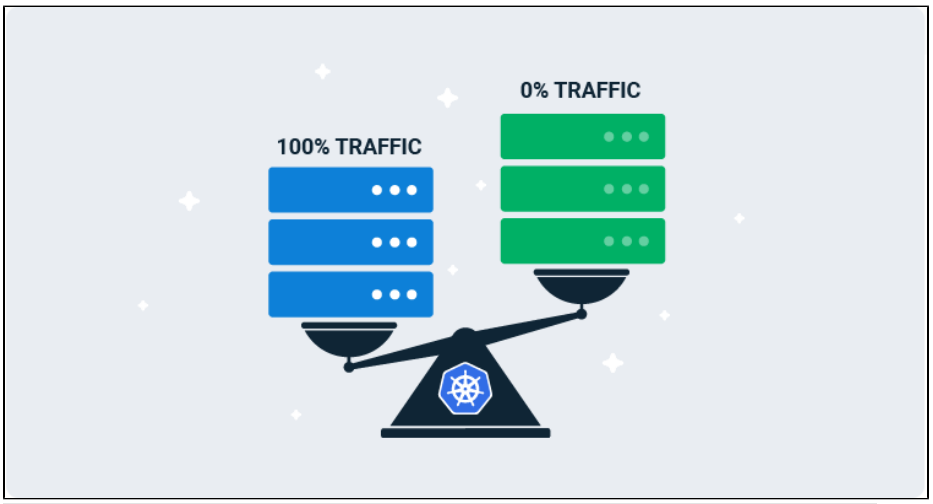
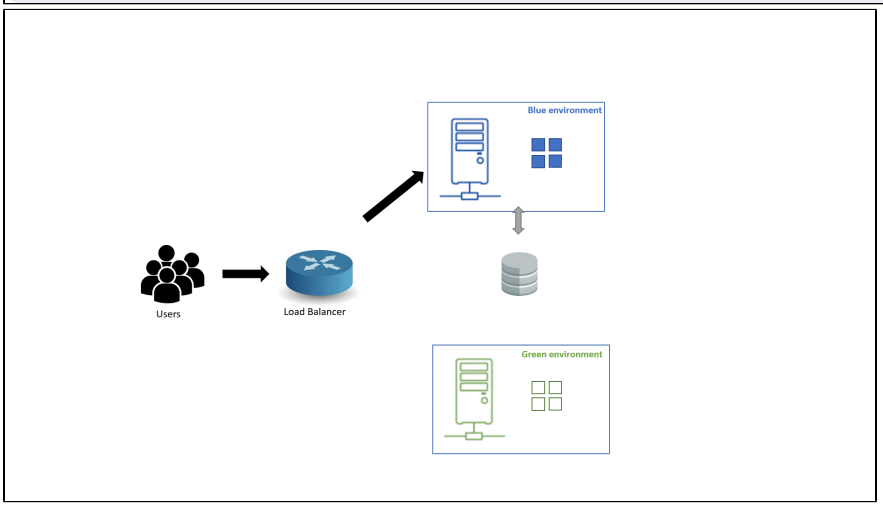
Blue-Green Deployment {Kubernetes [AWS EKS]}





