digital.ai™

Best Practices for DevOps:

ADVANCED DEPLOYMENT PATTERNS

EXECUTIVE SUMMARY

Getting new software to your users without interrupting their experience is hard; how do you upgrade an application that's in active use? And if you do take an application offline for an update, what happens if something goes wrong?

It's tempting to try to minimize downtime and reduce release risks by implementing extensive manual processes and procedures—but pre-release work that isn't automated slows down delivery and makes it hard to roll out new features.

The good news is that you can avoid time-consuming manual work while reducing release risks by following DevOps principles to automate the release and deployment process. And as you automate, using advanced deployment patterns can help you speed up the software delivery cycle while maintaining control over the way your applications are deployed.

This white paper gives you insights into the DevOps best practice of advanced deployment patterns: Blue/Green deployments, rolling updates, canary releases, dark launches, and feature toggles. It describes how each pattern works, the advantages and disadvantages of each one, considerations for implementing them, and best practices when applying them.

Taking Automation to the Next Level with Advanced Deployment Patterns

Application downtime is expensive, so most enterprises strive to minimize or, better yet, eliminate it. Reducing downtime when you need to roll out software patches or new features is a technical challenge. But as an organization starts to automate software delivery pipelines and deliver applications to Production more often, avoiding downtime with application release evolves from a "nice-to-have" into something that's critical for the business.

But simply automating manual activities isn't always enough; organizations need to ensure that DevOps teams implement automated processes in a consistent way, so that those processes can be scaled across the enterprise. Advanced deployment patterns provide a flexible structure that you can use when automating the release of new software.

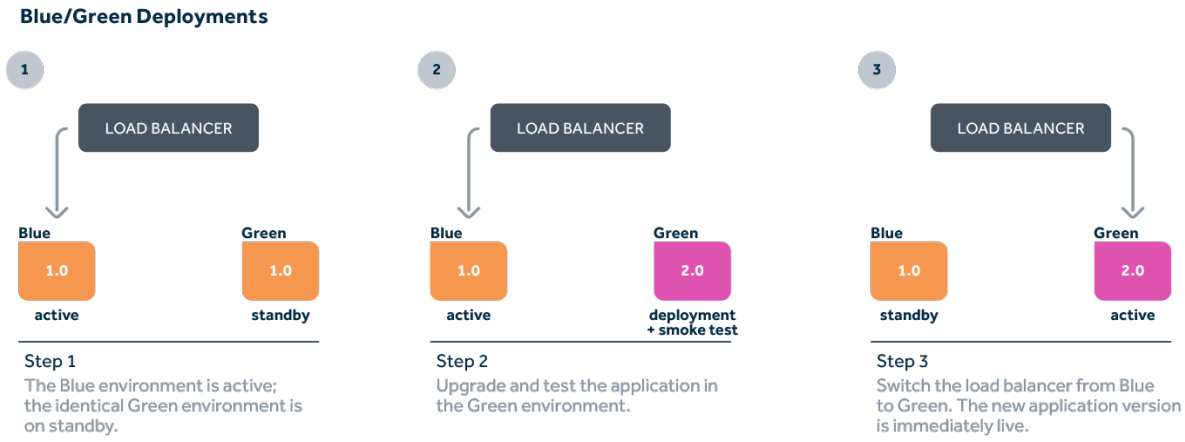
With deployment patterns, you can control the technical deployment of new software versions as well as the rollout of new features to users. You can even release features to limited user groups and test them in Production with much less risk than if you made them generally available.

Deployment patterns enable you to:

- Minimize application downtime and work toward the goal of zero downtime
- Predict and contain release and deployment risks
- Manage and resolve incidents with minimal impact on end users
- Achieve predictable, repeatable software deployments
- Address failed deployments in a reliable, effective way

Blue/Green Deployments

In the Blue/Green deployment pattern, a load balancer directs traffic to the active (Blue) environment while you upgrade the standby (Green) environment. After smoke testing the application in the Green environment and establishing that it is operating correctly, you adjust the load balancer to direct traffic from the Blue environment to the Green environment.



Advantages of Blue/Green Deployments

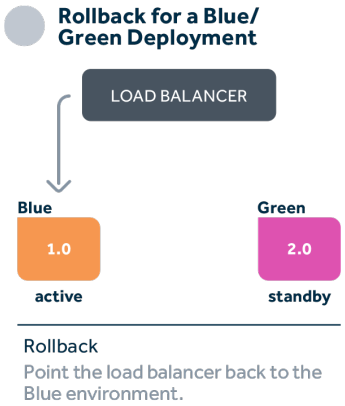
The Blue/Green deployment pattern provides a safe way to upgrade applications without interrupting their use. Blue/Green deployments work particularly well for monolithic applications that can take significant time to deploy, because you have full control over the point at which users can access the new version of the software.

