Patterns

Mojtaba Shahin

Week 10: Lectorial – Part 1

Content

- Part 1
  - DevOps and Deployment Patterns
- Part 2
  - Decomposition Patterns
  - Legacy Modernization Patterns
  - C4 Model Dynamic and Deployment Diagrams

Acknowledgements

- Most of the **texts** and **images** in the slides come from the following sources:
  - https://c4model.com/
  - https://github.com/structurizr/examples
  - https://structurizr.com/share/36141
  - Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)
  - Pautasso, C., Software Architecture- Visual Lecture Notes, LeanPub, 2023 (https://leanpub.com/software-architecture/)
  - Introduction to DevOps by Len Bass – URL: goo.gl/iMwdfg
  - Len Bass, Ingo Weber, Liming Zhu, DevOps: A Software Architect's Perspective, 2015
  - Martin Fowler (2024), "Branch By Abstraction" https://martinfowler.com/bliki/BranchByAbstraction.html
  - Ian Cartwright, Rob Horn, and James Lewis (2024), Patterns of Legacy Displacement (https://martinfowler.com/articles/patterns-legacy-displacement/)
  - The University of Queensland's Software Architecture Course Materials, @ Richard Thomas, CC BY-SA 4.0 (https://github.com/CSSE6400)
  - Neal Ford, Mark Richards, Pramod Sadalage, Zhamak Dehghani, Software Architecture: the Hard Parts, O'Reilly Media, 2021

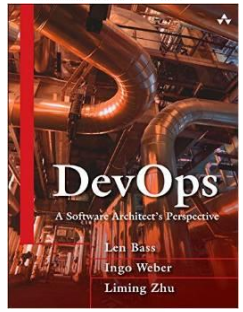
Acknowledgements

- Most of the **texts** and **images** in the slides come from the following sources:
  - Mojtaba Shahin, Mansooreh Zahedi, Muhammad Ali Babar and Liming Zhu, *An Empirical Study of Architecting for Continuous Delivery and Deployment*, In: Empirical Software Engineering, 24(3), 2019, Springer
  - Mojtaba Shahin, Muhammad Ali Babar and Liming Zhu, *Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices*, In: IEEE Access, 5 (99), 2017, IEEE.
  - Matthew Skelton, Chris O'Dell , Continuous Delivery with Windows and .NET, 2016, O'Reilly Media, Inc.

What are Architectural Patterns?

DevOps and Deployment Patterns

Traditional Development (Over the wall development)*



Board or marketing has idea

Developers implement the idea

Operators place the software in production

Time

* Source: Introduction to DevOps by Len Bass – URL: goo.gl/iMwdfg

Where Does the Time Go?*

- As Software Engineers, our view is that there are the following activities in software development
  - **Requirements**
  - **Design**
  - **Implementation**
  - **Test**

- **Code Complete**
- Different methodologies will organize these activities in different ways.
- Agile focuses on getting to Code Complete faster than other methods.

* Source: Introduction to DevOps by Len Bass – URL: goo.gl/iMwdfg

Two issues in the traditional development*

(1) Code Complete in the Development Side ≠ Code is ready for Production

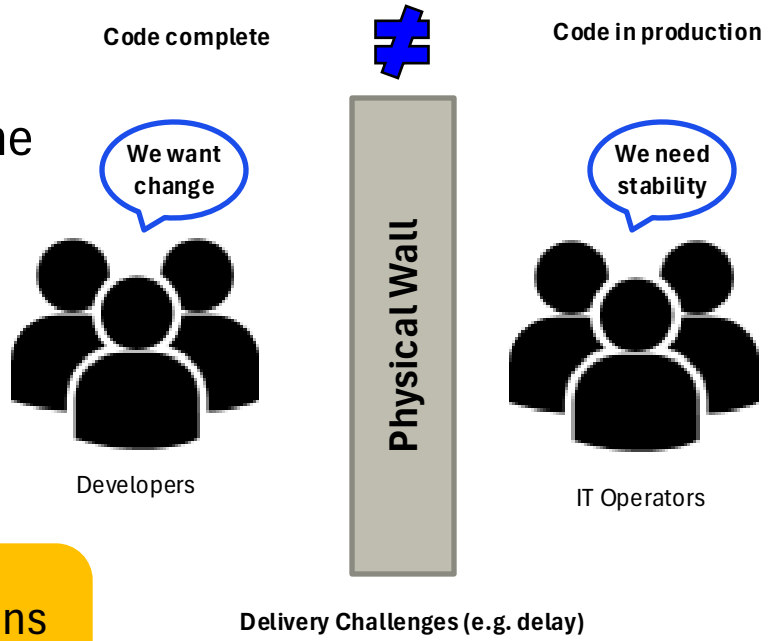
- In other words, when developers commit changes to version control systems and test them in their environment, it does not necessarily mean that those changes can be easily deployed in production environment.
- Between the completion of the code and the placing of the code into production there is a step called: **Deployment**
- Deploying the code can be **very time consuming** because of errors that could occur in the production environment.

* Source: Introduction to DevOps by Len Bass – URL: goo.gl/iMwdfg

Two issues in the traditional development*

(2) Developers and Operators might have **opposite goals**

- Developers (Dev) try to push new features into the production continuously. At the same time, the main concern of the Operations team (Ops) is system **dependability** and **availability**.

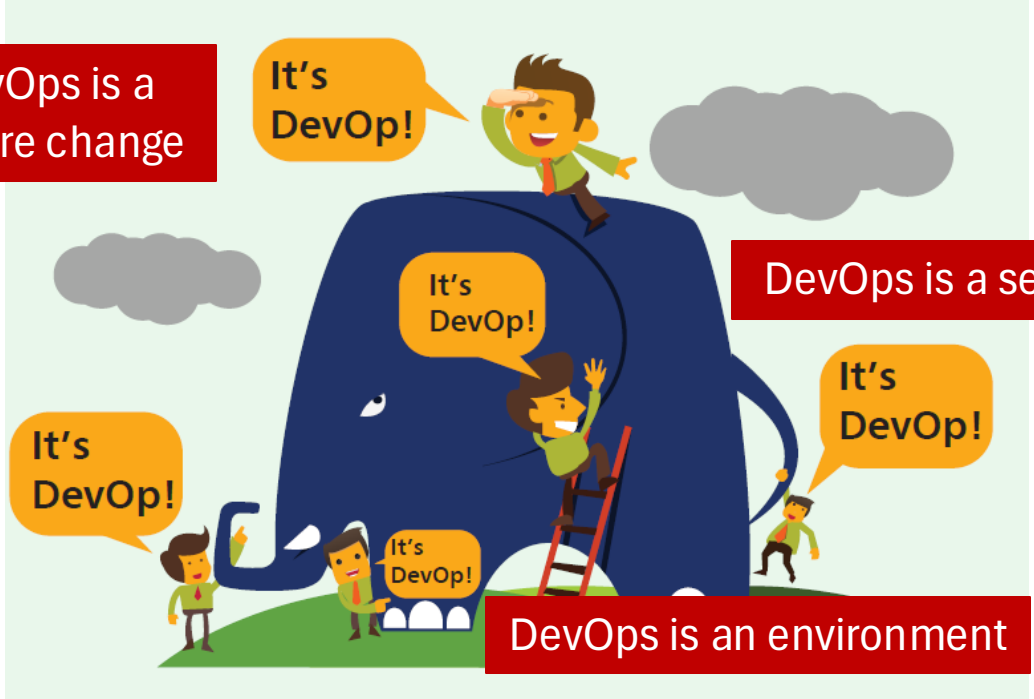


These two issues do not allow software organizations to release software quickly and in a reliable way.

*Source: http://dev2ops.org

What is DevOps?

DevOps is a culture change



DevOps is a set of practices

DevOps is a method

DevOps is an environment

Source: www.happiestminds.com/whitepapers/devops.pdf

DevOps Definitions

- **Development (Dev) and Operations (Ops)**

- "A software development method that promotes communication, collaboration, integration, automation, and measurement of cooperation between software developers and operations team"

- "DevOps is a set of practices aimed at reducing the time between committing a change to a system and placing changed code into the production, while ensuring high quality"- Len Bass, Ingo Weber, and Liming Zhu

Which Software Architecture Styles enable DevOps?

Continuous Practices in DevOps

Continuous Integration

vs.

