SUMMER OF KUBERNETES

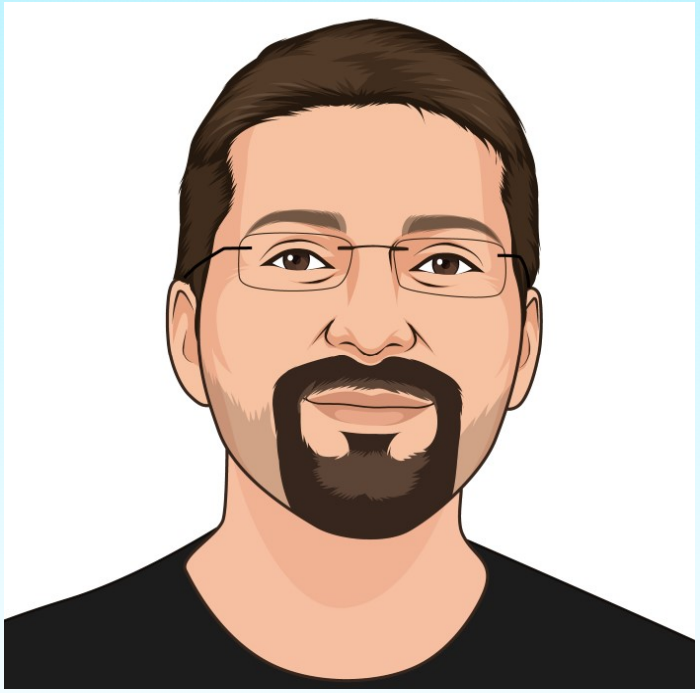
BROUGHT TO YOU BY AMBASSADOR LABS

Argo Rollouts

Progressive Delivery and Canary releases

Your host: Kostis Kapelonis

- Developer Advocate
- Company: Codefresh CI/CD/Gitops
- Check codefresh.io/blog
- Ex-Java dev (10+ years)
- Ex-Release manager (5+ years)
- Member of Argo Rollouts Github Org



SUMMER OF KUBERNETES
BROUGHT TO YOU BY AMBASSADOR LABS

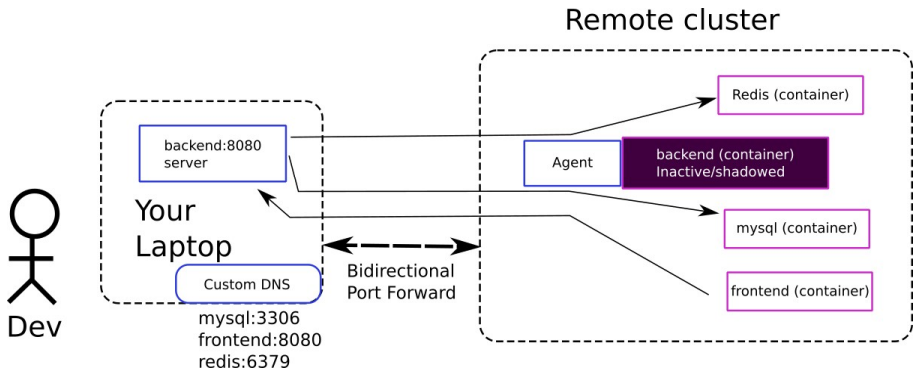
https://codefresh.io/kubernetes-tutorial/telepresence-2-local-development/

DEVOPS

Using Telepresence 2 for Kubernetes debugging and local development

15 min read

Kostis Kapelonis · Apr 15, 2021



Agenda

1. Vanilla Kubernetes deployments
2. What is progressive delivery
3. Blue/Green deployments
4. Canary deployments
5. Intro to Argo Rollouts
6. Demo/Exercise
7. Discussion and Q/A

SUMMER OF KUBERNETES
BROUGHT TO YOU BY AMBASSADOR LABS

SUMMER OF KUBERNETES

BROUGHT TO YOU BY AMBASSADOR LABS

Vanilla Kubernetes

What you get out of the box

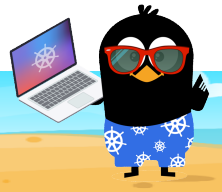
Default deployment strategies

Recreate

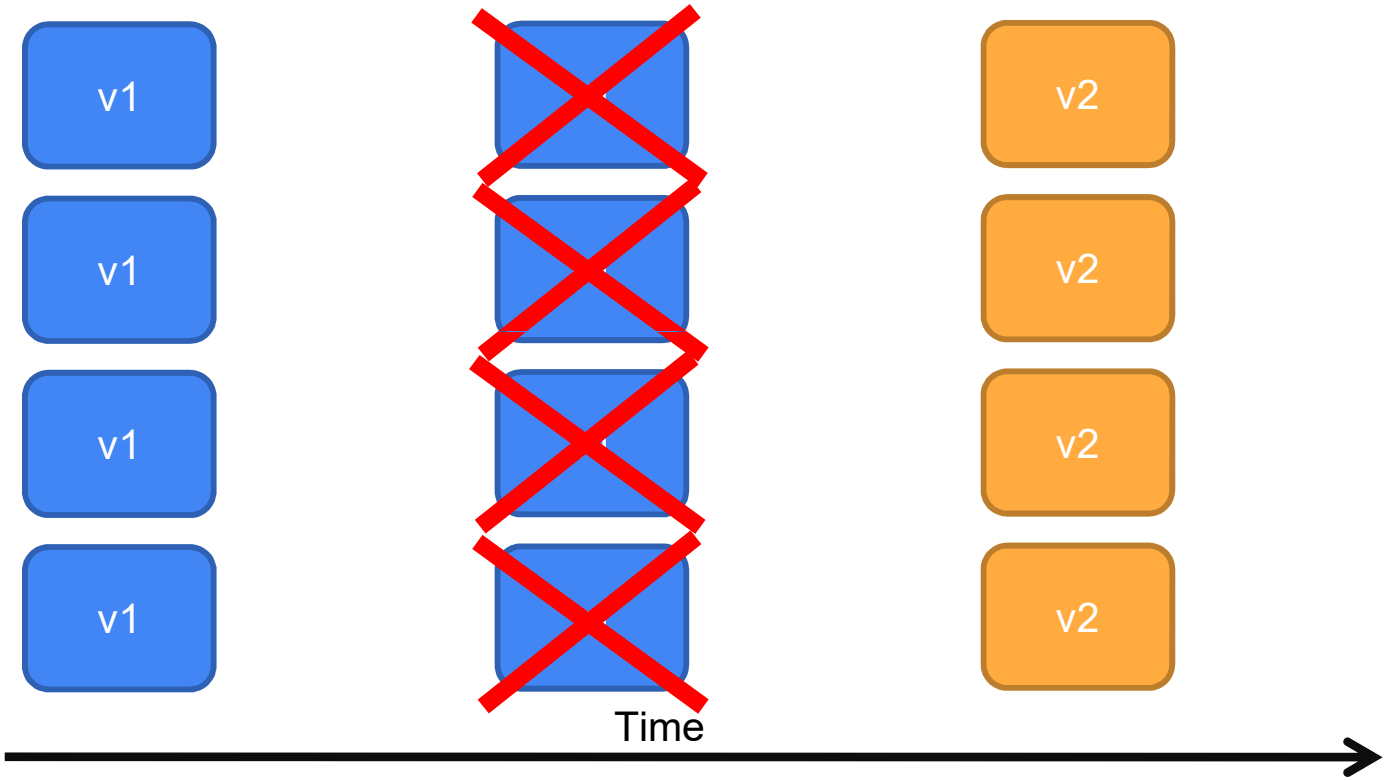
Deletes all pods and then starts the new ones

