**Deployment Strategies**

1. **Blue-Green:**

**You will be having 2 sets of your application with different versions, say blue & green and can switch the traffic between these two by shifting the service**

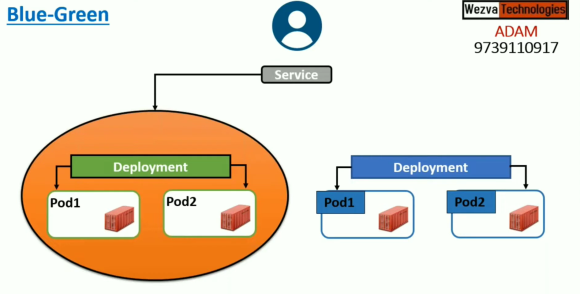
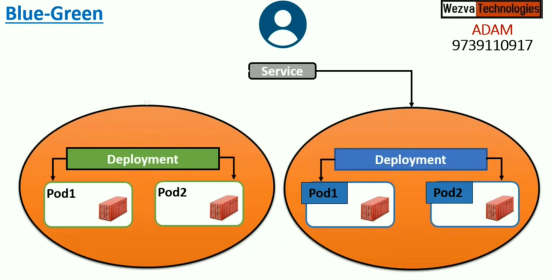
Consider green as your existing cluster where your application v1 Pods are running and serving the traffic currently.

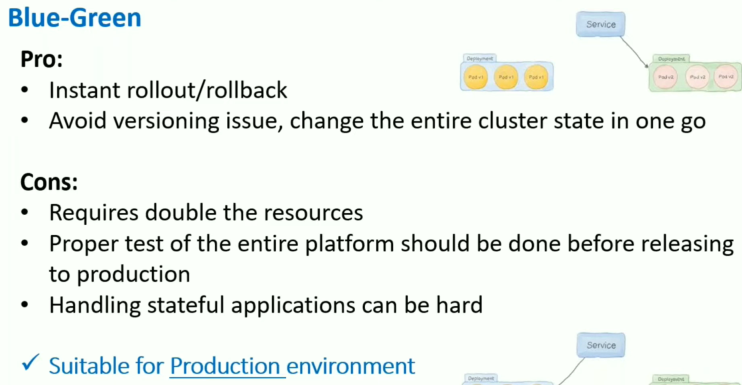
We create one more replica of your existing cluster in which we will be deploying app v2 Pods and keeping them up & running. But not serving the incoming traffic as far. This cluster we consider as blue.

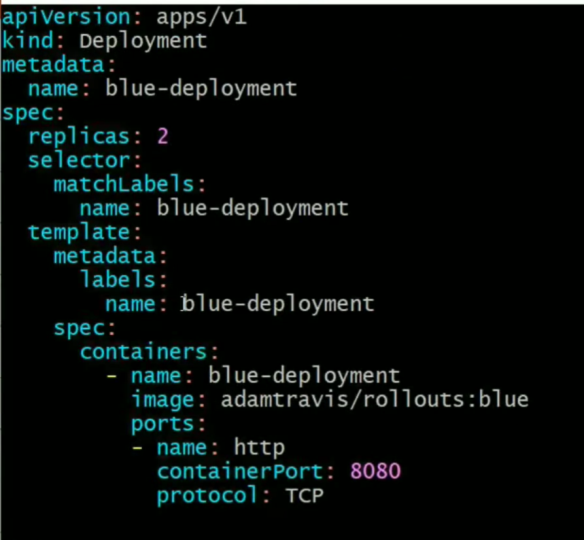
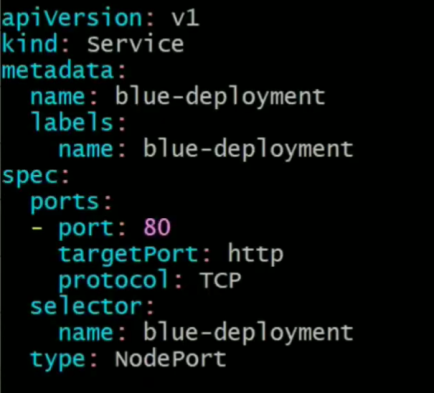
Now you can observe, both the versions of your application pods are setup and running, it’s just a time of seconds to shift the traffic to blue by shifting the service.