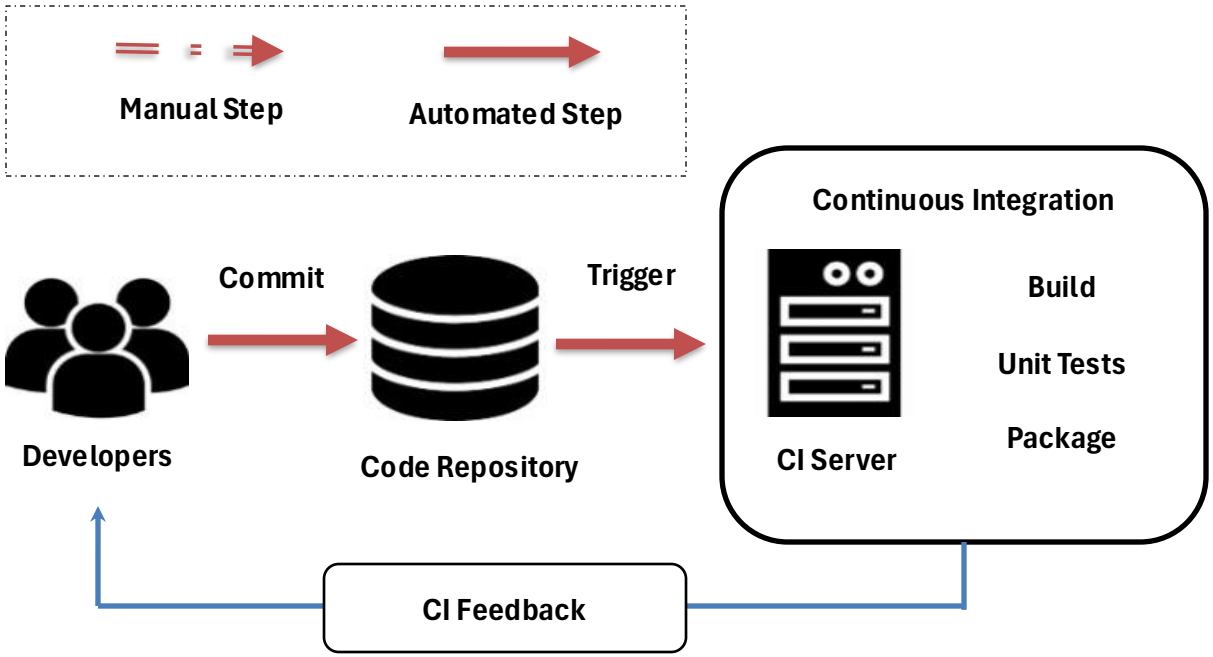
Continuous Delivery

vs.

Continuous Deployment

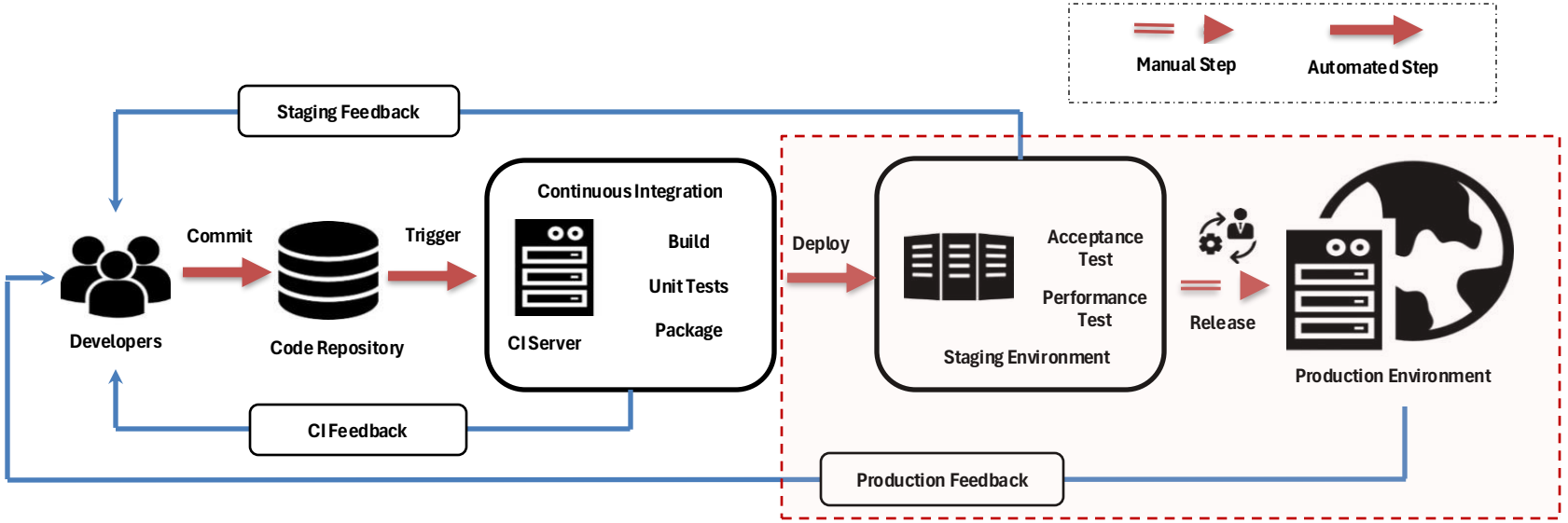
Continuous Integration (CI)

- Developers integrate and merge code **frequently** (e.g., multiple times per day) into CI Servers (e.g. Jenkins, Bamboo)



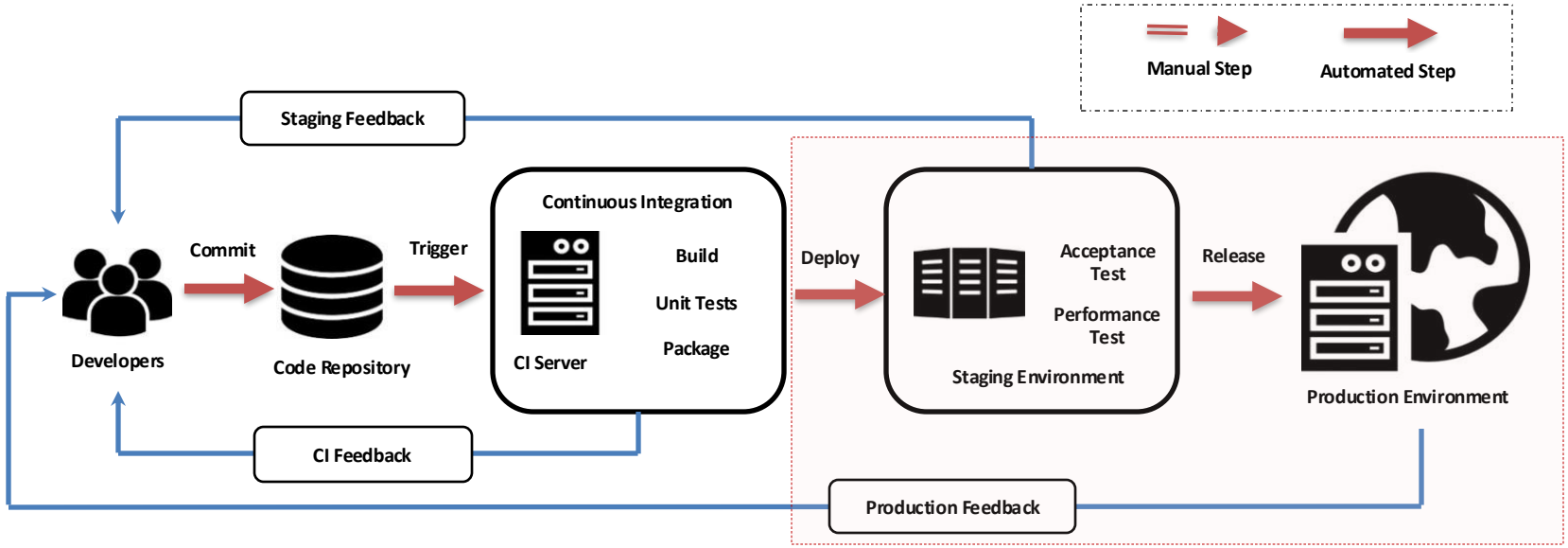
Continuous Delivery

- Extend CI
- Where an application is ***potentially*** capable of being deployed.
- Ensure an application is always at releasable state after successfully passing automated tests and quality checks
- Pull-based approach
- Applicable to all types of systems and organizations