Rolling Update

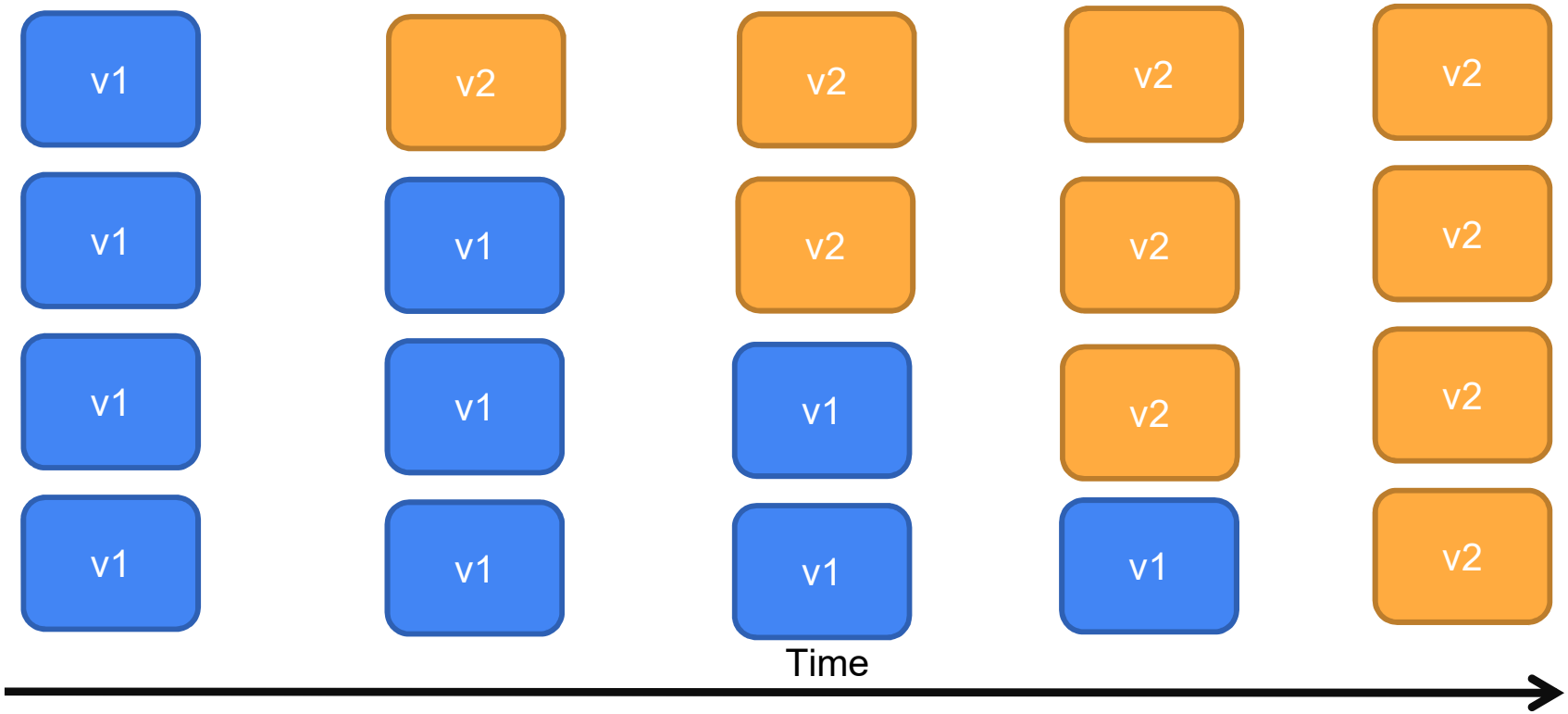
Replaces old pods with new ones (one-by-one or in batches)

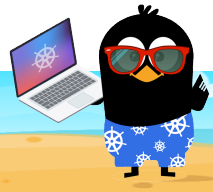


Recreate deployment strategy



Rolling Update deployment strategy





Issues with default strategies

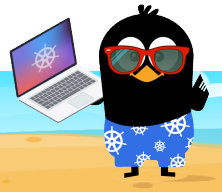
- The Recreate strategy results in downtime
- Rolling updates can only move forward
- You cannot control who sees new version and who sees old version
- Cannot easily run smoke tests or check metrics in the middle of a deployment
- Percentage of traffic that sees new version is always associated with number of pods (default K8s load balancing)
- In all cases rolling back requires starting a new deployment process

Choosing a strategy

Defined under spec.strategy.type

Either RollingUpdate or Recreate

```
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2021-07-16T10:11:39Z"
  generation: 1
  labels:
    app: kubernetes-bootcamp
  name: kubernetes-bootcamp
  namespace: default
  resourceVersion: "611"
  uid: 5abf60c9-f387-4c23-9e37-99bdfd6747f5
spec:
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: kubernetes-bootcamp
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: kubernetes-bootcamp
```



SUMMER OF KUBERNETES

BROUGHT TO YOU BY AMBASSADOR LABS

Progressive Delivery

Ask for more

We want:

- No downtime at all
- Fast rollbacks (almost instant)
- control the deployment process (pause/resume/approve/rollback)
- Specify the subset of users that see the new version
- Automate rollbacks using metrics

Photo by Austin Distel on Unsplash

Enterprise deployment strategies

- Deploy new version only to internal users
- Deploy new version on to a specific geographical location
- Run smoke tests in production
- Use Prometheus, Datadog, NewRelic to check new version
- Automate rollbacks using metrics

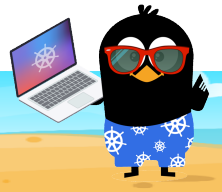
Adopting progressive delivery strategies

Blue/Green

Deploy new version while still keeping the old one around

Canaries

Gradually move live traffic to new version (while keeping the old one as well)