Introduction

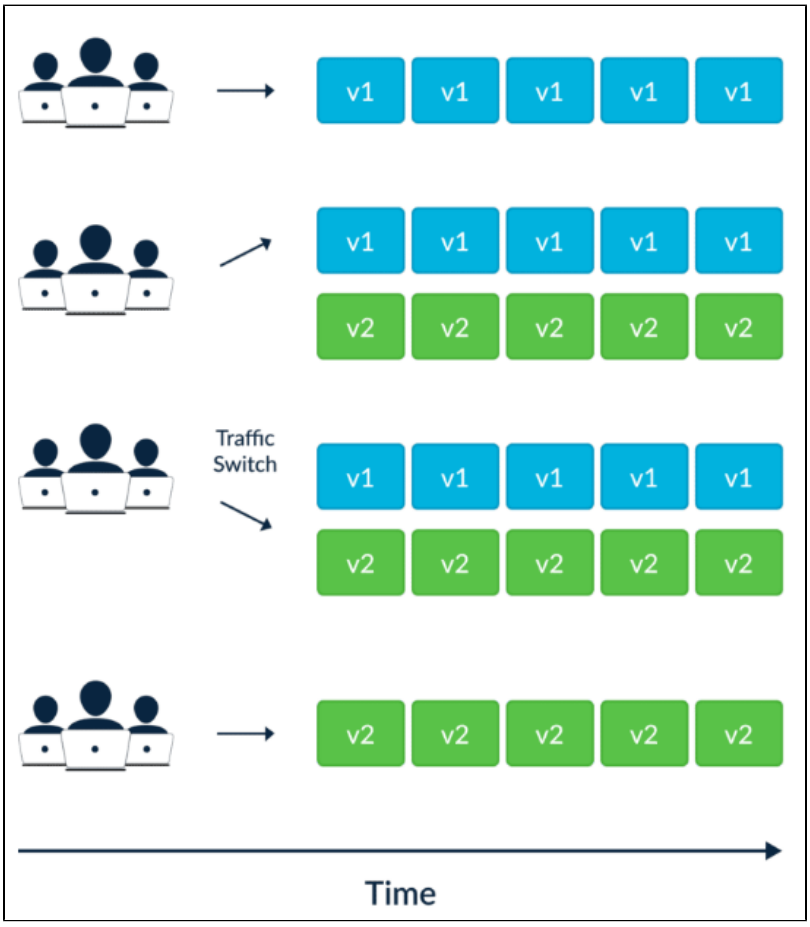
Organizations use modern application development approaches, such as microservices, to increase innovation, performance, security, and reliability. However, when working with legacy deployment systems, it can be difficult to maintain a fast deployment pace while maintaining control and security over each deployment. As a result, customers often turn to orchestration systems like Amazon Elastic Kubernetes Service (Amazon EKS) or Amazon Elastic Container Services (Amazon ECS) to manage their application workloads. Continuous Delivery is also a popular principle that helps customers define best practices for deploying applications and infrastructure, reducing risks, costs, and improving product quality and time to market. Applications running on AWS container services can be exposed through AWS Load Balancers and their associated domain names can be managed using Amazon Route 53.

What Is Blue/Green Deployment

- The gist of blue-green deployments is to have two identical environments, conventionally called blue and green, to do continuous, risk-free updates.
- The blue environment runs the existing software version, while the green environment runs the new version.
- Only one environment is live at any time, receiving all production traffic. This way, users access one while the other receives updates.
- Once the new version passes the relevant tests, it is safe to transfer the traffic to the new environment.
- If something goes wrong, traffic is switched to the previous version.

How Does a Blue/Green Deployment Process Work

1. **Deploy new version**—deploy the new (green) version alongside the current (blue) version. Test it to ensure it works as expected, and deploy changes to it if needed.
2. **Switch over traffic**—when the new version is ready, switch overall traffic from blue to green. This should be done seamlessly so end-users aren't interrupted.
3. **Monitor**—closely monitor how users interact with the new version and watch out for errors and issues.
4. **Deploy or rollback**—if there is a problem, immediately roll back by switching traffic back to the blue version. Otherwise, keep traffic on the green version and continue using it.
   The green version now becomes the blue (current) version, and a new version can be deployed alongside it as the "new green" version.