Disadvantages of Blue/Green Deployments

Adopting the Blue/Green deployment pattern can increase operational overhead because you have to maintain duplicate Production environments with identical infrastructure. Additionally, updating database schemas while following a Blue/Green approach requires caution, as the new version of the application cannot use the database until it has been upgraded.

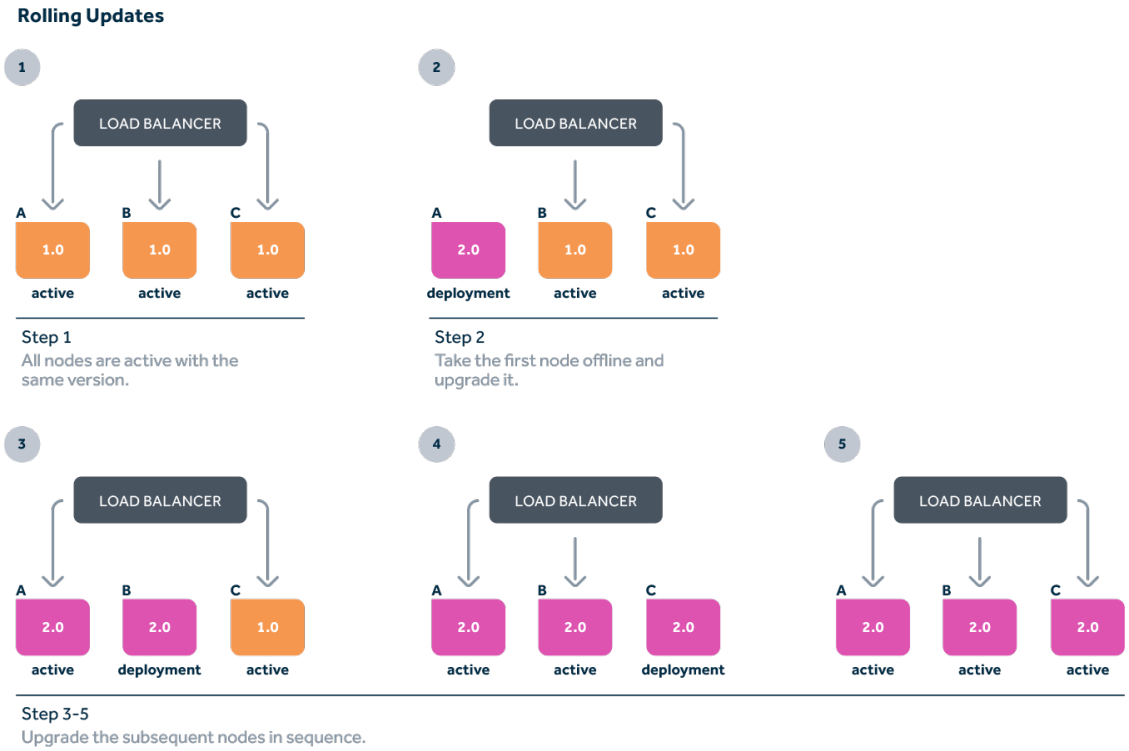
Rolling Back a Blue/Green Deployment

Blue/Green deployments offer fast and straightforward rollbacks. If the transition to the Green environment fails, or if something goes wrong once the Green environment is in active use, you simply adjust the load balancer to direct traffic back to the Blue environment.



Rolling Updates

In a rolling update, each server node in the environment is taken offline and upgraded, one by one. After a node is upgraded, manual or automatic smoke testing determines whether the application is functioning as expected. If the application is up and running properly, its server node is made available for user traffic, and the next node is taken offline to be upgraded.



Advantages of Rolling Updates

A rolling update is an incremental way of achieving zero-downtime deployments. Rolling updates work well for deployments that you expect to execute quickly, so you can minimize the impact on users.

digital.ai™

Disadvantages of Rolling Updates

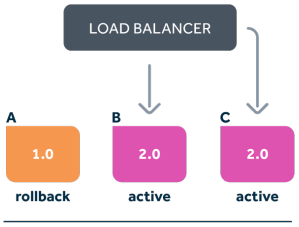
While a rolling update is executing, the environment will be somewhat unpredictable because different versions of the application are running in it at the same time. Also, the application architecture must support running in cluster mode because multiple instances of the application will access the database during the update.