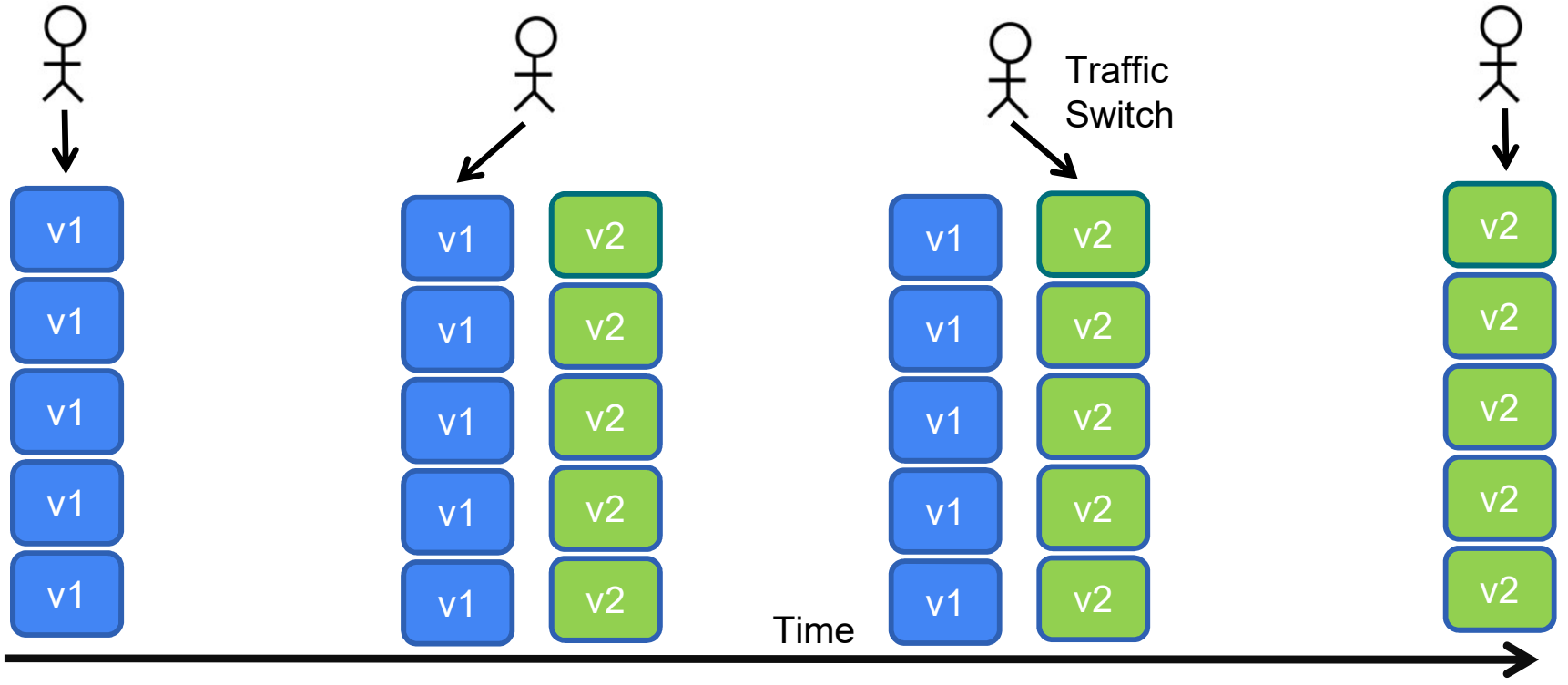
SUMMER OF KUBERNETES

BROUGHT TO YOU BY AMBASSADOR LABS

Blue/Green deployments

Easiest way to Progressive Delivery

Blue/Green deployment (a.k.a. Red/Black)



Blue/Green goals and assumptions

Pros

- No downtime
- Instant Rollback
- Simple to setup
- No ingress or service mesh required
- Can insert approvals and smoke tests

Cons

- Expensive for resources
- Needs 2x capacity
- All or nothing approach
- Cannot use metrics