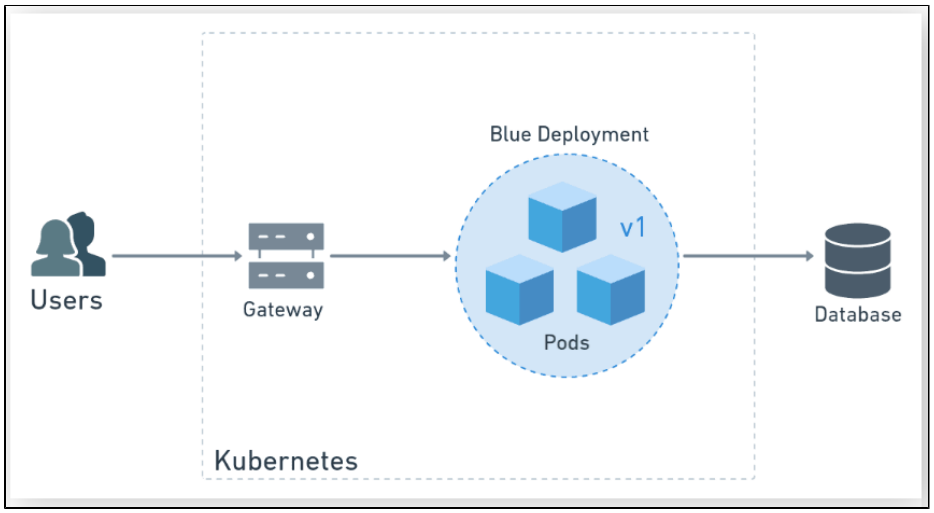
Solution

- Create two clusters (eks-blue and eks-green) that share the same VPC and use the AWS Load Balancer Controller and the External-DNS add-ons, to expose our applications.
- eks-blue on v1.22 and eks-green on v1.23
- Terraform (IaC) creates the clusters, installs the add-ons, and configure teams.
- Allow platform team to automate the workloads migration from a Blue Amazon EKS cluster to a Green EKS cluster.

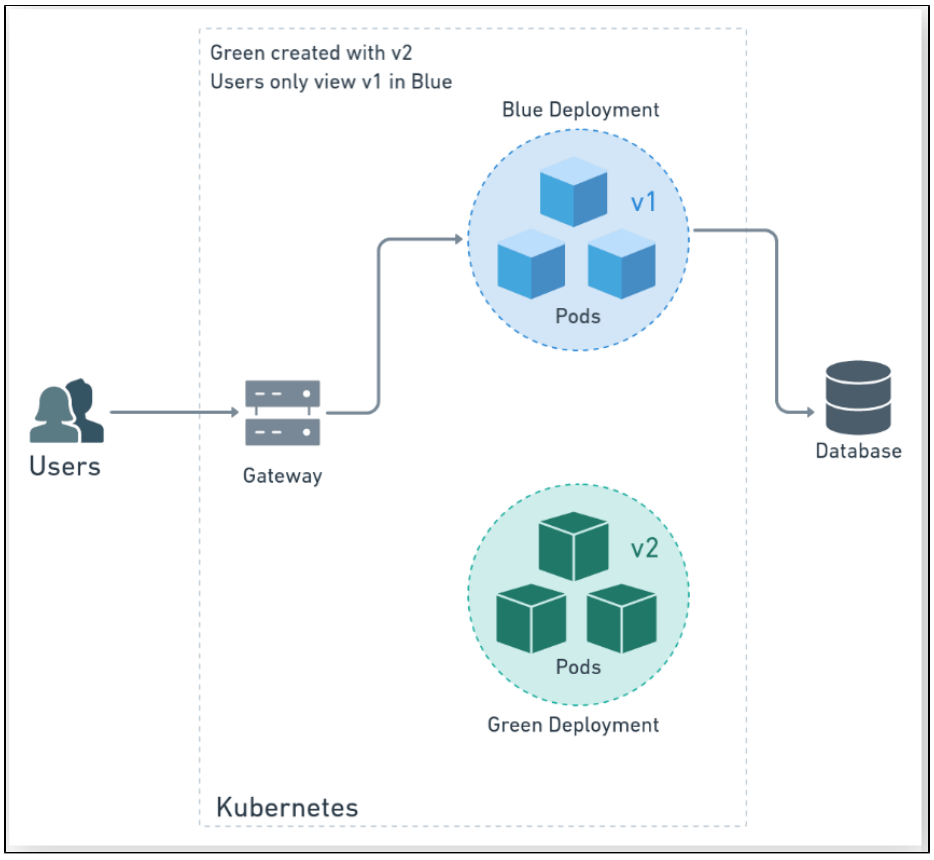
There are two types of teams we consider here:
The **Platform team** is responsible for the infrastructure automation code, and manages the Amazon EKS clusters (i.e., they have administrative rights).
The **Application teams** are responsible for the build and deployment of their workloads, with each team associated with a Kubernetes namespace (i.e., they have read-only rights in their namespace).

