HK Technights: DevOps & SRE Meetup

DevOps
Lightning Talk

Zero Downtime Deployment Strategies

Guidemy

Objectives

After this module you will be able to:

- Understand the importance of deployment strategies in software development.
- Recognize the different deployment strategies available to software engineers, including their benefits and drawbacks.
- Know how to choose the right deployment strategy based on the specific needs of the application.

- Overview
- Recreate
- Ramped
- Canary
- A/B Testing
- Blue/Green
- Shadow

Guidemy

What are Deployment Strategies?

- They determines **how** new features, bug fixes, and other changes are **pushed into production** environments.
- Today, we will explore some of the most common deployment strategies, their **benefits**, and **potential issues**, including:
  - Recreate
  - Ramped / Rolling Update
  - Canary
  - A/B Testing
  - Blue/Green
  - Shadow

- Overview
- Recreate
- Ramped
- Canary
- A/B Testing
- Blue/Green
- Shadow

Guidemy

Recreate Deployment

- A *simple but risky* method of deployment
  - Stop the current version, deploy the new version and reboot
- **Benefits**:
  - Cheap and easy to set up – No need for a load balancer
- **Limitations**:
  - Downtime for both upgrade and rollback processes
  - If something goes wrong, the application will be offline until the previous version is restored
  - Not suitable for applications that need to be available 24/7

Overview

Recreate

Ramped

Canary

A/B Testing

Blue/Green

Shadow

Guidemy

Ramped Deployment

- Also known as the *Rolling* Deployment Strategy
- Applicable only if there are *multiple instances* of an application
- Replace instances of the old version with the instances of the new version one instance at a time
- **Benefits**:
  - Downtime is minimized
  - Allows for RFB checks and performance monitoring (only after going live?)
- **Limitations** :
  - The application must support running different versions at the same time
  - *Rollback duration* can be long (again, one instance at a time)
  - Performing rolling update *manually* can be *time-consuming* and *error-prone* (use orchestration tools, e.g. Kubernetes)

- Overview
- Recreate
- Ramped
- Canary
- A/B Testing
- Blue/Green
- Shadow

Guidemy

Canary Deployment

- Deploy the new version to a small subset of users, and test it for a period before gradually increasing the number of users that receive the update (e.g. 25% → 50% → 75% → 100%)
- Mostly used when there is little confidence about the stability of the new release
- **Benefits**:
  - Allows for real-world testing without risking the entire user base
- **Limitations** :
  - Slow in nature
  - Ensuring a specific amount of users receive the update can be difficult to implement
- Selecting the target audience who gets the updates first are more of a business decision than a technical decision

Origin of Canary Deployment

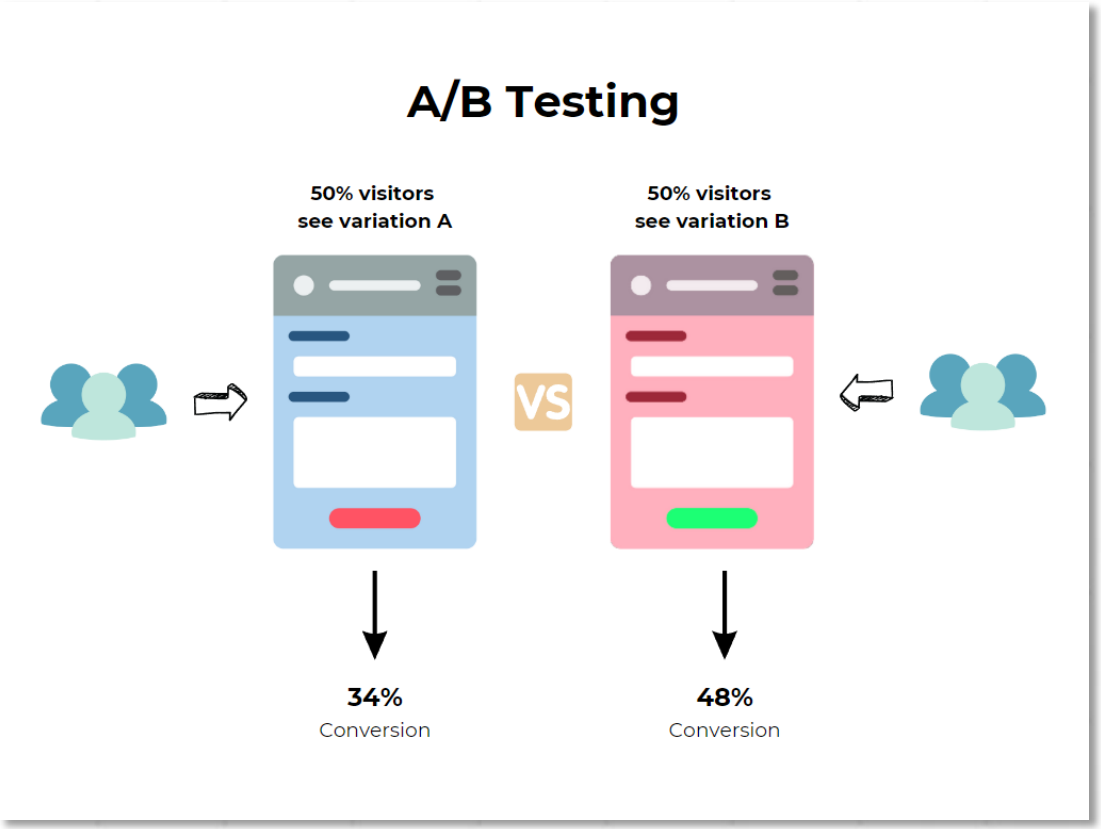
- The origin of canary testing is from **coal mining**, where miners would bring canaries, **a type of small, yellow songbird**, into the mines to **detect dangerous gases**.
- The presence of dangerous gases would kill the canary **before** it could kill the miners.
- If the canaries became ill or died, it was **a sign of danger** for the miners to evacuate.
- Similarly, the end-users testing the new application version also provide **an early warning for any glitches** before the feature is released for all users.