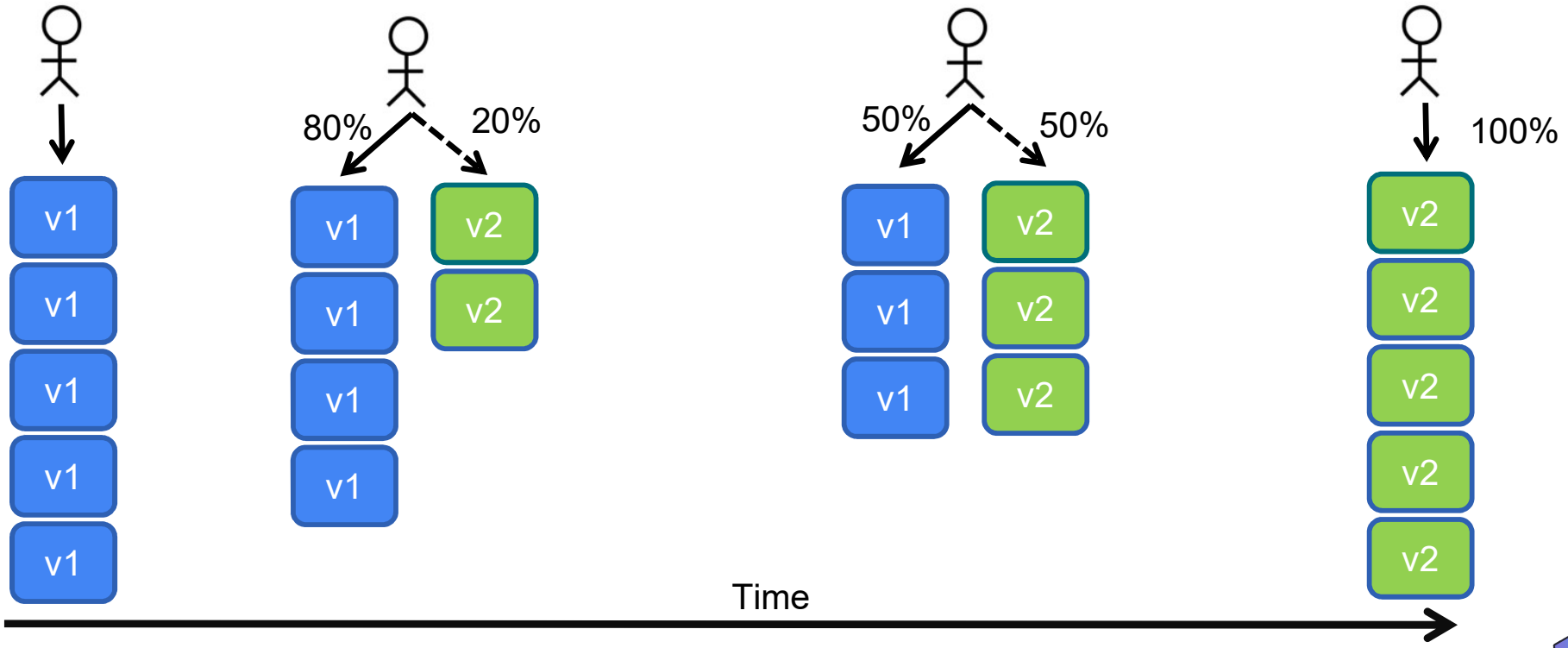
SUMMER OF KUBERNETES

BROUGHT TO YOU BY AMBASSADOR LABS

Canary deployments

The flexible way to Progressive Delivery

Canary deployment



Canary goals and assumptions

Pros

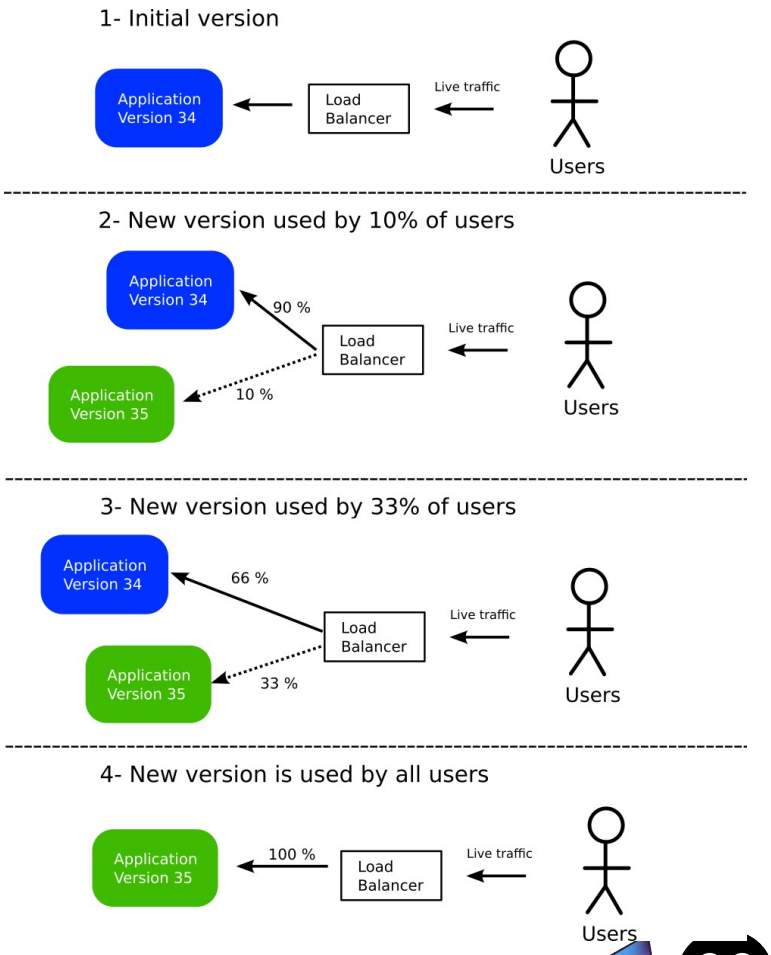
- No downtime
- Instant Rollback
- Decide who will see new version
- Can insert approvals and smoke tests
- Can use metrics
- Resource efficient

Cons

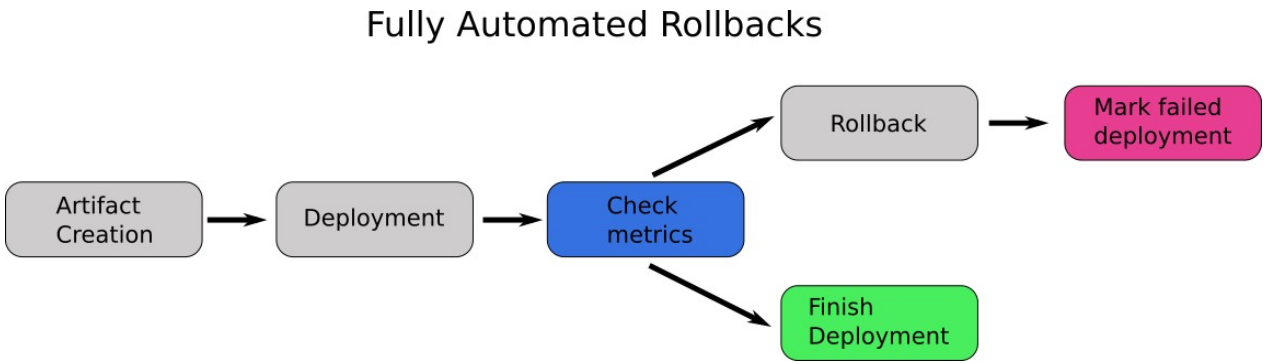
- Complex to setup
- Requires a gateway or service mesh

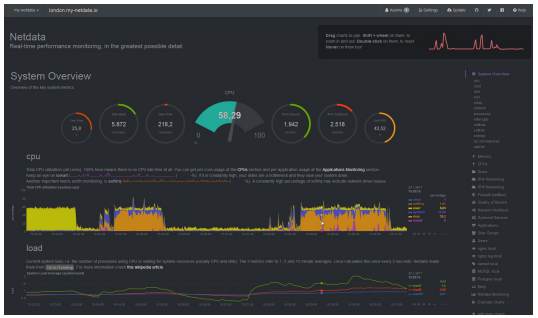
Flexible scenarios

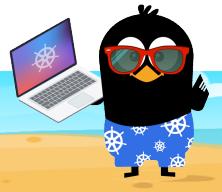
- Choose percentage (20%, 50%, 100%), (33%, 66%, 80%, 100%)
- Timeout between each stage
- Run tests between each stage
- Check your metrics at each stage



Automatic Rollbacks based on metrics







Deploy on Friday at 5pm

https://unsplash.com/photos/vvLBPW3uS4Q

SUMMER OF KUBERNETES

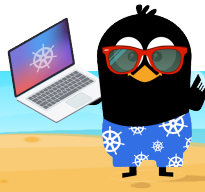
BROUGHT TO YOU BY AMBASSADOR LABS

Argo Rollouts

Progressive Delivery for Kubernetes

What is Argo Rollouts

- A Kubernetes controller (you install it on the cluster)
- It is self-contained
- Argo CD is NOT needed on the same cluster
- Introduces a new Kubernetes Resource (called rollout)
- Only responds to events on Rollouts
- When a Rollout resource changes it performs a deployment according to your defined strategy