Blue/Green Deployments with Kubernetes

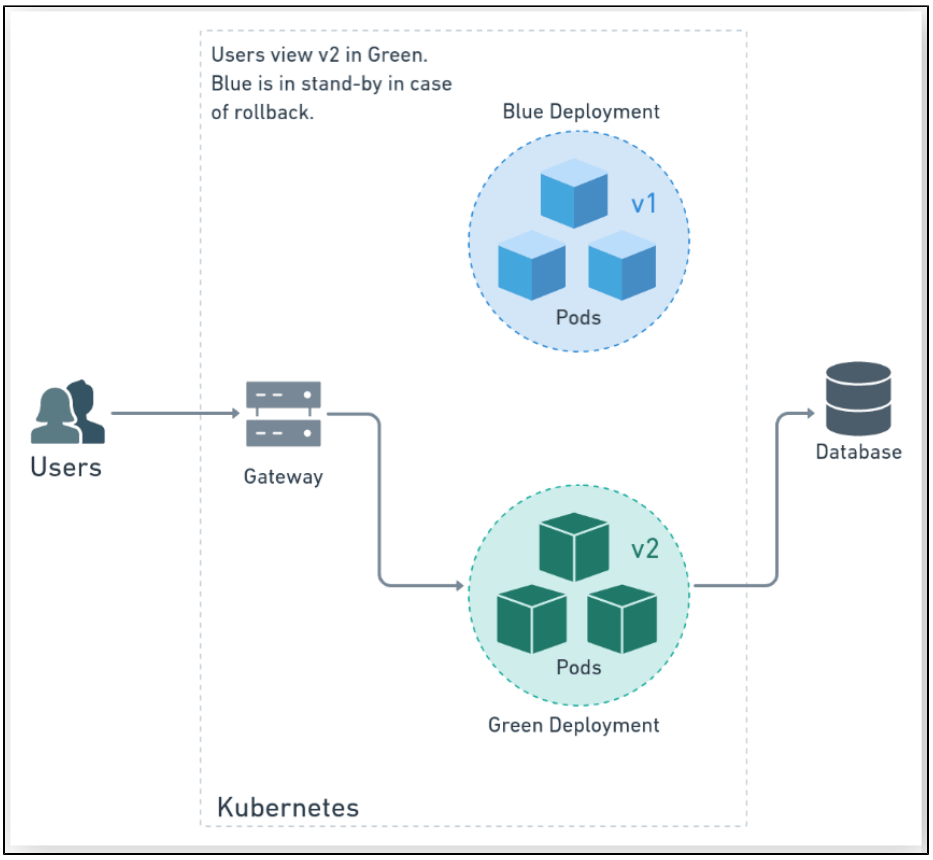
In Kubernetes, we run applications with deployments and pods.



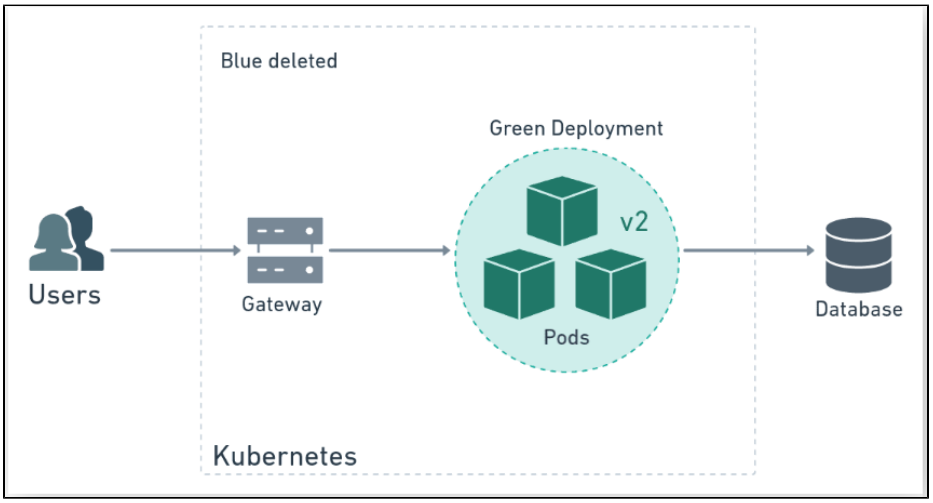
Create a brand-new production environment called green. Declare a new deployment, and the k8s platform takes care of the rest.
Users are not yet aware of the change as the blue environment keeps on working unaffected. They won't see any change until we switch traffic over from blue to green.



Test green on the same Kubernetes cluster where blue is running.



Once the users moved from blue to green and happy with the result, delete blue to free up resources.



Deployment Strategies available in Kubernetes

Recreate deployments delete existing pods before creating new pods
Use this option when container versions cannot be mixed. This strategy does result in downtime.

Rolling update deployments deploys new pods while remove older pods
This option requires that two container versions can run side by side, and avoids downtime.

Blue/Green deployments create a new deployment resource and points the service to new pods.
This strategy requires that two container versions can run side by side and ensures that traffic is cut over to the new pods in a single operation with no downtime.