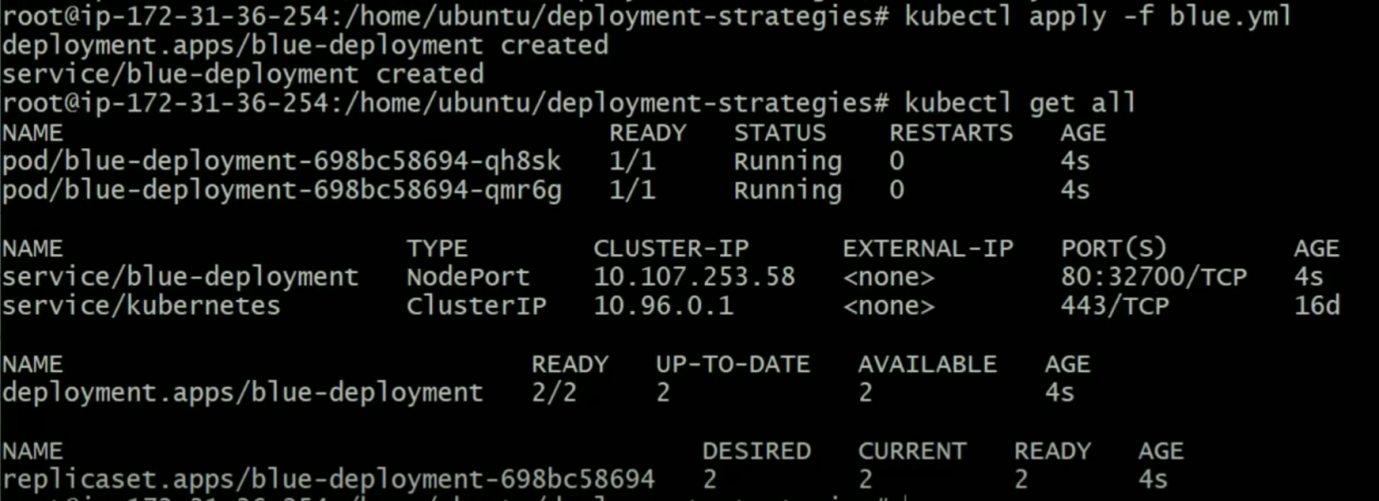
We just the replace the pod labels in the service with blue deployment Pod labels so that the service is routes the traffic to blue.

If there is any issue with blue application, we can simply rollback to green by shifting the service in fraction of sections.