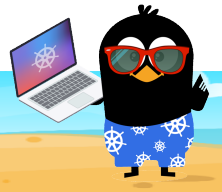
The Rollout resource

Output

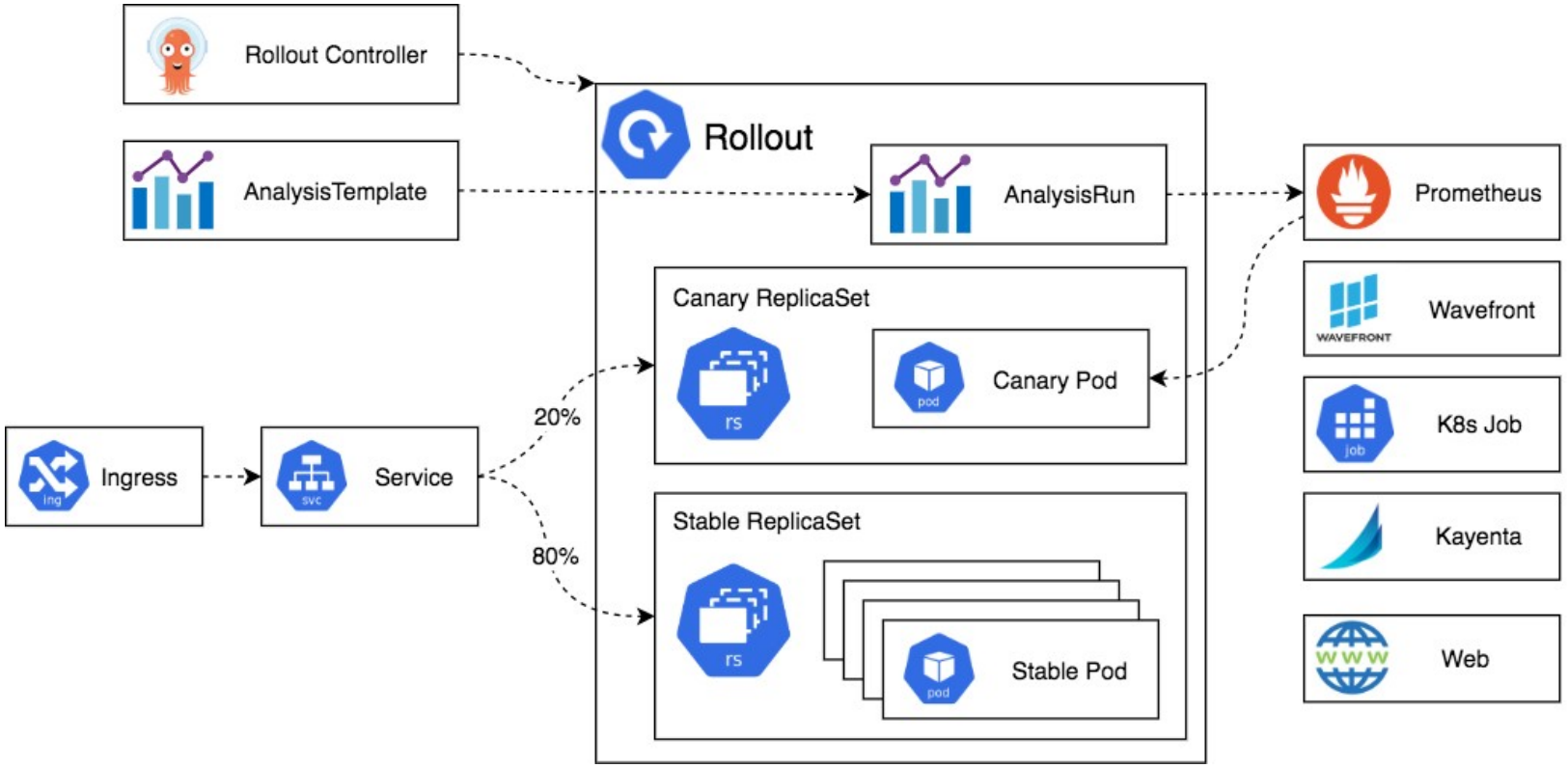
```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Rollout
3  metadata:
4    name: demo-app
5  spec:
6    replicas: 2
7    revisionHistoryLimit: 2
8    selector:
9      matchLabels:
10       app: demo-app
11   template:
12     metadata:
13       labels:
14         app: demo-app
15     spec:
16       containers:
17         - name: application-container
18           image: my-app-image:v1
19           imagePullPolicy: Always
20           ports:
21             - containerPort: 8080
22 strategy:
23   blueGreen:
24     activeService: rollout-bluegreen-active
25     previewService: rollout-bluegreen-preview
26     autoPromotionEnabled: false
27
```

Same as deployment

Extra Rollout properties



Argo Rollouts architecture



Installation

1. kubectl create namespace argo-rollouts
2. kubectl apply -n argo-rollouts -f https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml

How the Argo Rollouts controller works

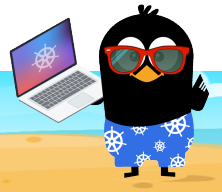
1. It will sit in the cluster waiting for events
2. Events to non-Rollouts resources are ignored
3. If a rollout resources changes the controller will take over
    1. First deployment – just deploy the app
    2. Subsequent deployment follow the defined strategy from the spec

- You can mix and match with normal deployments
- You can change the rollout with kubectl, git commit, api event, pipeline etc.