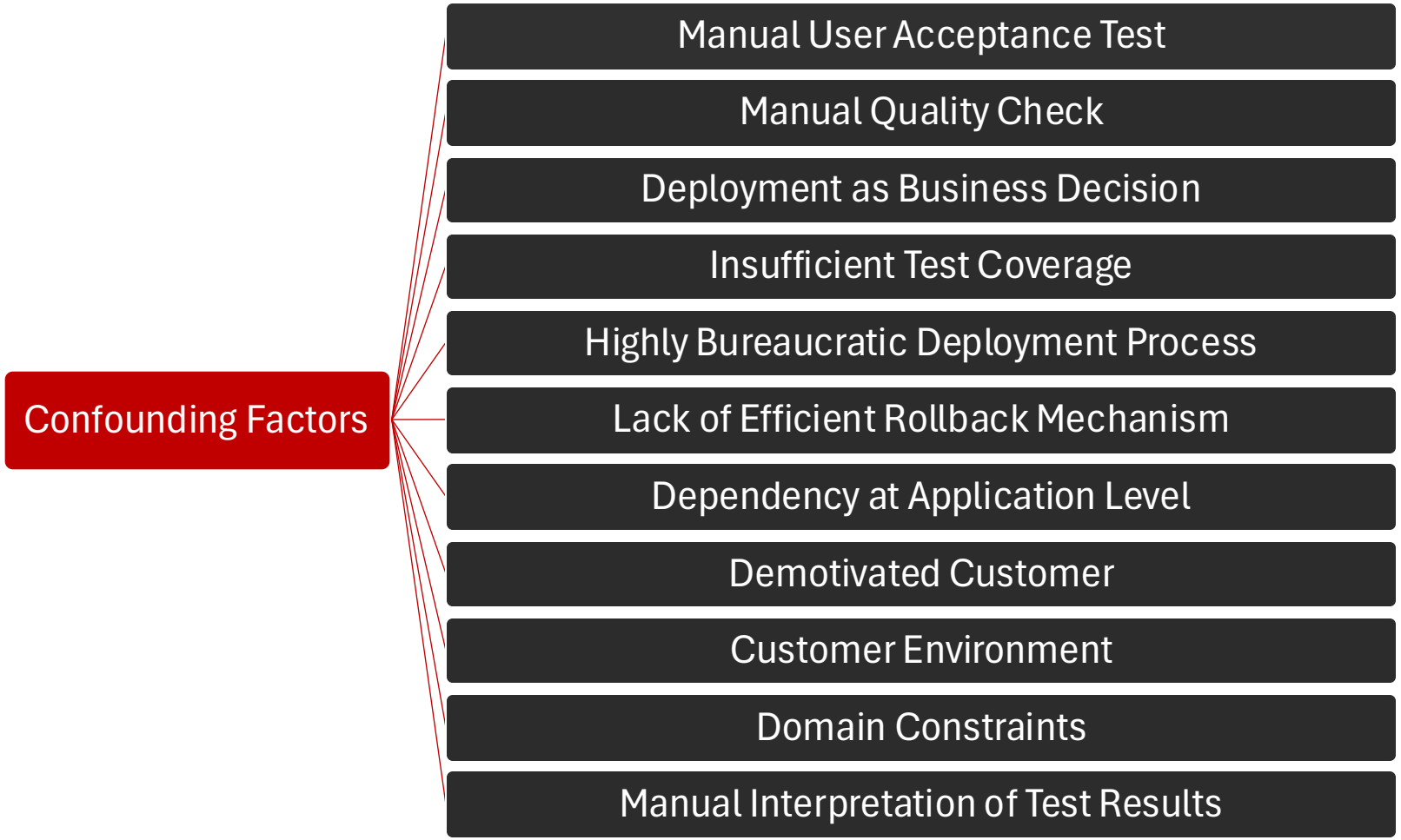


Continuous Deployment

- Where an application is ***automatically*** deployed to production on ***every update***.
- No manual steps or decision-making process for ***when*** and ***what*** to release
- Push-based approach
- Maybe suitable for certain types of organizations or systems

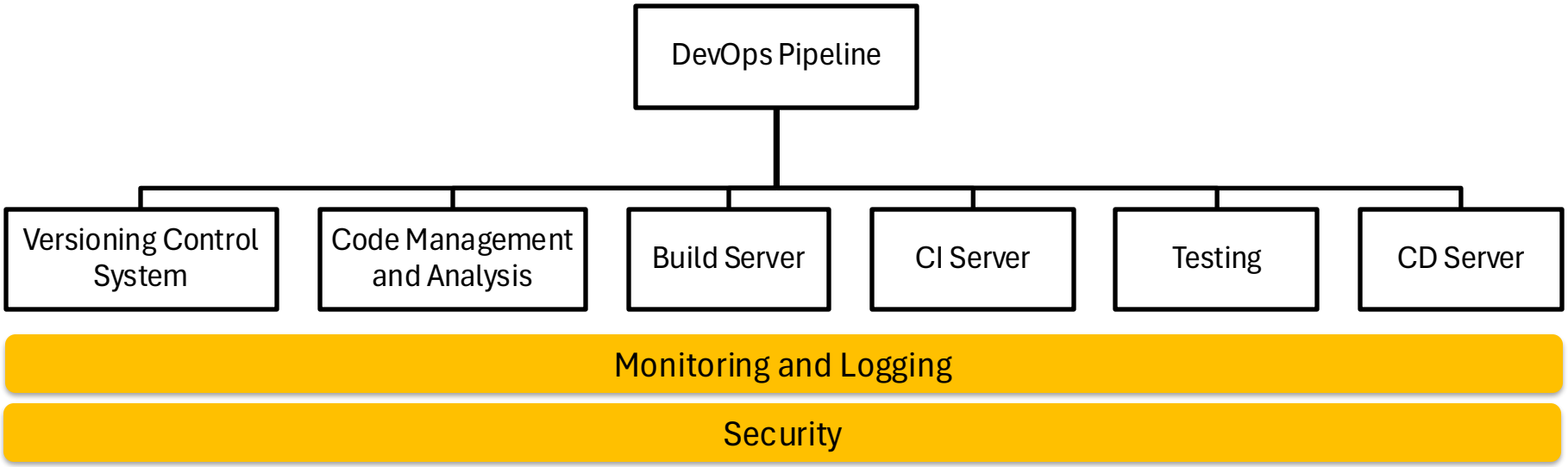


Confounders in moving from Continuous Delivery to Continuous Deployment



DevOps Pipeline/CI-CD Pipeline or DevOps toolchain

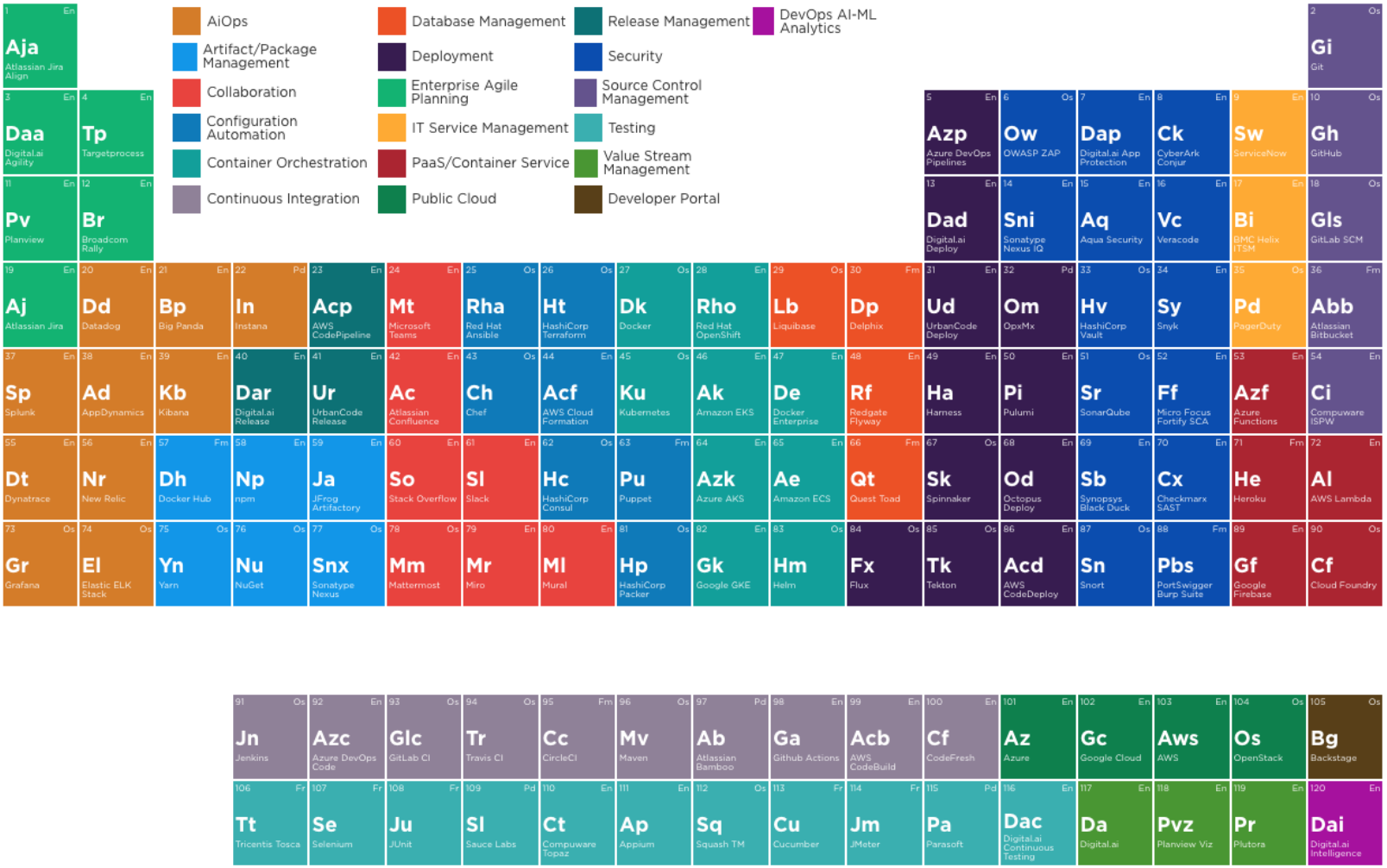
- To implement DevOps, organizations need to design and build a DevOps pipeline
- DevOps pipeline is a toolchain to transfer **code** from **code repository** to the **production environment**.
- In general, a DevOps pipeline should include several stages (e.g., build and packaging) to transfer code from code repository to the production environment
- **Automation** is a critical practice in DevOps pipeline; however, sometimes manual tasks (e.g., quality assurance tasks) are unavoidable in the pipeline.