Finally, taking servers offline means that there is less capacity for user traffic during the update.

Rolling Back a Rolling Update

To roll back a rolling update, you apply the same strategy of taking servers offline one-by-one to be downgraded. This approach ensures that a version of the application is always available for users.

Rollback for Rolling Updates



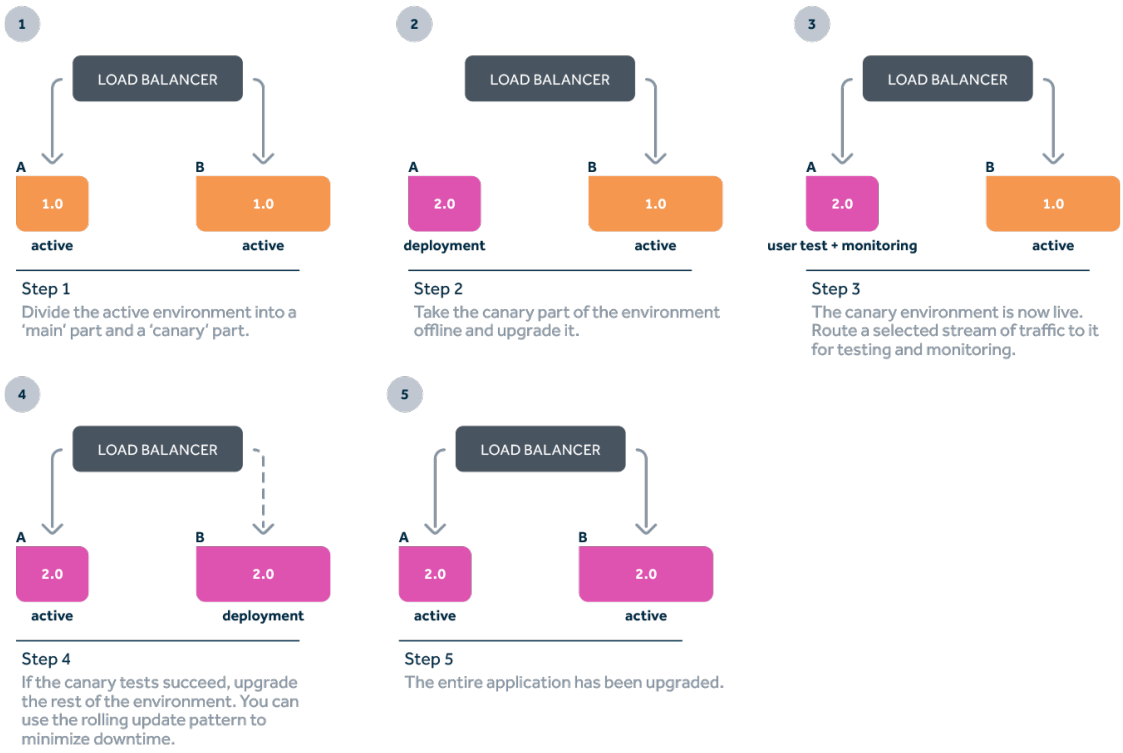
Rollback
Apply the same pattern, downgrading nodes one by one.

Canary Releases

In a canary release, you upgrade an application on a subset of the infrastructure and allow a limited set of users to access the new version. This approach allows you to test the new software under a Production-like load, evaluate how well it meets users' needs, and assess whether new features are profitable.

A canary release is similar to a rolling update because it involves releasing a new feature in a staged manner. However, when you perform a rolling update, you only verify that the application is technically stable and functional before moving on to the next stage of the rollout. With a canary release, you evaluate users' reactions to a new feature or functionality before deciding whether to release it more widely.

Deploying a Canary Release



Step 1
Divide the active environment into a 'main' part and a 'canary' part.

Step 2
Take the canary part of the environment offline and upgrade it.

Step 3
The canary environment is now live. Route a selected stream of traffic to it for testing and monitoring.

Step 4
If the canary tests succeed, upgrade the rest of the environment. You can use the rolling update pattern to minimize downtime.

Step 5
The entire application has been upgraded.