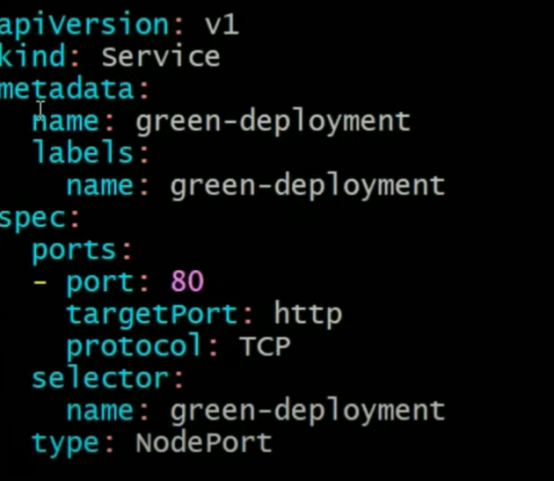
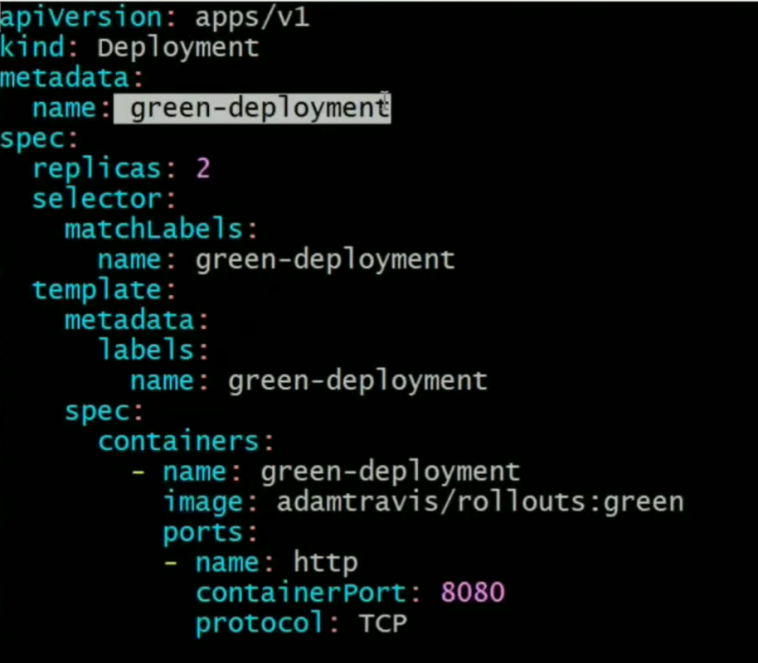