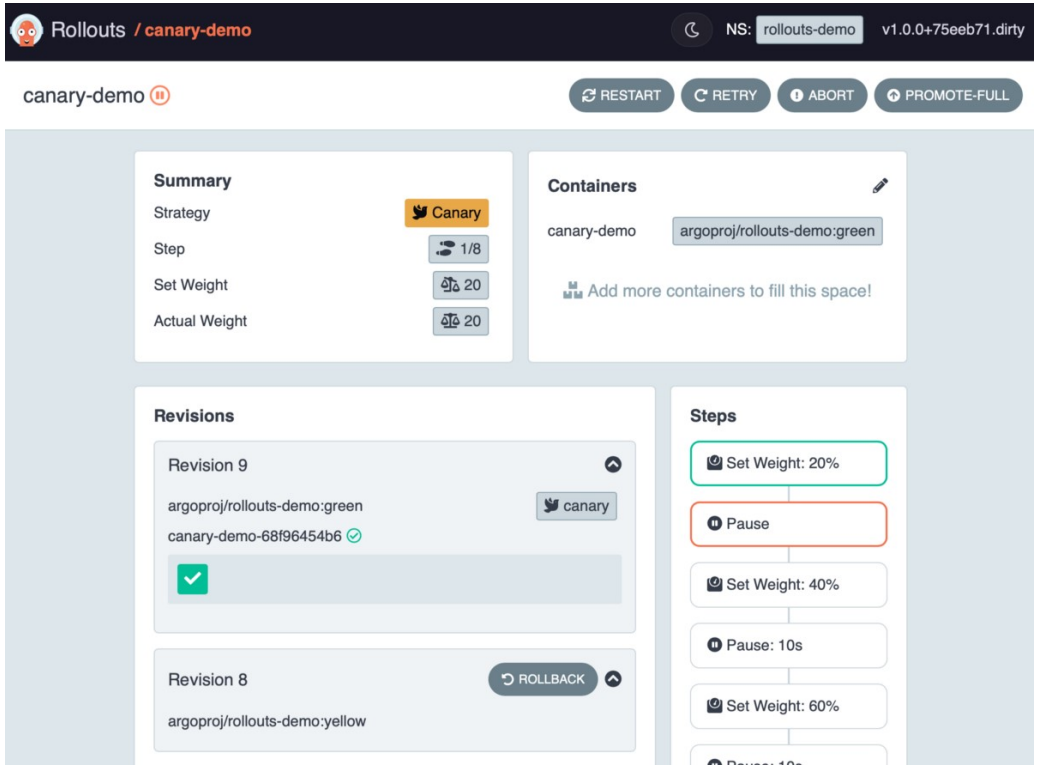
Argo Rollouts CLI

```
Name:            spring-sample-app-deployment
Namespace:       blue-green
Status:          ॥ Paused
Message:         BlueGreenPause
Strategy:        BlueGreen
Images:          kostiscodefresh/argo-rollouts-blue-green-sample-app:2b364ac (preview)
                 kostiscodefresh/argo-rollouts-blue-green-sample-app:main (stable, active)
Replicas:
  Desired:       2
  Current:       4
  Updated:       2
  Ready:         2
  Available:     2

NAME                                              KIND        STATUS        AGE    INFO
⟳ spring-sample-app-deployment                    Rollout     ॥ Paused      7d22h
├──# revision:29
│  └──⧉ spring-sample-app-deployment-5db99f8d9     ReplicaSet  ✔ Healthy     10s    preview
│     ├──□ spring-sample-app-deployment-5db99f8d9-bgtrt  Pod  ✔ Running     10s    ready:1/1
│     └──□ spring-sample-app-deployment-5db99f8d9-s728n  Pod  ✔ Running     10s    ready:1/1
├──# revision:28
│  └──⧉ spring-sample-app-deployment-86d59cbd9f     ReplicaSet  ✔ Healthy     5d21h  stable,active
│     ├──□ spring-sample-app-deployment-86d59cbd9f-7vs2m  Pod  ✔ Running     21h    ready:1/1
│     └──□ spring-sample-app-deployment-86d59cbd9f-9t67t  Pod  ✔ Running     21h    ready:1/1
├──# revision:27
│  └──⧉ spring-sample-app-deployment-7cd68d9965     ReplicaSet  • ScaledDown  45h
└──# revision:26
   └──⧉ spring-sample-app-deployment-6c5b7c6d99     ReplicaSet  • ScaledDown  5d21h
```

Argo Rollouts UI



SUMMER OF KUBERNETES

BROUGHT TO YOU BY AMBASSADOR LABS

Blue/Green deployments

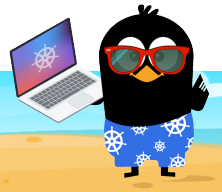
Using Argo Rollouts

Blue/Green deployments

1. The simplest way to start using Argo Rollouts
2. The major settings are the service for blue and for green
3. Active service is what your users will see
4. Preview service can be used for smoke test
5. You can pause the promotion or have a timeout

