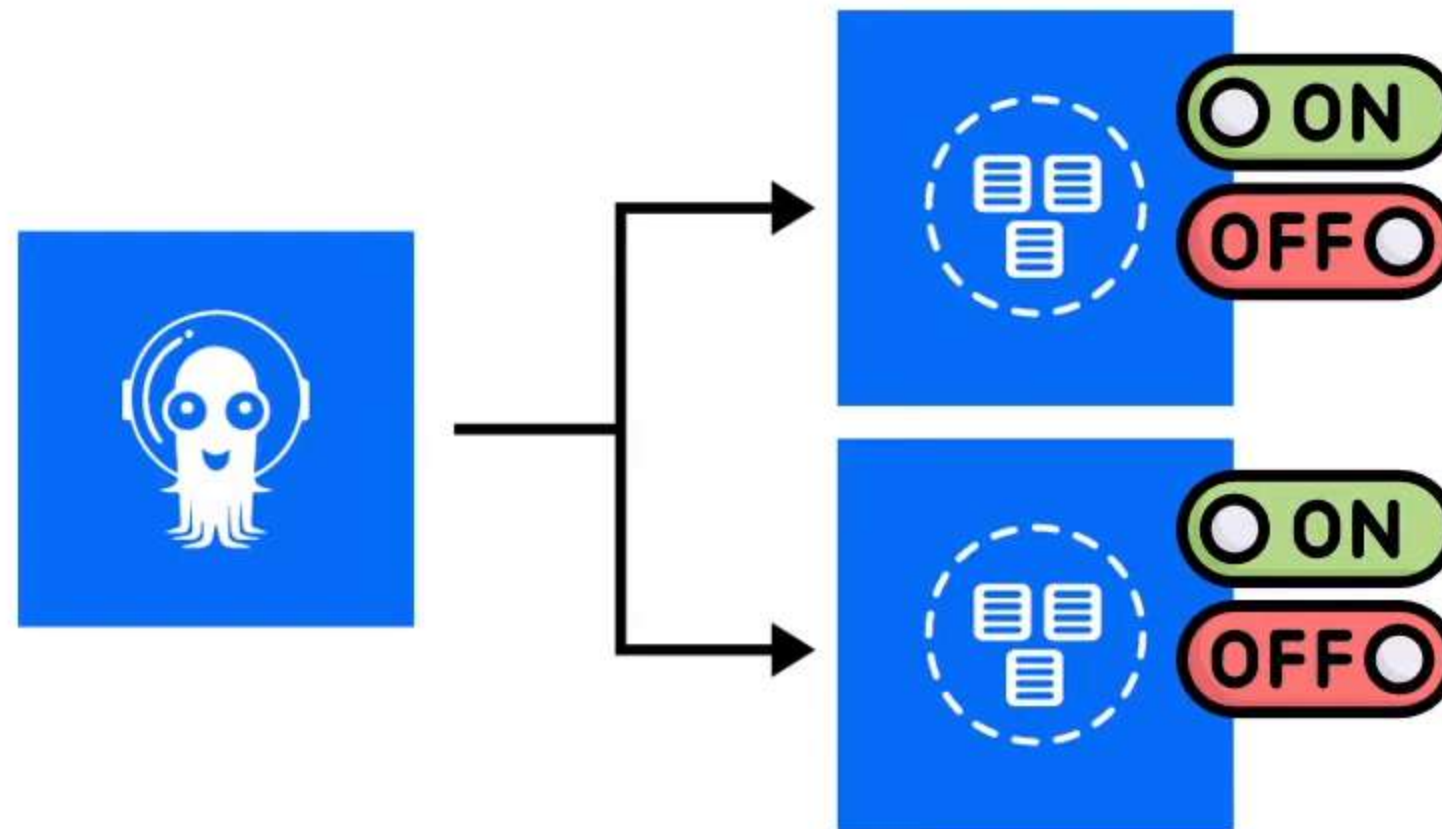
Deployment History Command

You can check the history of previous deploys with the following command:

```
kubectl rollout history deploy/<deployment name>
```

```
andrewbrown:~/environment $ kubectl rollout history deploy/my-app
deployment.apps/my-app
REVISION   CHANGE-CAUSE
1          kubectl apply --filename=k8s/my-app.yml --record=true
```

Deployment Rollout Status

You can the status of you deploy with the rollout status command:

```
kubectl rollout status deployment/<deployment name>
```

```
andrewbrown:~/environment $ kubectl rollout status deploy/my-app
Waiting for deployment "my-app" rollout to finish: 1 of 2 updated replicas
are available...
deployment "my-app" successfully rolled out
```

Deployment Rollback

You can rollback to the previous deploy (shown in the rollout history) with rollout undo

```
kubectl rollout undo deployment/<deployment name>
```

```
andrewbrown:~/environment $ kubectl rollout undo deploy/my-app
deployment.apps/my-app rolled back
andrewbrown:~/environment $ kubectl rollout status deploy/my-app
Waiting for deployment "my-app" rollout to finish: 1 out of 2 new
.
```