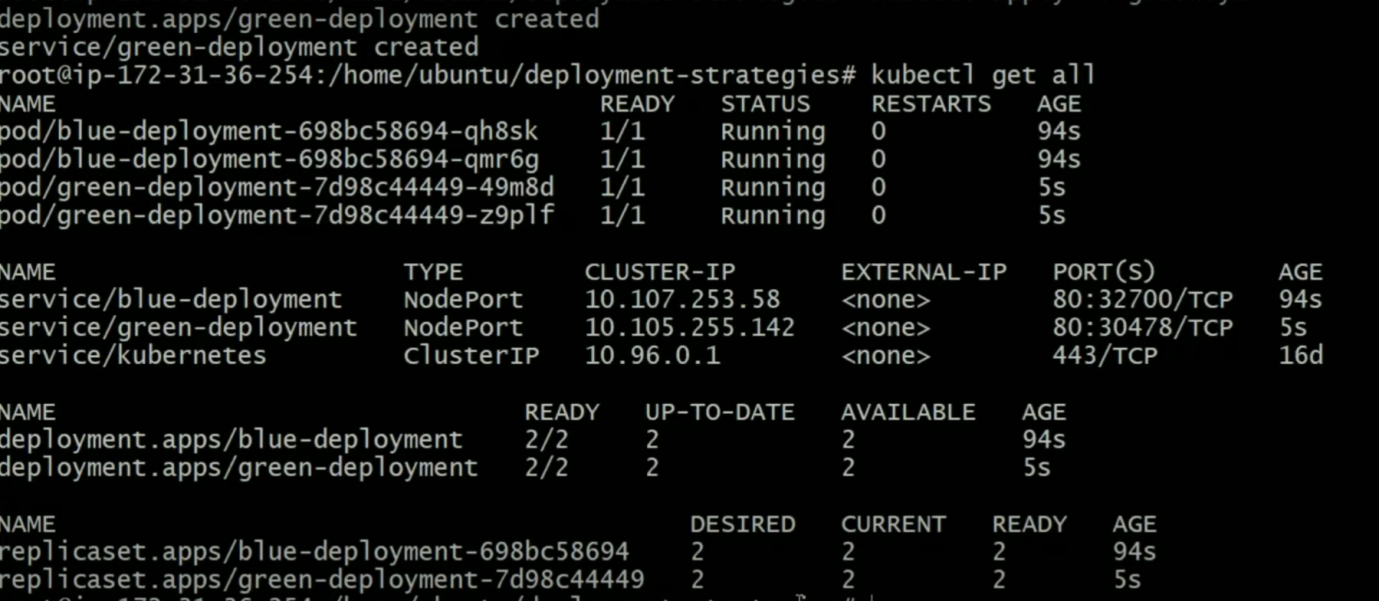
 



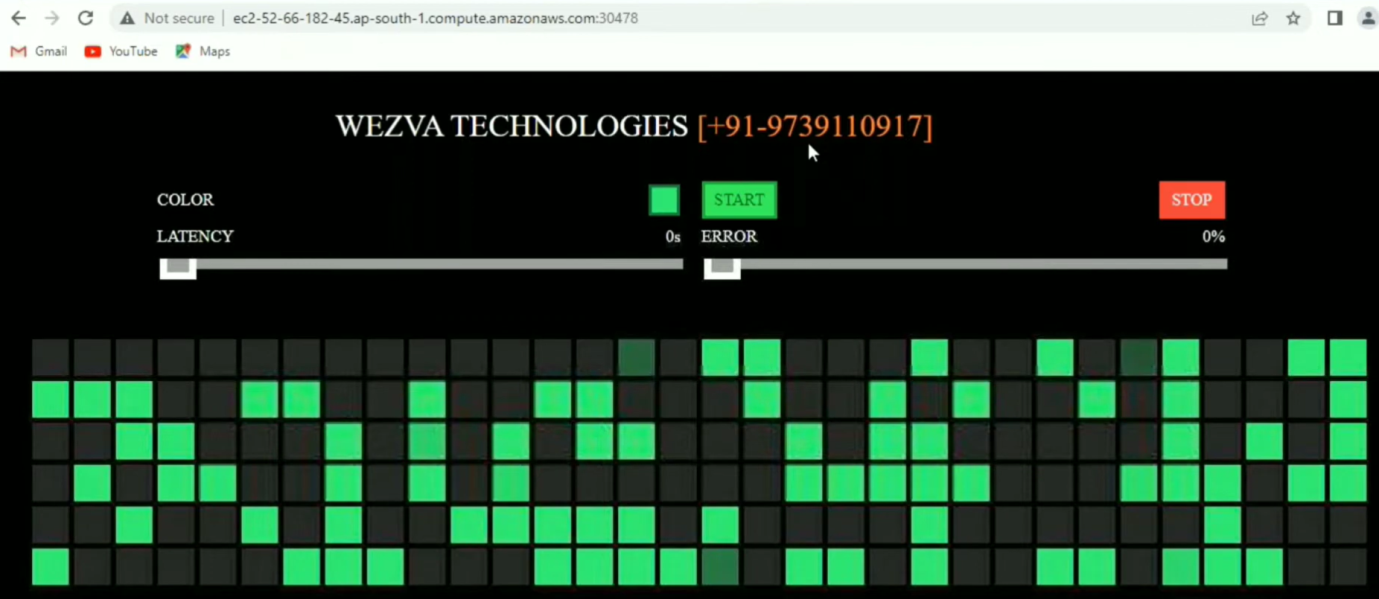
**You can access the blue application V2 with ec2 node IP and the port 32700**