- Overview
- Recreate
- Ramped
- Canary
- A/B Testing
- Blue/Green
- Shadow

Guidemy

A/B Testing Deployment

- Deploy two different versions to different subsets of users, then compare to see which version performs better
- Mainly used to get user feedback about changes, such as new features or user interface changes
- How to split the traffic is a business decision
- Benefits
  - Several versions run in parallel which allows for real-world testing
- Limitations
  - Requires intelligent (and often pricey) load balancer
  - Hard to troubleshoot errors (distributed tracing required)
- Can get simpler by making use of feature flags



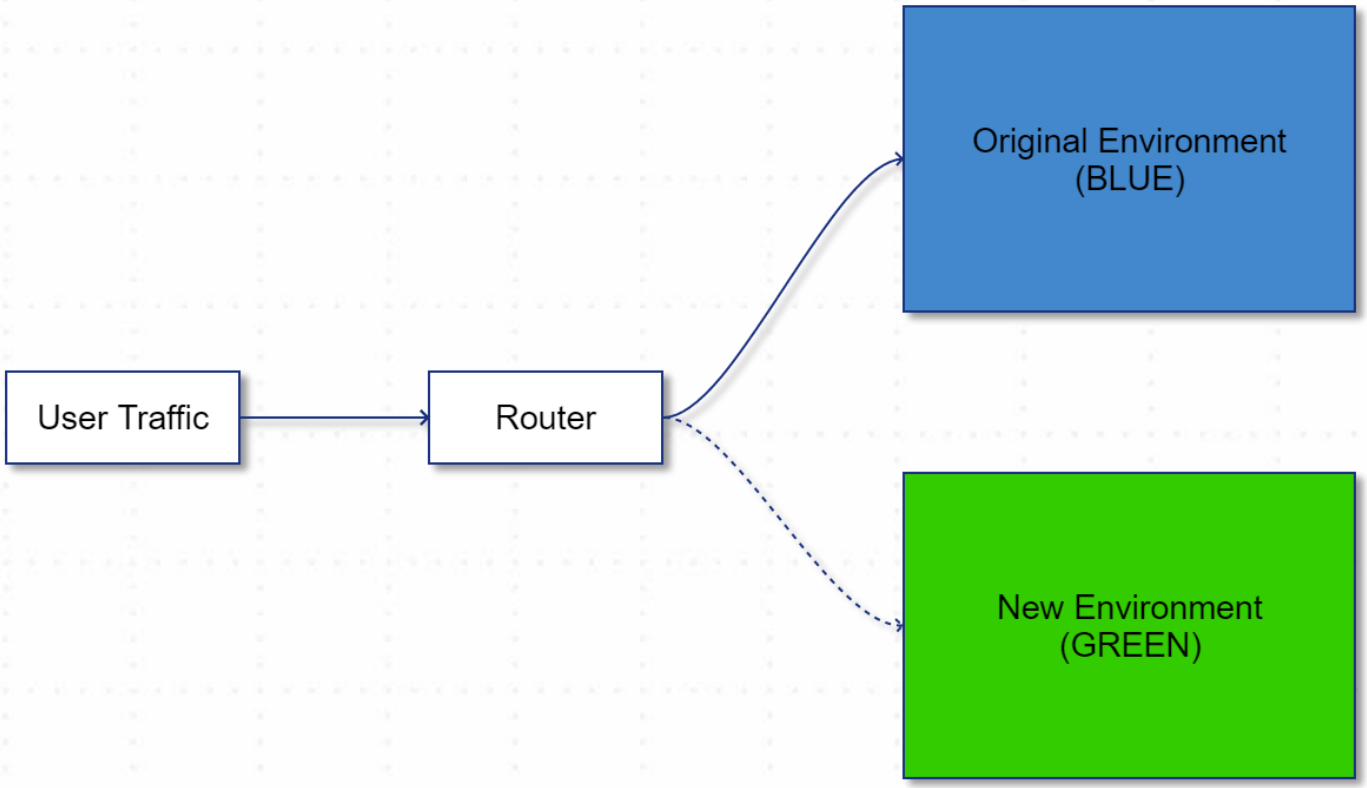
- Overview
- Recreate
- Ramped
- Canary
- A/B Testing
- **Blue/Green**
- Shadow