Advantages of Canary Releases

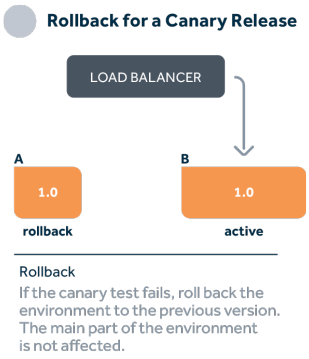
Canary releases are a good way to release experimental or beta features to users and gather their feedback. For example, you can use a canary release to roll out a new feature to users in a specific geographic region. If those users like the feature, you can deploy a canary to another region and test the user response there; or, you could choose to upgrade all remaining nodes in the environment, possibly by applying another release pattern to minimize application downtime.

Disadvantages of Canary Releases

Canary releases pose similar risks to rolling updates because multiple versions of the same application run in the environment while the canary release is deployed. For that reason, the environment will be somewhat unpredictable during the canary timeframe, and the application architecture must support running in cluster mode so that multiple instances can access the database.

Rolling Back a Canary Release

Rolling back a canary release is relatively easy because you only need to roll back the application version on servers where the new application version is deployed. User traffic can be redirected to nodes running the older version of the application, so users will not experience application downtime.



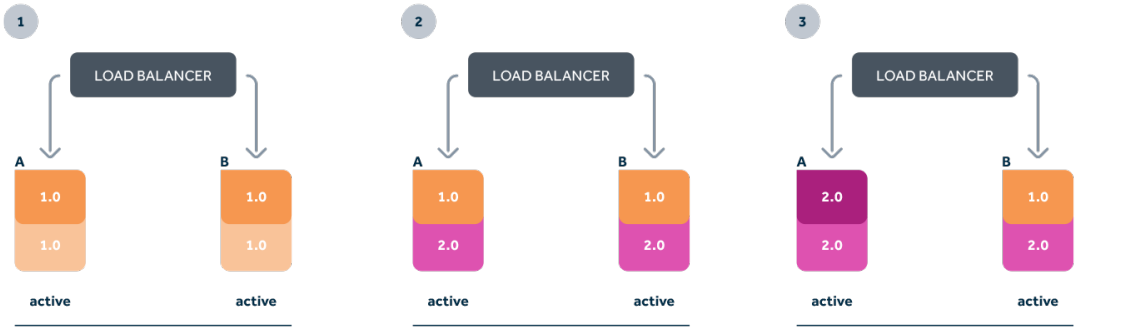
Dark Launches and Feature Toggles

Dark launching a feature means releasing an early version to Production so you can test it with realistic user traffic and usage patterns without actually exposing the feature to users. Dark launching is particularly effective for testing the back end of a feature because you can run back-end functionality in the background without the user being aware of it. When it's time to give users access to the feature you simply deploy its front-end interface.

For example, you could implement the back end of a product recommendation feature on an ecommerce website and test its performance without showing the recommendations to site visitors. When you're satisfied with the feature's performance, you can deploy the front-end recommendation interface on the website.

Feature toggles—also known as feature flags—allow you to further decouple the deployment of different software versions from the release of features to users. You can deploy new versions of an application as often as needed, with certain features disabled: releasing a feature to users is simply a matter of toggling it "on."

Deploying Dark Launches and Feature Toggles



Advantages of Dark Launches and Feature Toggles

Like canary releases, dark launching and toggling features allow you to roll out new features in a controlled way. However, dark launches and feature toggles do not require you to run multiple versions of an application in an environment simultaneously, which you must do for a canary release.

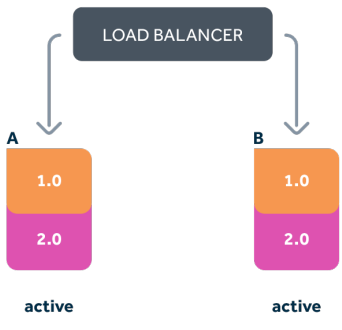
Disadvantages of a Dark Launch or Feature Toggle

Dark launches and feature toggles can be difficult to adopt because they require you to change the application that you want to deploy. Development teams must build the capability to dark launch or toggle a feature into the application at the code level, but doing so can be a challenge for mature applications with a large code base.

Rolling Back a Dark Launch or Feature Toggle

Rolling back a feature that has been dark launched or controlled by a feature toggle is relatively simple. In the case of a dark launch, you simply remove the interface that allows users to access the feature. And in the case of a feature toggle, you disable access to the feature.

Rollback for a Dark Launch or Feature Toggle



Considerations for Adopting Advanced Deployment Patterns

While advanced deployment patterns can help you take release and deployment automation to the next level, keep the following considerations in mind before you start applying them.

- **You might need to change your applications.** Minimizing or eliminating downtime often requires multiple versions of the application to be active in Production at the same time, which might require you to change your applications at the code level.