Supporting Tools for DevOps Pipeline

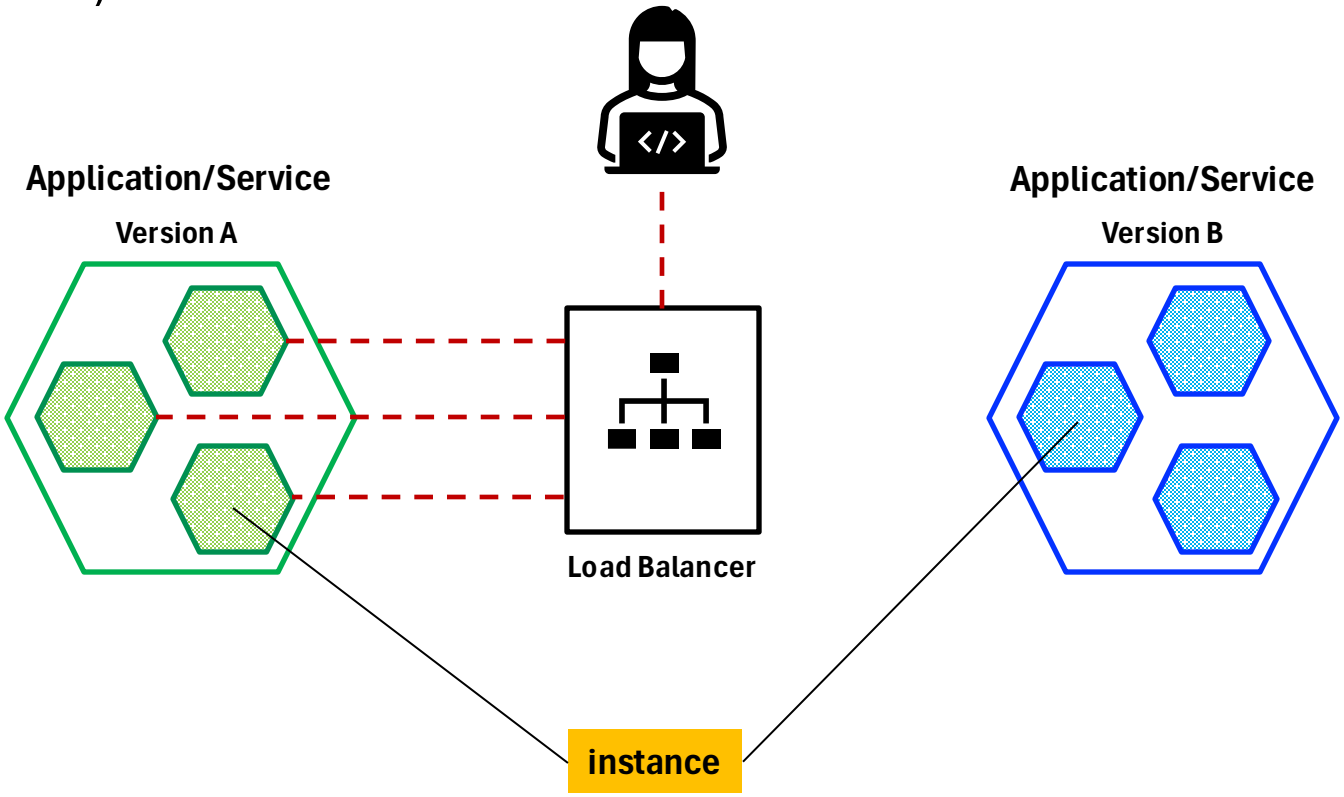
RMIT UNIVERSITY

https://digital.ai/periodic-table-of-devops-tools



Deployment Strategies*

N instances of an application/service (**Version A**) are currently deployed and running. The **goal** is to replace the N instances of the current version of the application/service (**Version A**) with a new version of the application/service (**Version B**).



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

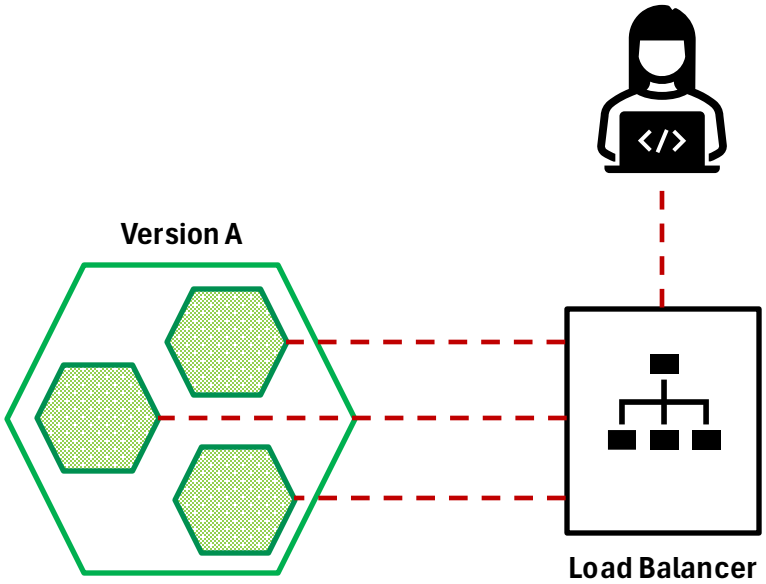
Deployment Strategies*

- **Recreate**: Version A is terminated then version B is rolled out.
- **Ramped** (also known as rolling-update or incremental): Version B is slowly rolled out and replacing version A.
- **Blue/Green**: Version B is released alongside version A, then the traffic is switched to version B.
- **Canary**: Version B is released to a subset of users, then proceed to a full rollout.
- **A/B testing**: Version B is released to a subset of users under specific condition.
- **Shadow**: Version B receives real-world traffic alongside version A and doesn't impact the response.

*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

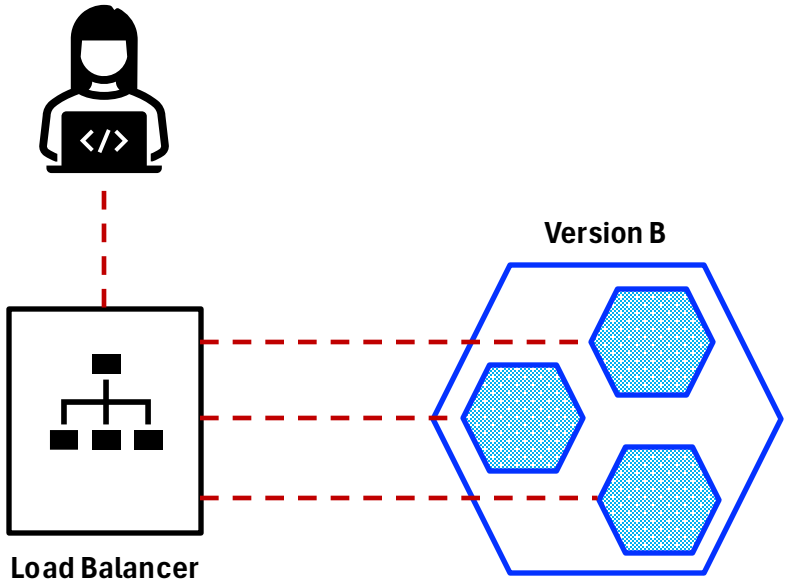
Recreate Deployment*

- Version A is **terminated** then version B is rolled out.
- The recreate strategy is a dummy deployment which consists of shutting down version A then deploying version B after version A is turned off.