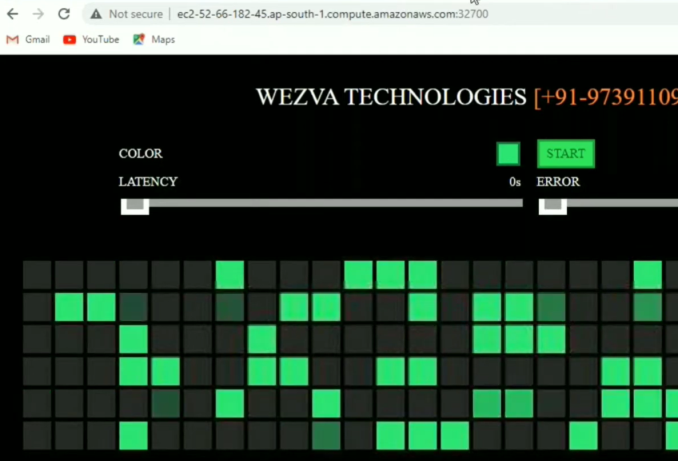
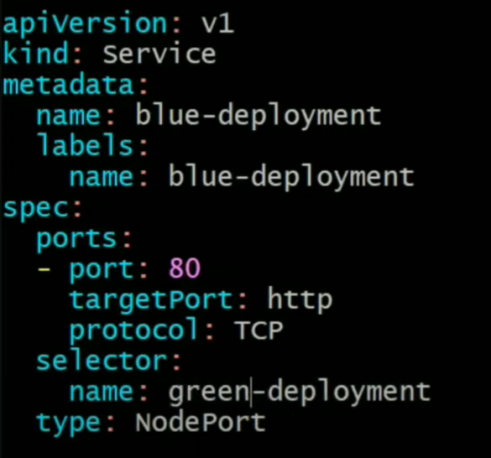
**You can access the green application with ec2 node IP and the port 30478**



Now you see, your 2 sets of application is running with different versions, Now it’s time to switch between the traffic. Just change the Pod labels in the Service(from blue🡪green or green🡪blue) that’s it.

If you face issue with **blue app v2**, just change pod labels to green in blue service **32700**.

Now the service refers to **green deployment V1** even on **32700**



If see any issue with **green app v1**, just change the Pod labels to blue in green service **30478**

Now the service refers to **blue deployment V2** even on **30478**

| **Version** | **App Port** | **NodePort** | **Description** |
| --- | --- | --- | --- |
| v1 | 80 | 32000 | Current live app |
| v2 | 80 | 31000 | New candidate app |

You're using **Ingress** and want to **dynamically route traffic to either v1 or v2** based on which version should be live.

**🛠️ Solution: Use Ingress with Kubernetes Service Abstraction**

Ingress doesn’t talk to NodePorts directly — it talks to **Services by name** (ClusterIP/NodePort/LoadBalancer). So, here’s how we do it:

**✅ Step-by-Step Approach**

**🧱 Step 1: Create Two NodePort Services**

# v1 Service (NodePort 32000)

apiVersion: v1

kind: Service

metadata:

name: myapp-v1

spec:

type: NodePort

selector:

app: myapp

version: v1

ports:

- port: 80

targetPort: 80

nodePort: 32000

# v2 Service (NodePort 31000)

apiVersion: v1

kind: Service

metadata:

name: myapp-v2

spec:

type: NodePort

selector:

app: myapp

version: v2

ports:

- port: 80

targetPort: 80

nodePort: 31000

**🧭 Step 2: Use a Shared Ingress but Switch the Target Service**

Your Ingress can dynamically route to either myapp-v1 or myapp-v2.

**🔄 Option A: Update Ingress Manually to Point to v2**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: myapp-ingress

spec:

rules:

- host: myapp.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: myapp-v2 # Switch to myapp-v1 or myapp-v2

port:

number: 80

💡 This allows you to **cut over from v1 to v2** instantly by updating one field in Ingress.

**🔄 Option B: Use a Stable Service and Swap Its Selector**

Create a **myapp-live** service that always routes to the live version:

# Stable Service

apiVersion: v1

kind: Service

metadata:

name: myapp-live

spec:

selector:

app: myapp

version: v1 # or v2

ports:

- port: 80

targetPort: 80

Then, your Ingress always points to:

backend:

service:

name: myapp-live

port:

number: 80

💥 Now to switch traffic:

* Just update the selector.version in myapp-live service from v1 to v2.

✅ This avoids touching Ingress and keeps routing logic clean.

**🌐 Bonus: Weighted Routing with Ingress (e.g., for Canary)**

If you’re using **NGINX Ingress + service mesh like Istio or Linkerd**, you can do **traffic splitting** like:

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: myapp-v1

weight: 30

- path: /

pathType: Prefix

backend:

service:

name: myapp-v2

weight: 70