- **Running multiple application versions at the same time means that the database must be compatible with all active versions.** You must decouple database updates from application updates. In addition, Development teams must design database changes to be backward compatible. In practice, maintaining compatibility has consequences for both database changes and application logic.

- **You might need to change your infrastructure.** Some deployment patterns require you to create and maintain multiple environments that end users can access. Many DevOps teams achieve a multi-environment setup by using virtual infrastructure in the form of cloud-based environments. They might also containerize applications so that they can more easily stop, start, scale up, scale down, and load balance them using container management and orchestration tools.

  However, shifting from on-premises infrastructure and traditional middleware platforms to containers and the cloud cannot be done overnight. Most applications require architecture changes to run effectively in such environments.

Best Practices When Using Advanced Deployment Patterns

While advanced deployment patterns can help improve your DevOps performance, each pattern involves a certain level of risk. However, there are several best practices that you can follow to reduce the risk of adopting advanced deployment patterns.

- **Limit the number of application versions that are live at the same time to two.** This approach reduces the complexity of deployments, infrastructure management, and database compatibility.

- **Deploy in small increments, only releasing a few features at a time.** This approach, which is one of the core principles of DevOps, reduces deployment risk because the more frequently you release software, the more reliable your release process becomes, as teams have a greater incentive to fix problems and improve the process.

  Deploying small increments also reduces lead time, which is the time that elapses between the start of feature development until that feature is available to end users. Reduced lead time is one of the key goals of Continuous Delivery, as it enables you to deliver value to users and to the market faster.

- **Manage risk by refactoring monolithic applications into microservices that you can deploy independently.** Microservices can give you more control over the impact of changes to the environment. They can also increase development throughput and enable faster development cycles, as new features are typically delivered as one or more microservices and don't require changing the entire application.

Get the Most Out of Advanced Deployment Patterns with Application Release Orchestration

In an enterprise environment, standardizing and automating releases and deployments is key for optimizing software delivery. Application Release Orchestration (ARO), which encompasses both release orchestration and deployment automation, enables you to standardize and automate release and deployment by orchestrating the build, test, provisioning, configuration management, and deployment tools in your software delivery pipeline.

ARO functionality is also sometimes called Application Release Automation (ARA), Continuous Delivery Management, or Continuous Delivery and Release Automation (CDRA).

You can use Application Release Orchestration to implement advanced deployment patterns in a consistent way across applications (from monoliths to microservices) and infrastructure (from on-premises to the cloud). ARO can also help you combine deployment patterns in sophisticated ways, so you can maximize the benefits of the patterns you're using.

The fastest way to get started with advanced deployment patterns is to adopt an Application Release Orchestration solution that supports them out of the box, so you don't have to write scripts or build workflows to create the pattern you want. The right ARO solution ensures patterns are executed consistently while still supporting flexibility. An ARO solution that comes with pre-defined deployment pattern templates allows DevOps teams to try out different patterns and see what works best for their applications and infrastructure.

The Digital.ai Value Stream Platform is an ARO solution that supports automation and orchestration for your entire software delivery pipeline. With Digital.ai, you can create scalable release and deployment processes that DevOps teams can reuse across applications and environments—all without writing scripts or building workflows by hand.

The Digital.ai Value Stream Platform includes fully integrated, out-of-the-box support for Blue/Green deployments, rolling updates, canary releases, dark launches, and feature toggles. It lets you take control of release and deployment processes and at the same time, gives your teams additional flexibility to best meet their applications' unique needs.

The Digital.ai Value Stream Platform makes release and deployment practices across the enterprise reusable, scalable, and secure through:

- **Proven templates that work for Production deployments at enterprise scale**
- **Integrated support for both automated and manual release activities**
- **Visibility into the status of running releases and deployments**
- **Proactive notification of delays and bottlenecks in the process**
- **Detailed error analysis for troubleshooting failed releases and deployments**
- **Fully auditable releases and deployments, with detailed reports and analytics**
- **Granular role-based access control over all applications and environments**

About Digital.ai

Digital.ai enables enterprises to focus on outcomes instead of outputs, create greater business value faster, and deliver secure digital experiences their customers trust. The Digital.ai Value Stream Platform seamlessly integrates all the disparate tools and processes across the various value streams, uses data and AI/ML to create connective tissue between them, and provides the real-time, contextual insights required to drive and sustain successful digital transformation. With Digital.ai, enterprises have the visibility they've been seeking to deliver value, drive growth, increase profitability, reduce security risk, and improve customer experience.

Learn more at digital.ai