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

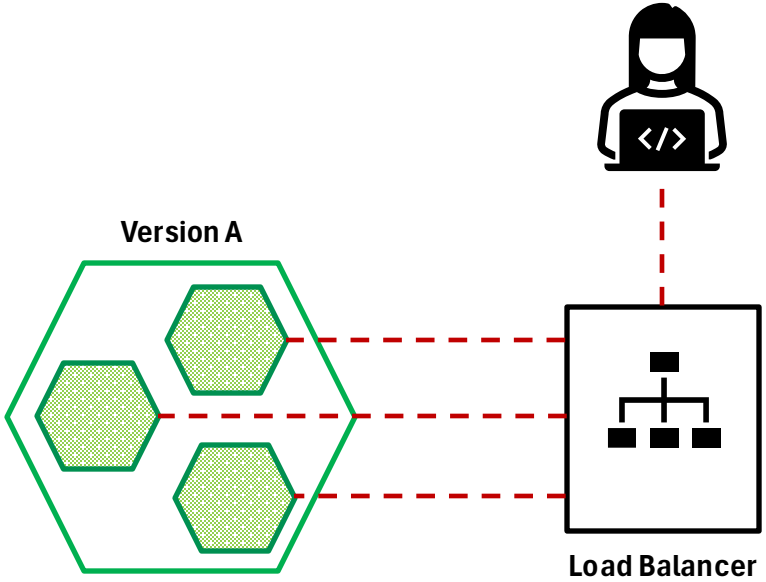
Recreate Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

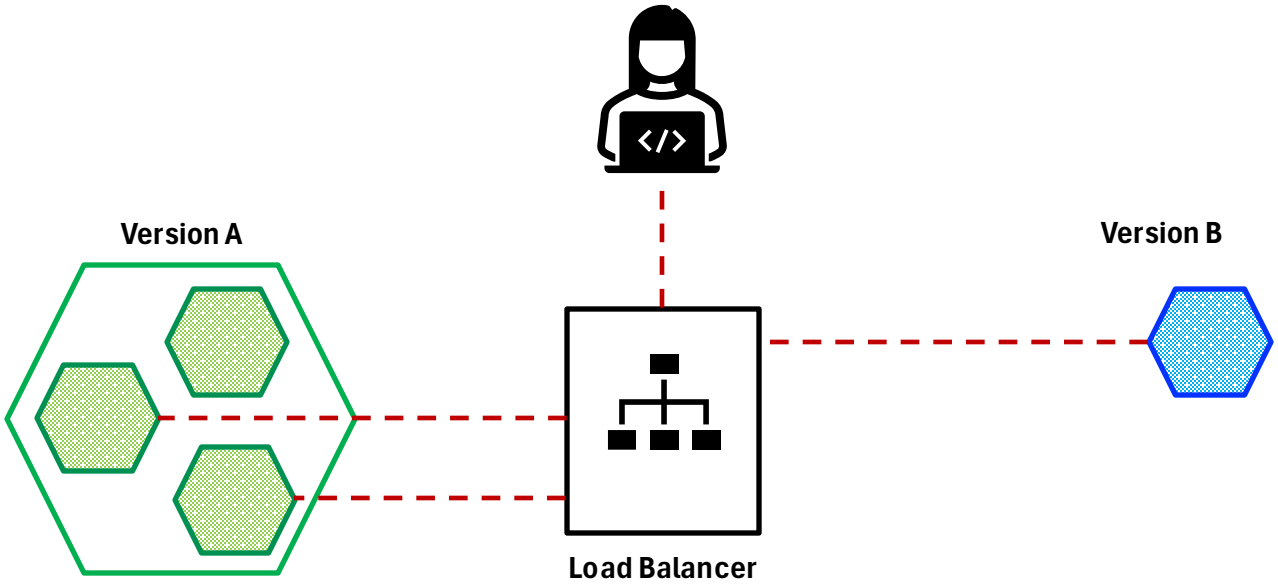
Ramped Deployment*

- The ramped deployment strategy consists of **slowly rolling out** a version of an application by replacing instances one after the other until all the instances are rolled out. It usually follows the following process: with a pool of version A behind a load balancer, one instance of version B is deployed. When the service is ready to accept traffic, the instance is added to the pool. Then, one instance of version A is removed from the pool and shut down.



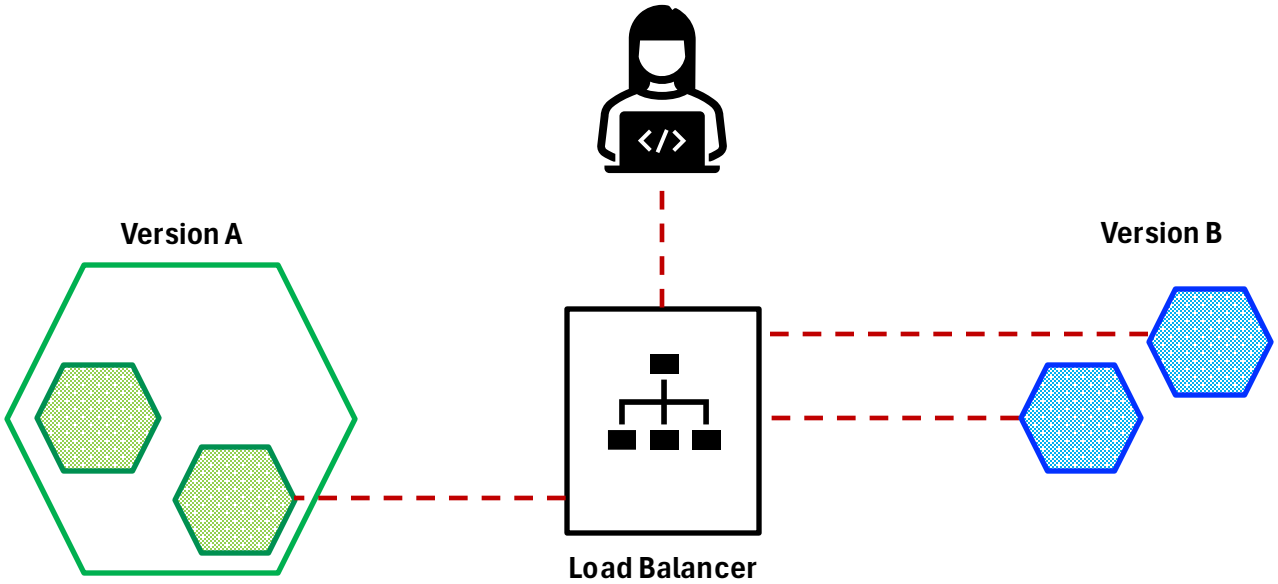
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Ramped Deployment*



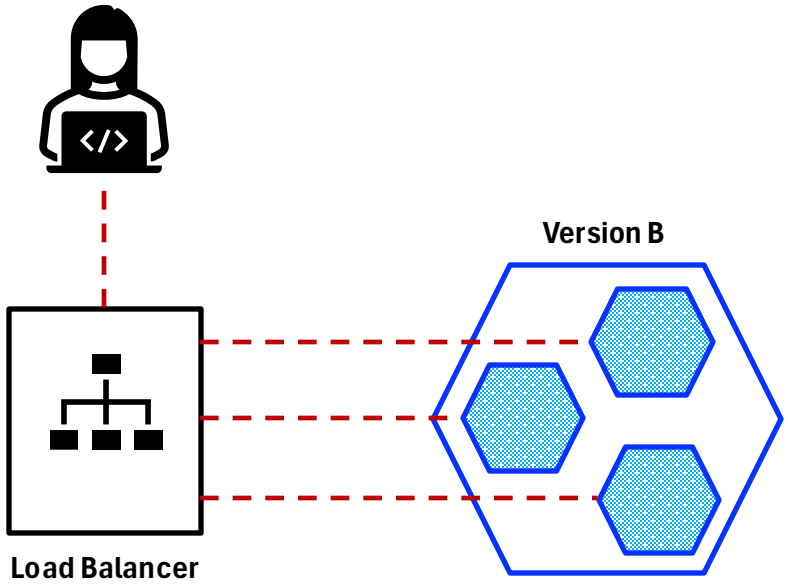
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Ramped Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