Guidemy

Blue/Green Deployment

- Have two identical environments: **one active (Blue)** and **one inactive (Green)**
  - First, deploy the new version on the inactive environment (Green), then switch the traffic from the active environment (Blue) to the inactive environment (Green) using a load balancer.
  - Rename the updated version from Green to Blue, and the old version from Blue to Green.
  - Repeat the same procedure again in the next release.
- **Benefits**:
  - Instant rollout / rollback – Zero downtime
  - Avoid versioning issue, the entire application state is changed in one go
- **Limitations** :
  - Expensive as it requires double the resources
  - Handling stateful applications can be hard
  - Beware of non-backward-compatible changes when rolling back
  - The deployment mechanism itself should be well-tested before releasing to production

Blue/Green Deployment



- Overview
- Recreate
- Ramped
- Canary
- A/B Testing
- Blue/Green
- Shadow

Guidemy

Shadow Deployment

- Very similar to the Blue/Green strategy, except:
  - Both environments receive the request, but only responses from the original version go back to the users.
- Useful for testing new versions of an application without affecting users
- **Benefits**:
  - Allows for performance testing with production traffic
  - No impact on users and no rollout until stability and performance are satisfactory
- **Limitations** :
  - Expensive and complex to set up and can create serious issues, e.g. charging a user twice (one solution is use mock services on the Shadow version)

Comparison of Deployment Strategies

Strategy	Description	Zero Downtime	Real Traffic Testing	Targeted Users	Cloud Cost	Rollback Duration	Negative Impact on User	Complexity of Setup
Recreate	Version A is terminated then version B is rolled out	N	N	N	*	***	***	-
Ramped	Version B is slowly rolled out and replacing version A	Y	N	N	*	***	*	*
Canary	Version B is released to a subset of users, then proceed to a full rollout	Y	Y	N	*	*	*	**
A/B Testing	Version B is released to a subset of users under specific condition	Y	Y	Y	*	*	*	***
Blue/Green	Version B is released alongside version A, then the traffic is switched to version B	Y	N	N	***	-	**	**
Shadow	Version B receives real wolrd traffic alongside version A but doesn't impact the response	Y	Y	N	***	-	-	***

Summary

- When it comes to production:
  - **Ramped** or **Blue/Green** Deployment is usually a good fit, but proper testing of the new version is necessary.
- **Blue/Green** and **Shadow** deployment have more impact on the budget as it requires double the resources.
- If the application lacks in tests or if there is little confidence about the impact/stability of the new version, consider:
  - **Canary, A/B Testing or Shadow Deployment**
- If your business requires testing of a new feature amongst **a specific pool of users** that can be filtered depending on **demographic factors** such as geo-location, language, operating system or device, then consider using the **A/B Testing technique**.