Blue/Green deployments

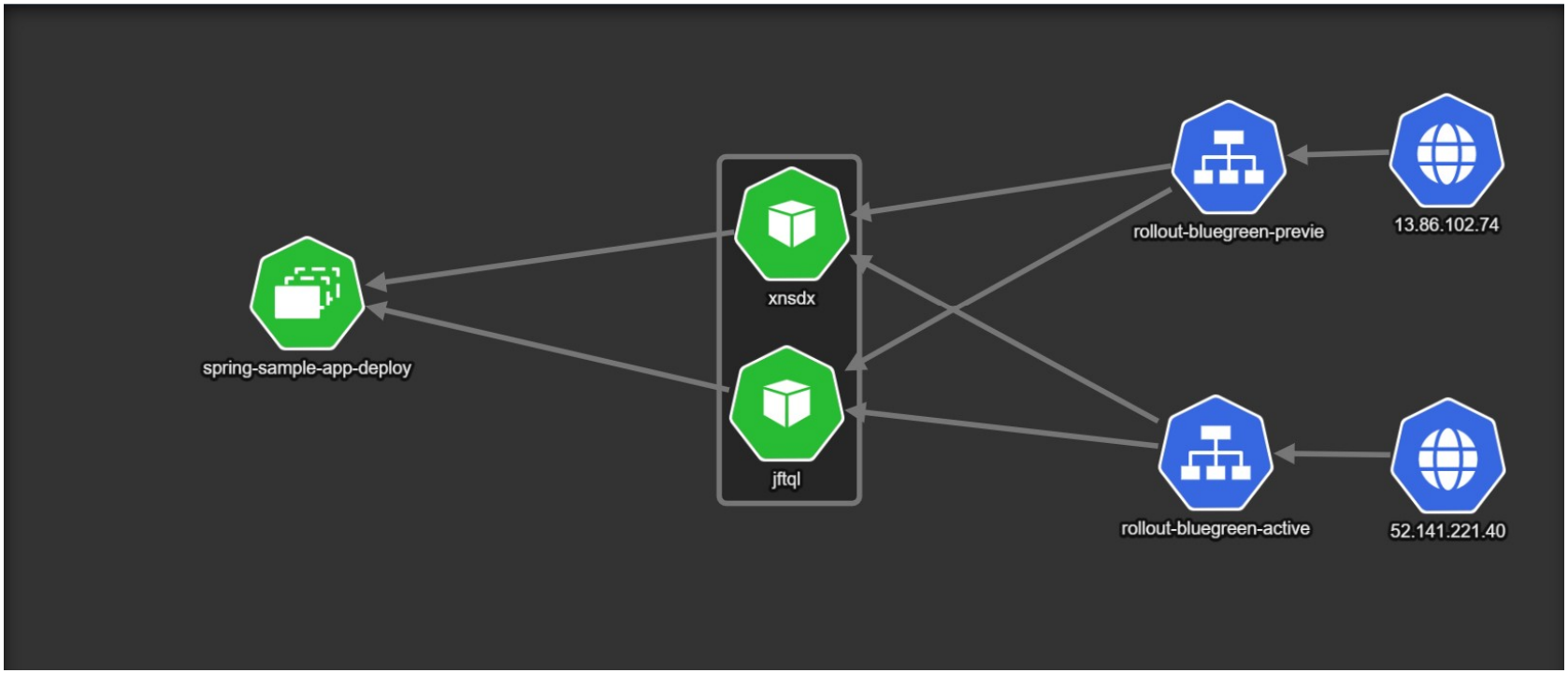
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: demo-app
spec:
  replicas: 2
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: demo-app
  template:
    metadata:
      labels:
        app: demo-app
    spec:
      containers:
        - name: application-container
          image: my-app-image:v1
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
  strategy:
    blueGreen:
      activeService: rollout-bluegreen-active
      previewService: rollout-bluegreen-preview
      autoPromotionEnabled: false
```

Blue/Green deployments

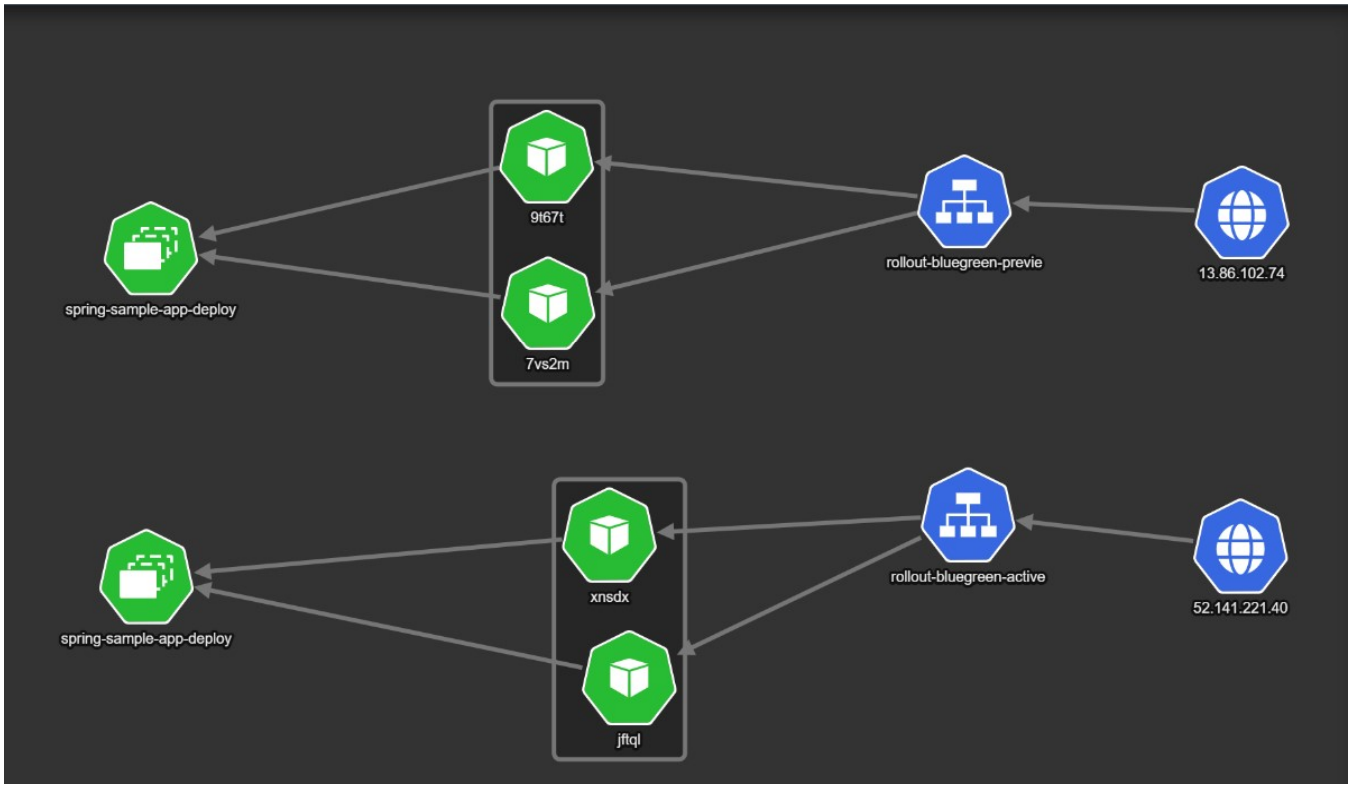
1. Change the image in rollouts
2. kubectl apply –f rollout.yaml
3. All your users see the old version
4. Run smoke tests on preview service