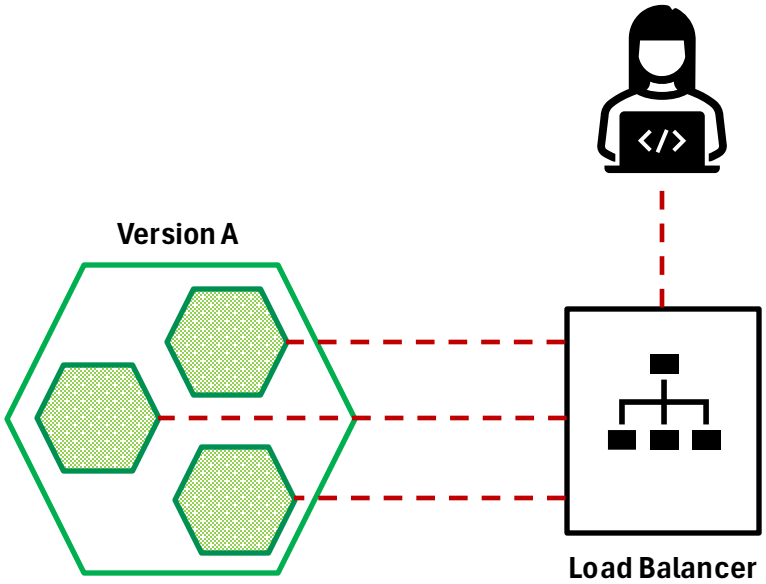
Ramped Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

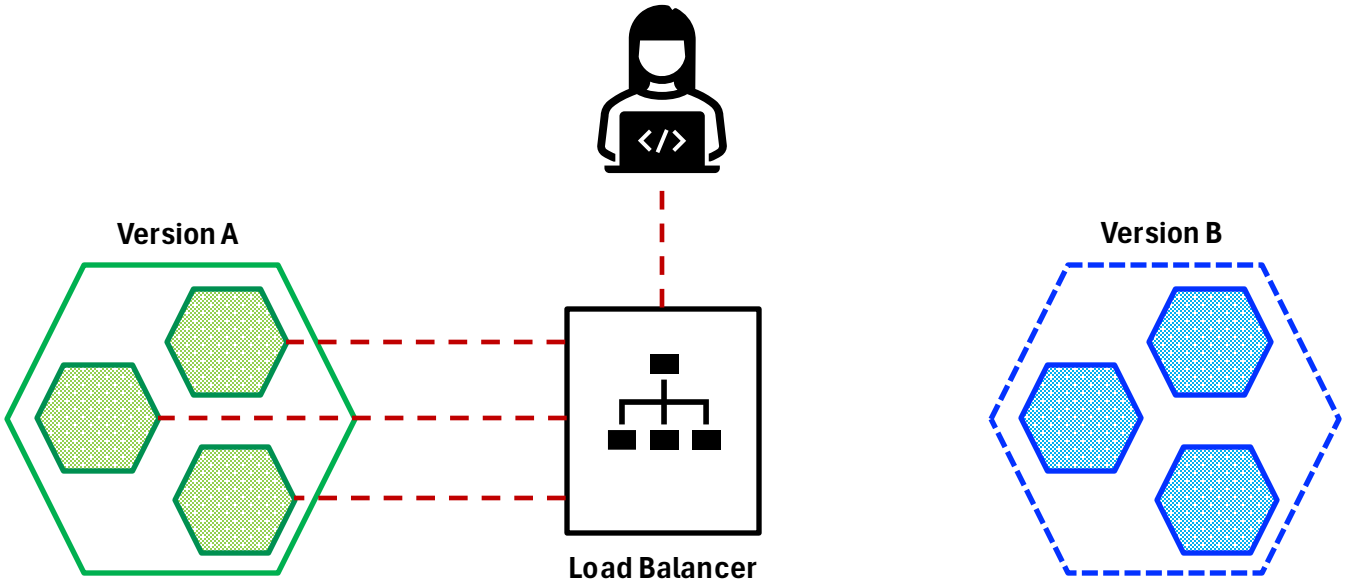
Blue/Green Deployment*

- The blue/green deployment strategy differs from a ramped deployment, version B (blue) is deployed **alongside** version A (green) with exactly the same amount of instances. After testing that the new version meets all the requirements, the traffic is switched from version A to version B at the load balancer level.



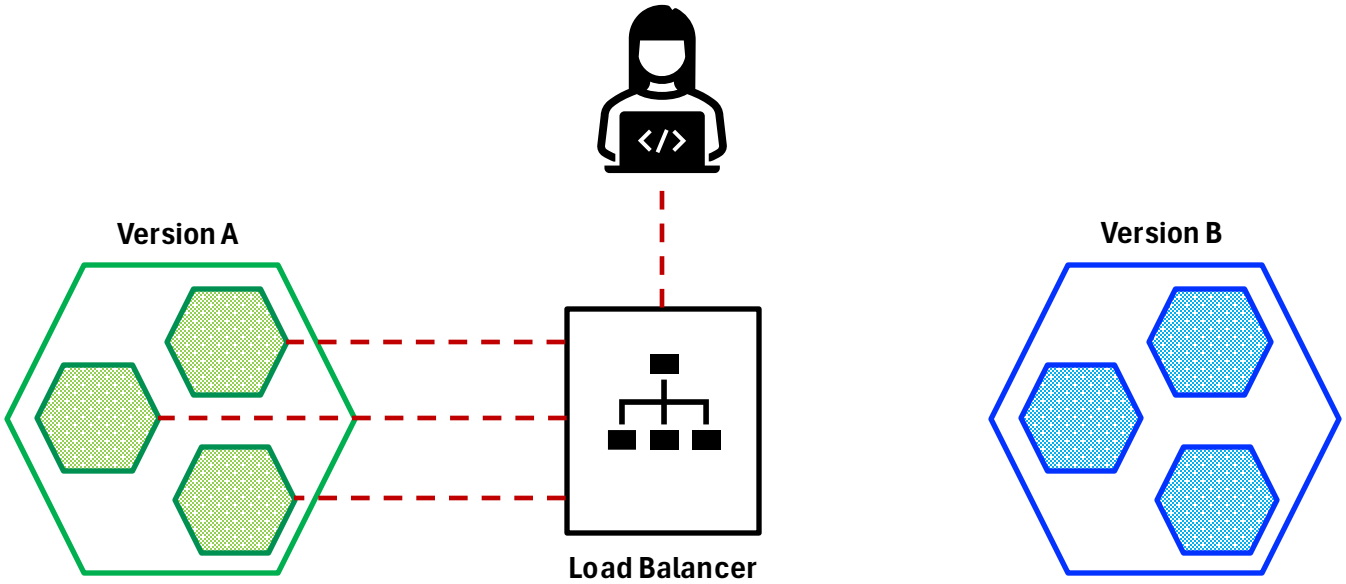
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Blue/Green Deployment*



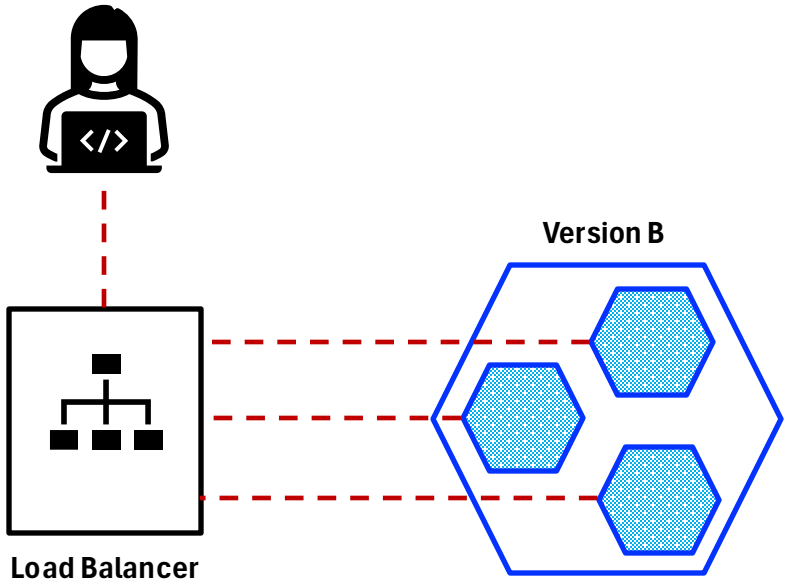
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Blue/Green Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

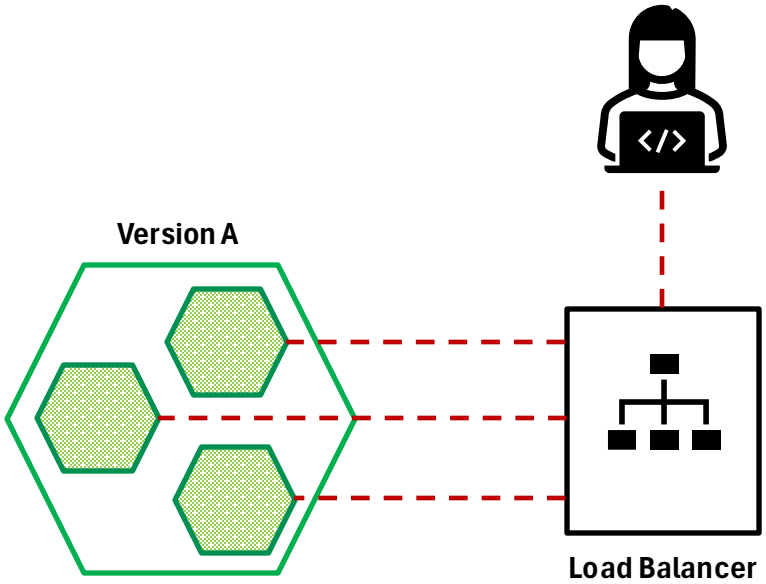
Blue/Green Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