Step	Description	Environment Specific Requirements	References
Deploy changes to green environment	The first step for a blue/green deployment is to deploy the new changes to the new green environment. This should eventually be a full build to ensure repeatability.	• Separate Jira ticket linked to original preprod work item • Change request required • PR required with approval • Deployment plan	Full Cluster Deployment Process:
Test green environment	A test plan and schedule will need to be arranged with the testers and executed against the green environment.	• Jira ticket • Test Plan	
Switch blue to green	Once the green environment had been tested and soaked for 1 - 2 weeks (minor - major changes) it can be switched over from blue to green. This process is detailed under the Switching from Blue to Green section of this document.	• New CR • Jira Ticket • PR with Approval	Switching from Blue to Green
Monitor green	Now that the blue environment is running the new changes, there should be some light touch monitoring to ensure platform stability. Initially the focus should be on monitoring the relevant slack alert channels. In addition to this the testers should also run some light tests to ensure the UI is functioning as expected.	• Periodic checks	Production Slack Channel: #prod-automation-engineering-integration-support
Shutdown blue /old green	To shutdown the blue/old green cluster, the auto-scaling groups desired counts will need to be set to 0. This should be maintained using Terraform so that git is a source of truth of the cluster state.	• New CR • Jira ticket • PR with approval	

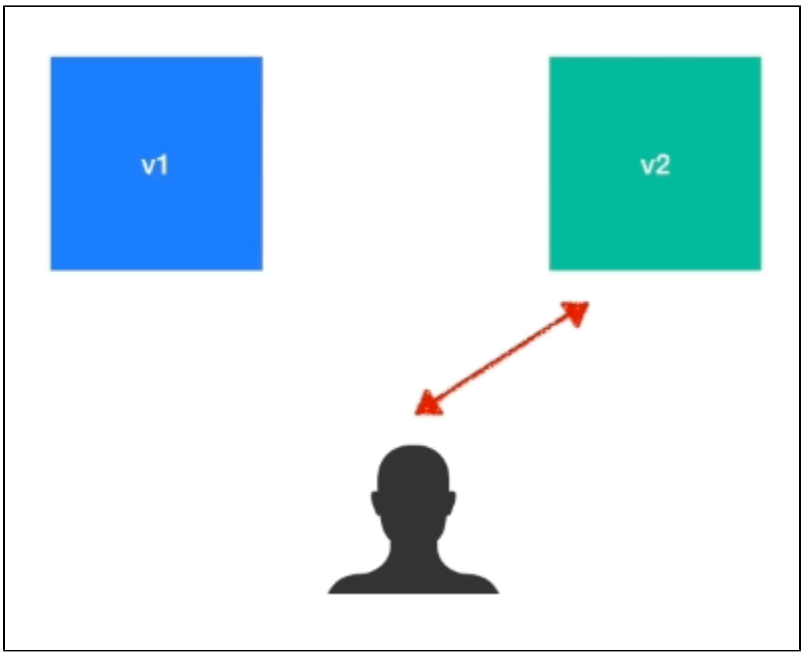
Key Points

- Both preprod and production blue/green deployments should be more or less identical with the key differences being environment and store data.
- Soaking should be done against the blue environment for at least 1 week for minor changes and 2 weeks (or 10 days including weekend) for major changes.
    - Minor changes are changes that have a minimal impact on the differences between the environments i.e. basic configuration changes
    - Major changes are those that introduce significant differences between the environments i.e. k8s version upgrades, security patches etc..
- TEST and SRVCES blue deployments can be done side by side. If there are any issues with these deployments they will be identified under the blue environment reducing the impact.
- The dev environment should be used for testing cluster state against major changes. It's also best-practice to rebuild the entire cluster when performing major changes so that the process is repeatable before moving to TEST.

Switching from Blue to Green

To switch a cluster or instance (TEST) from blue to green, a DNS change will need to be made against the two common names relevant to each environment.
These common names are listed under the Environment Layouts section below.

DNS is currently managed using Route53 and have access to make changes.



