**🟦🟩 What Is Blue-Green Deployment?**

**✅ Concept:**

* You have **two identical environments**: blue (live) and green (new).
* Only one is live at a time.
* When you want to deploy a new version, you deploy it to the **inactive environment**.
* After testing, you **switch all traffic** from blue to green

**✅ Example:**

* Blue (v1) is live.
* You deploy v2 to Green.
* After testing, flip DNS or load balancer to Green

**✅ Pros:**

* Easy rollback: just switch back to Blue.
* No partial users see broken updates.
* Simple to understand

A screenshot of a phone

AI-generated content may be incorrect.

**🟡🟠 What Is Canary Deployment?**

**✅ Concept:**

Release new version to a **small percentage of users first** (e.g., 5% or 10%). If it works well, gradually roll out to more.

Just like the *canary in the coal mine*—you check if the environment is safe before sending everyone in.

**✅ Example:**

* v1 is live to 100%.
* Deploy v2 alongside it.
* Send **10% traffic to v2** (canary), **90% to v1**.
* Monitor logs, metrics, errors, etc.
* Gradually increase to 25%, 50%, 100% (full rollout).
* If something breaks, just kill v2 and go back to 100% v1.

A diagram of a computer program

AI-generated content may be incorrect.

**🔧 How Canary Works (in Kubernetes)**

**Here’s how it works behind the scenes:**

🧱 You have:

* Two Deployments: v1 (stable), v2 (canary)
* Two Services: routing traffic to each deployment
* Ingress: decides how to split traffic

**🎯 Traffic splitting:**

**Using NGINX ingress annotations like:**

annotations:

nginx.ingress.kubernetes.io/canary: "true"

nginx.ingress.kubernetes.io/canary-weight: "10"

**This tells NGINX:**

**Route 90% of requests to the stable version, and 10% to the canary version.**

**You can also split based on:**

* Weight (percentage)
* Header (e.g., only users with X-User-Type: beta)
* Cookie (target beta users)

**Use Case**

"In our example, we have two versions of a simple NGINX-based app:"

* myapp-v1 (Stable – 90% traffic)
* myapp-v2 (Canary – 10% traffic)

"Both apps serve different HTML content from ConfigMaps, so we can visually see which version we’re hitting."

**How It Works**

"We use the **NGINX Ingress Controller**, which supports canary deployments natively through annotations."

"We define two Ingress resources:"

1. **Main Ingress** (for v1, stable)
2. **Canary Ingress** (for v2, with annotation canary-weight: 10)

"Both Ingresses use the same path /, so traffic is split based on the configured weight."

**🛡️ Optionally: Use Flagger for Automated Canary**

If you want intelligent rollout with metrics (success rate, latency, etc.), use [Flagger](https://flagger.app) with Prometheus.

**Best Practices**

* Enable **logging and metrics** to detect anomalies.
* Integrate with **Azure Monitor / Log Analytics** for observability.
* Keep the canary small at first (e.g., 5%-10%).

**✅ 1. What to Monitor**

When 10% of traffic is routed to the canary (new version), you monitor:

A screenshot of a computer

AI-generated content may be incorrect.

✅ You can collect these using:

* **Fluent Bit + Azure Log Analytics / ELK**
* **Prometheus + Grafana**
* **OpenTelemetry + Azure Monitor**
* **Application Performance Monitoring tools** (like New Relic, Dynatrace)

**📊 2. Real-time Canary Observability with Prometheus**

Here’s how to observe error rates and request count in Prometheus:

# Error rate for v2

rate(http\_requests\_total{app="myapp", version="v2", status=~"5.."}[1m])

# Total traffic comparison

rate(http\_requests\_total{app="myapp", version="v1"}[1m])

rate(http\_requests\_total{app="myapp", version="v2"}[1m])

You can add Grafana dashboard panels to alert when:

* Error rate > 1%
* Response latency > 300ms
* Request success ratio < 99%

**🔁 3. Rollback Canary — 2 Ways**

**🔸 Option 1: Rollback via Ingress (Instant Switch)**

You can rollback by **removing the canary Ingress**, or setting canary-weight to 0.

kubectl patch ingress myapp-ingress-canary \

-p '{"metadata":{"annotations":{"nginx.ingress.kubernetes.io/canary-weight":"0"}}}'

Or

kubectl delete ingress myapp-ingress-canary

✅ This instantly stops all traffic to v2 and sends 100% to v1.

**🔸 Option 2: Rollback Deployment**

If you used a **new version in the same Deployment**, you can:

kubectl rollout undo deployment myapp-v2

But usually in canary strategy, you deploy v2 in a **separate Deployment**, so rollback is just deleting it:

kubectl delete deployment myapp-v2

**✅ 4. Automate Alert-based Rollback (Optional)**

Use tools like **Argo Rollouts**, **Flagger**, or CI/CD pipelines that:

* Continuously monitor metrics
* Automatically rollback if:
  + Error rate > threshold
  + Latency increases
  + Alert triggers