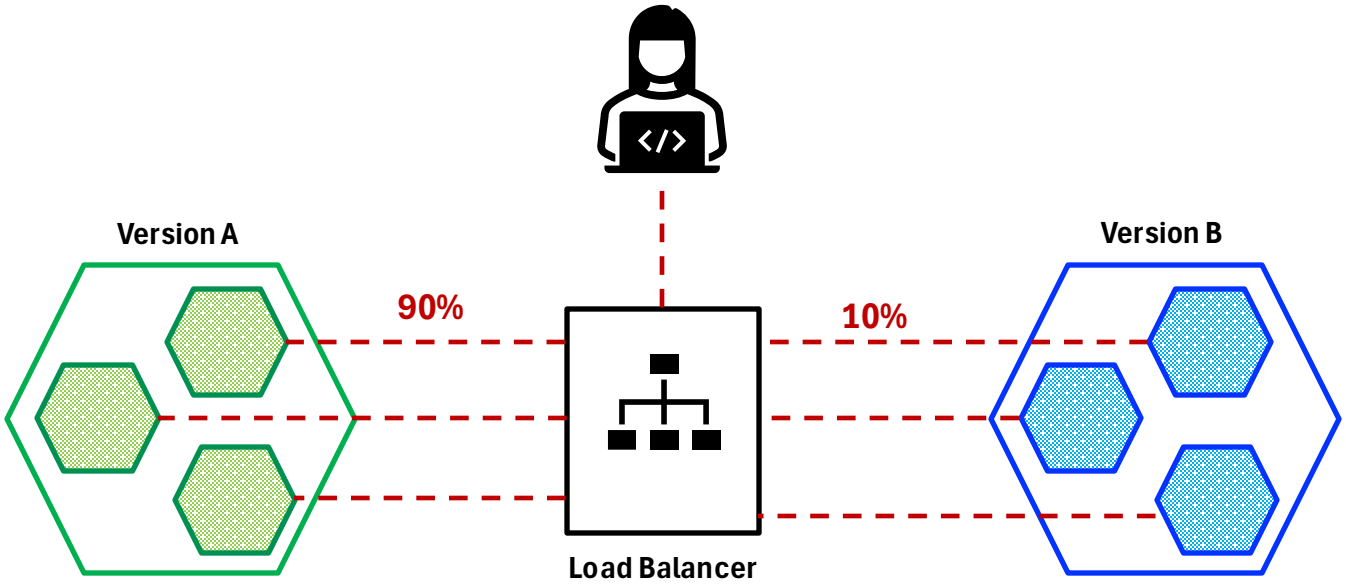
Canary Deployment*

- A canary deployment consists of **gradually shifting production traffic** from version A to version B. Usually the traffic is split based on weight. For example, 90% of the requests go to version A, 10% go to version B.
- This technique is mostly used when the tests are lacking or not reliable or if there is little confidence about the stability of the new release on the platform.



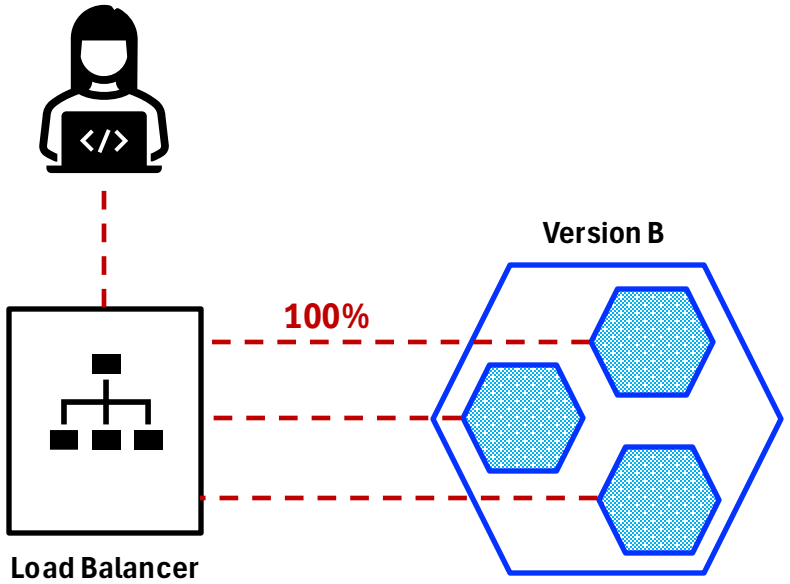
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Canary Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Canary Deployment*



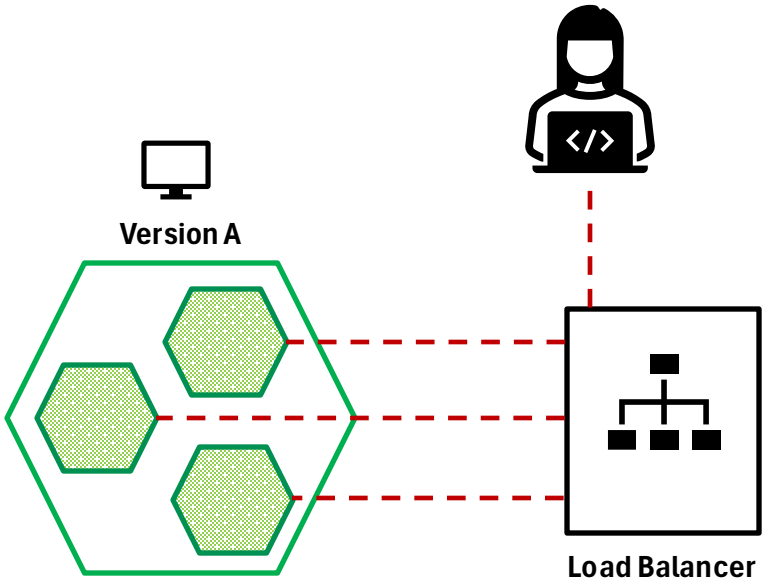
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

A/B testing Deployment

- A/B testing deployments consists of **routing a subset of users to a new functionality under specific conditions**. It is usually a technique for making business decisions based on statistics, rather than a deployment strategy. However, it is related and can be implemented by adding extra functionality to a canary deployment so we will briefly discuss it here.
- A/B testing is a way of testing features in your application for various reasons like **usability**, **popularity**, etc.**
- Here is a list of conditions that can be used to distribute traffic amongst the versions:
  - Geolocalisation
  - Technology support: browser version, screen size, operating system, etc.
  - Language

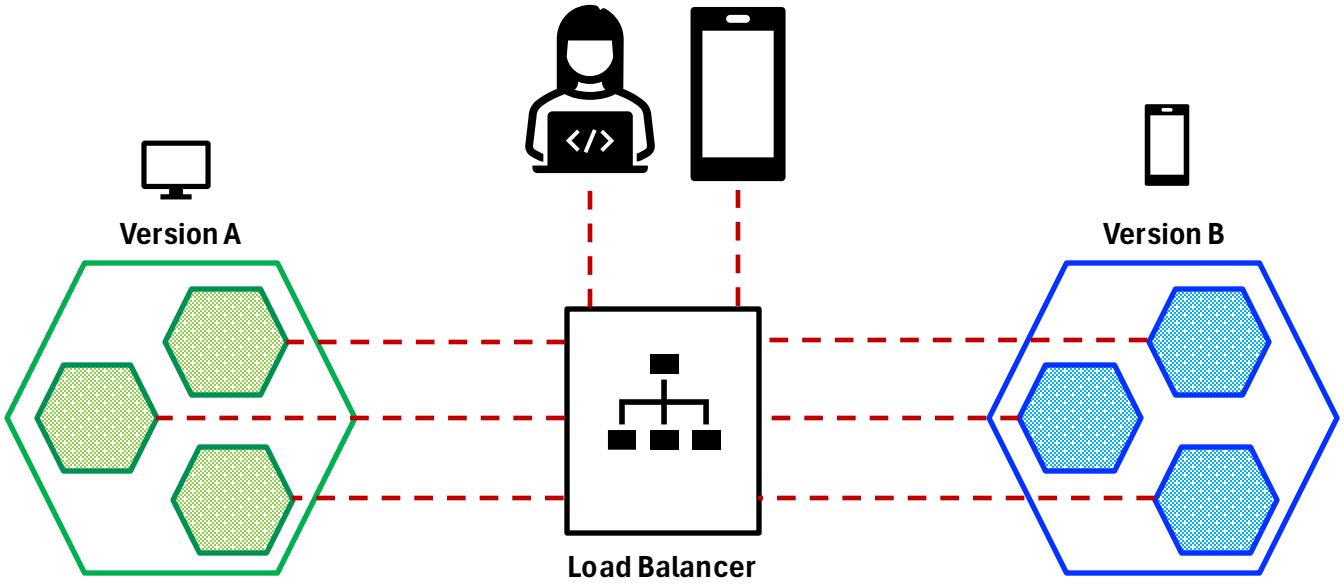
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)
**SourceL https://blog.christianposta.com/deploy/blue-green-deployments-a-b-testing-and-canary-releases/