5. Use the “kubectl argo rollout promote” command to move everybody to the new version

Initial state

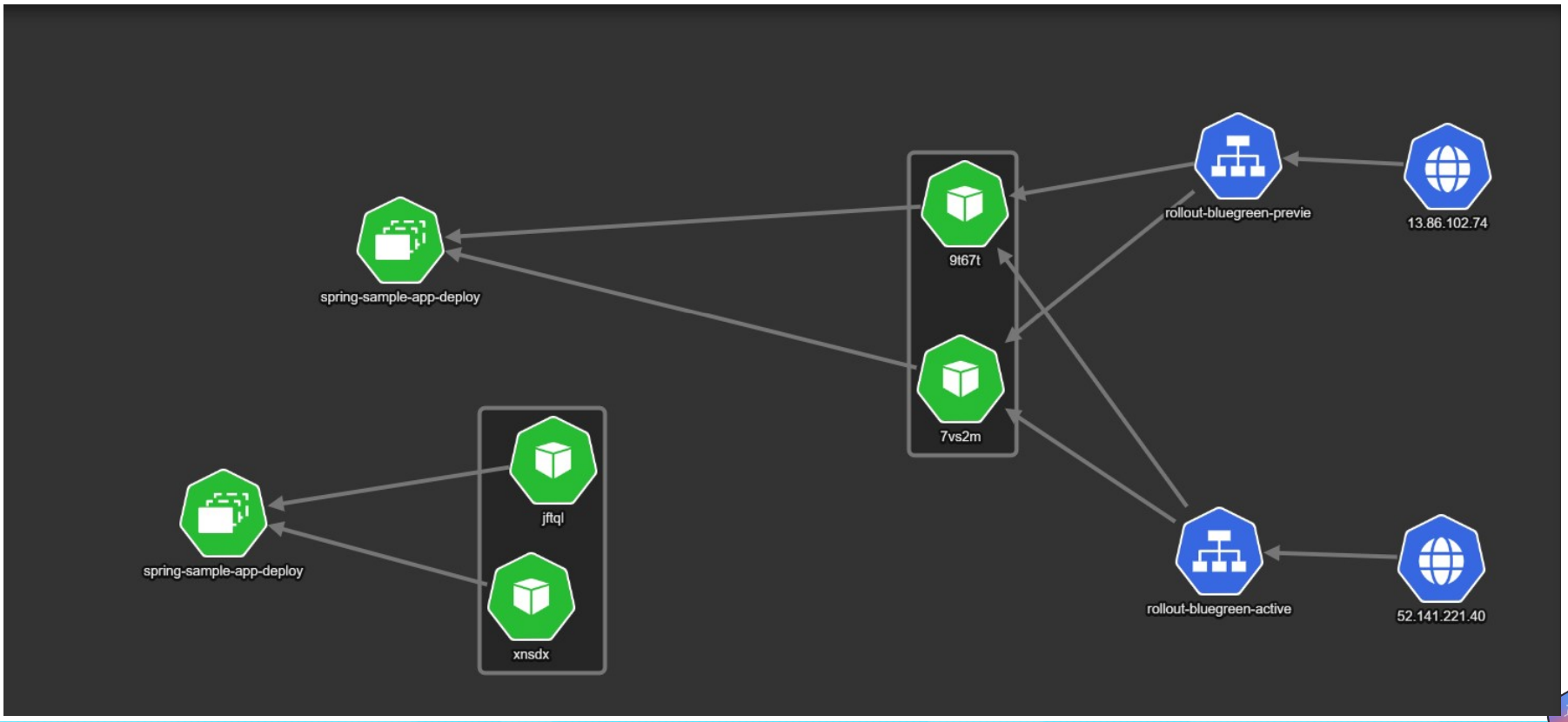


https://github.com/benc-uk/kubeview

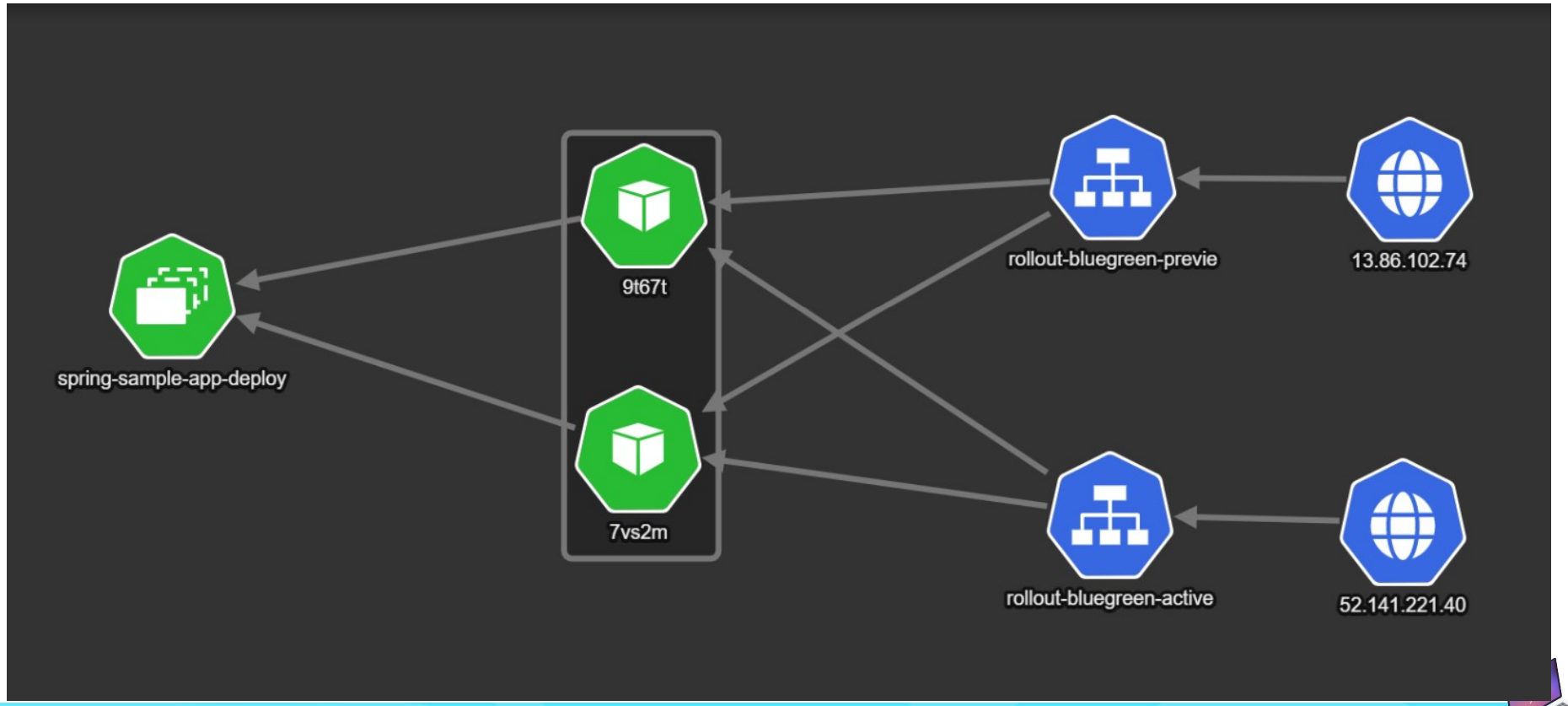
New version active (all users still on old version)



New version active (all users view new version)



Old version discarded (back to initial state)



SUMMER OF KUBERNETES

BROUGHT TO YOU BY AMBASSADOR LABS

Canary deployments

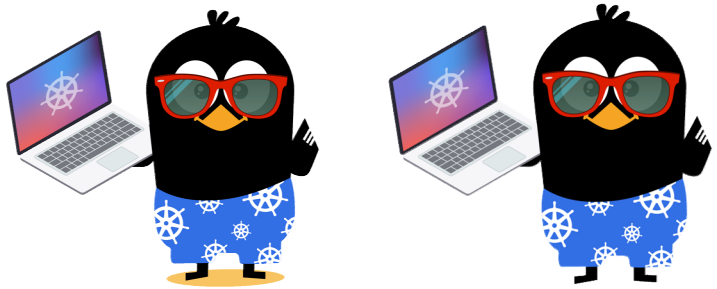
Using Argo Rollouts

Demo app

1. https://github.com/kostis-codefresh/summer-of-k8s-app-manifests
2. https://github.com/kostis-codefresh/summer-of-k8s-app

SUMMER OF KUBERNETES
BROUGHT TO YOU BY AMBASSADOR LABS

SUMMER OF KUBERNETES
BROUGHT TO YOU BY AMBASSADOR LABS



https://a8r.io/slack (at the #summer-of-k8s channel)

https://www.getambassador.io/summer-of-k8s/ship/week3/