Updating Route53 Records

<>

Script Files

File	Purpose

Execution

Preprod - Blue	Preprod - Green
`CodePipeline-Blue`	`CodePipeline-Green`

Environment Layouts

Preproduction(SRVCES)

Common Domain Names

Domain	Role	Record Set
srvces.infra.digital.corp.in.cld	Used for the **green** cluster and essentially is what's used to switch from blue to green.	"A" record type pointing to the three active (green) core members
srvces.infra.digital.corp.in.cld	Used for the **blue** cluster to perform testing/soaking before promoting to the active green cluster.	"A" record type pointing to the three passive (blue) core members

Instance Details

Cluster 1

Instance	IP	Host Address
		srvces.infra.digital.corp.in.cld

Cluster 2

Instance	IP	Host Address
		srvces.infra.digital.corp.in.cld

Production

Common Domain Names

Domain	Role	Record Set
prod.infra.digital.corp.in.cld	Used for the **green** cluster and essentially is what's used to switch from blue to green.	"A" record type pointing to the three active (**green**) core members
prod.infra.digital.corp.in.cld	Used for the **blue** cluster to perform testing/soaking before promoting to the active green cluster.	"A" record type pointing to the three passive (**blue**) core members

Instance Details

Cluster 1

Instance	IP	Host Address
		prod.infra.digital.corp.in.cld

Cluster 2

Instance	IP	Host Address
		prod.infra.digital.corp.in.cld

How to Organize Releases

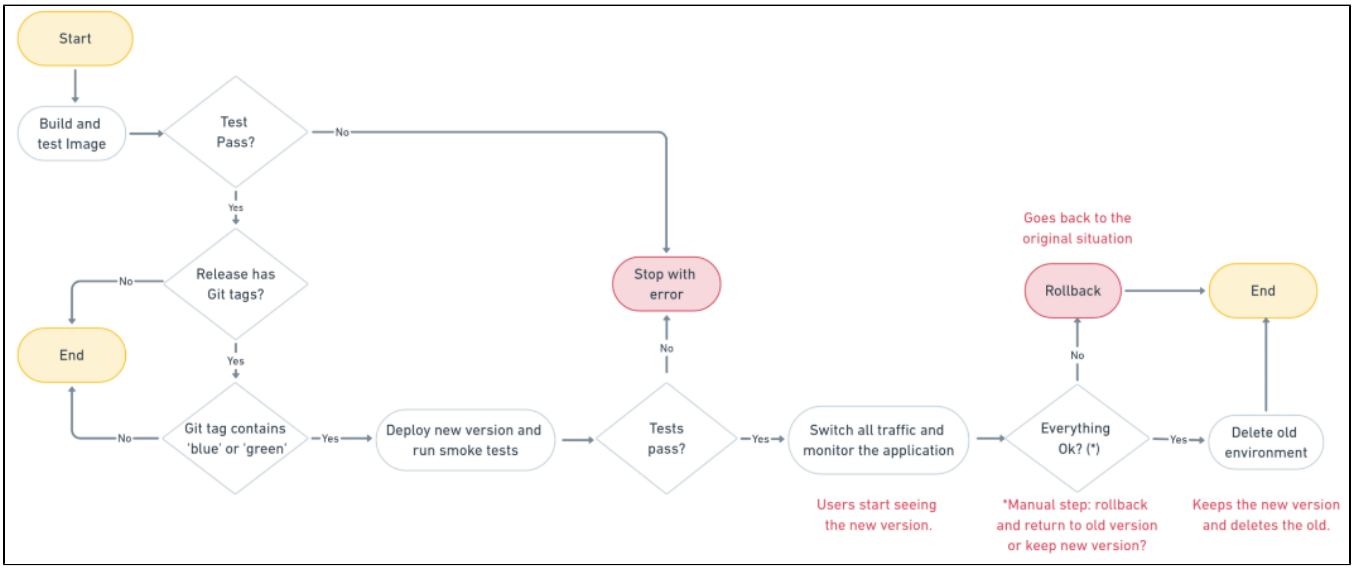
Step 1: Decide Which Pipeline Should Start

Step 2: Deploy

Step 3: Test the Deployment

Step 4: Go Live

Step 5: Cleanup or Rollback



Blue/Green Deployment: Use Cases and Benefits

1. **Seamless customer experience**: users don't experience any downtime.
2. **Instant rollbacks**: we can undo the change without adverse effects.
3. **No upgrade-time schedules for developers**: no need to wait for maintenance windows
4. **Testing parity**: the newer versions can be accurately tested in real-world scenarios

Blue/Green Deployment: Challenges

1. **User routing:** Failed or stuck user transactions/sessions when reverted instantly
2. **Costs:** High infrastructure costs
3. **Code compatibility:** running two versions of the applications in parallel.