A/B Testing Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

A/B Testing Deployment*



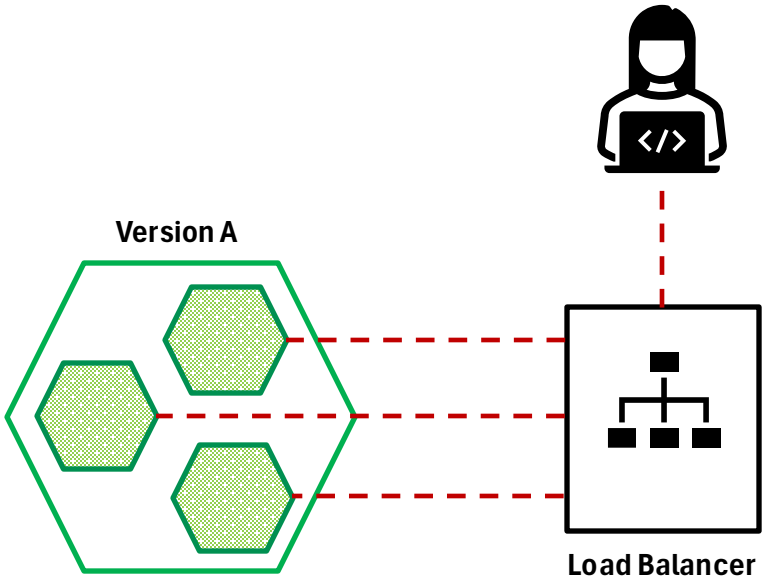
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Shadow Deployment*

- A shadow deployment consists of releasing version B alongside version A, forking version A's incoming requests and sending them to version B as well.
  - The responses from version B are not shown to users
  - A rollout (making Version B available to users) will only begin after Version B has proven to meet the required stability and performance.

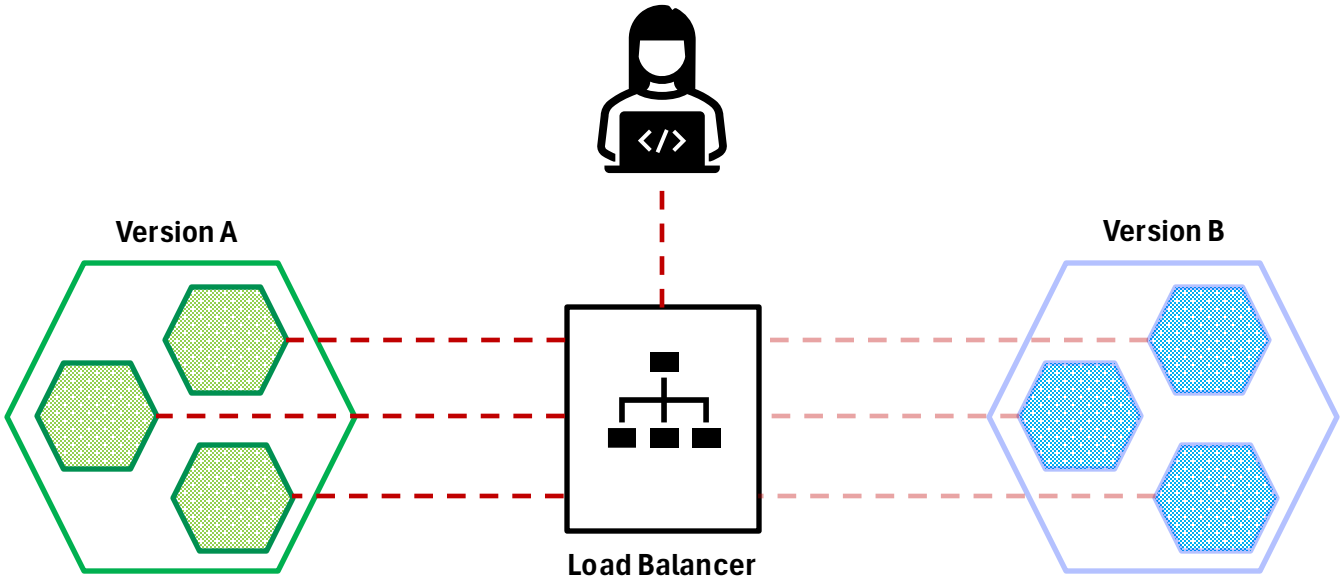
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Shadow Deployment*



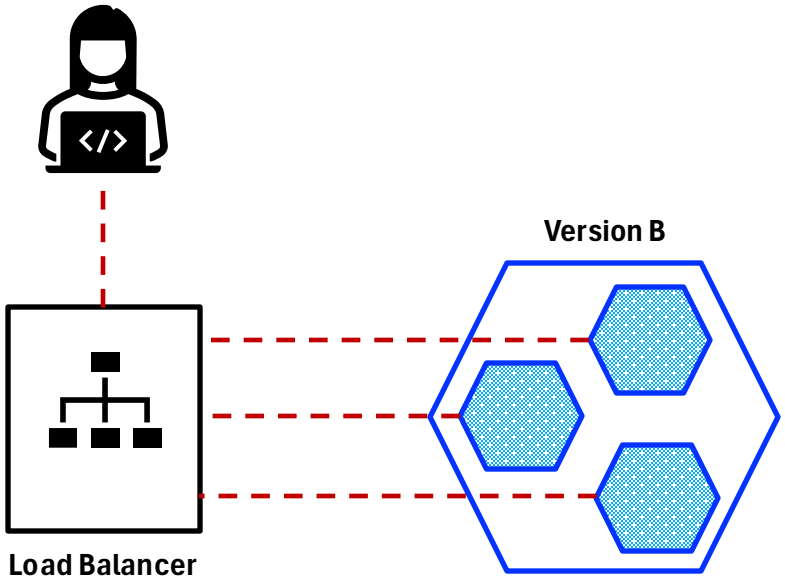
*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Shadow Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Shadow Deployment*



*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Which one to choose?*

DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Container Solutions

Strategy	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	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Which one to choose?*

Blue/Green and Shadow strategies have more impact on the **budget** as it requires double resource capacity.

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Container Solutions

Strategy	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	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Which one to choose?*

If the application **lacks in tests** or if there is **little confidence** about the **impact/stability** of the software, then a Canary, A/B testing or Shadow release can be used.

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Container Solutions

Strategy	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	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

Which one to choose?*

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system, then you may want to use the A/B testing technique.

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Container Solutions

Strategy	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	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

*Source: Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)

References

- https://c4model.com/
- https://github.com/structurizr/examples
- https://structurizr.com/share/36141
- Tremel, E. (2017): Six strategies for application deployment (https://thenewstack.io/deployment-strategies/)
- Pautasso, C., Software Architecture- Visual Lecture Notes, LeanPub, 2023 (https://leanpub.com/software-architecture/)
- Introduction to DevOps by Len Bass – URL: goo.gl/iMwdfg
- Len Bass, Ingo Weber, Liming Zhu, DevOps: A Software Architect's Perspective, 2015
- Martin Fowler (2024), "Branch By Abstraction" https://martinfowler.com/bliki/BranchByAbstraction.html
- Ian Cartwright, Rob Horn, and James Lewis (2024), Patterns of Legacy Displacement (https://martinfowler.com/articles/patterns-legacy-displacement/)
- The University of Queensland's Software Architecture Course Materials, @ Richard Thomas, CC BY-SA 4.0 (https://github.com/CSSE6400)
- Neal Ford, Mark Richards, Pramod Sadalage, Zhamak Dehghani, Software Architecture: the Hard Parts, O'Reilly Media, 2021

References

- Mojtaba Shahin, Mansooreh Zahedi, Muhammad Ali Babar and Liming Zhu, An Empirical Study of Architecting for Continuous Delivery and Deployment, In: Empirical Software Engineering, 24(3), 2019, Springer
- Mojtaba Shahin, Muhammad Ali Babar and Liming Zhu, Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices, In: IEEE Access, 5 (99), 2017, IEEE.
- Matthew Skelton, Chris O'Dell , Continuous Delivery with Windows and .NET, 2016, O'Reilly Media, Inc.