Efficiency in Motion: Mastering Continuous Delivery without Compromising Stability

Naresh Waswani

Senior Architect, Simpplr Inc.

About Me

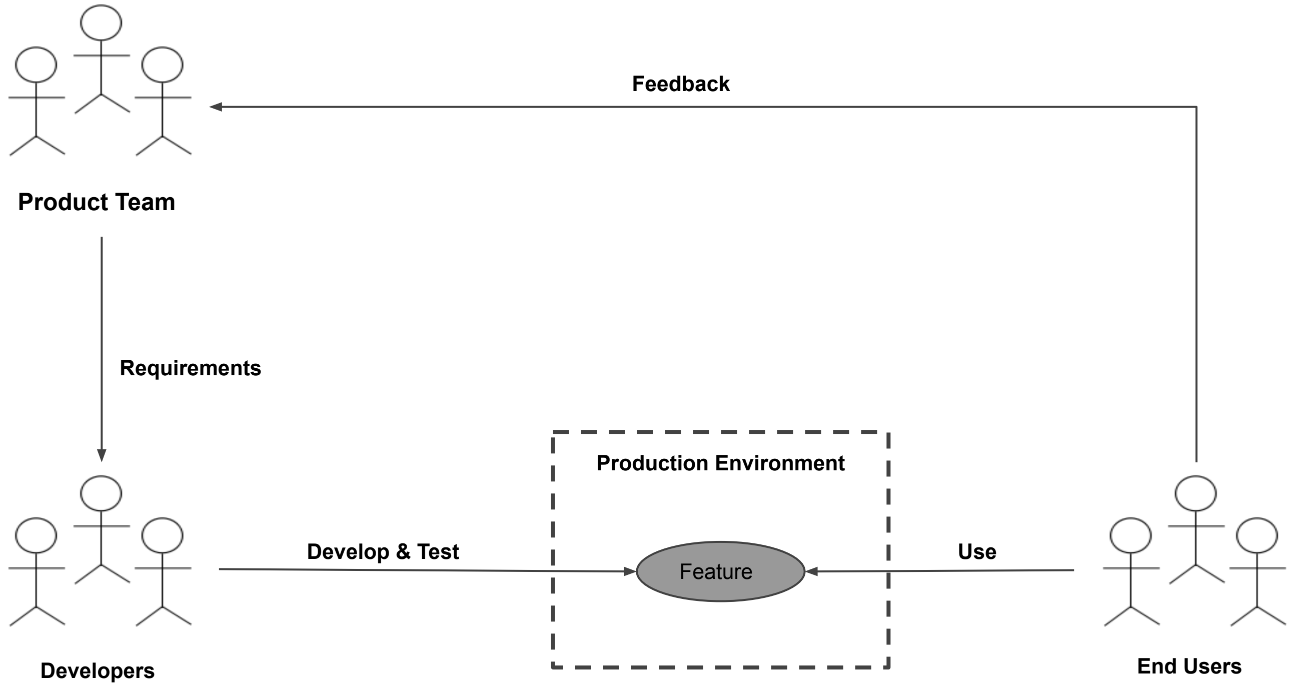
Work as a Senior Architect with Simpplr Inc.

Specialize in creating software products using Microservices and Event Driven Architecture

Publish Technical blogs on - https://waswani.medium.com

To Connect - https://www.linkedin.com/in/nwaswani/

Let’s Dive In

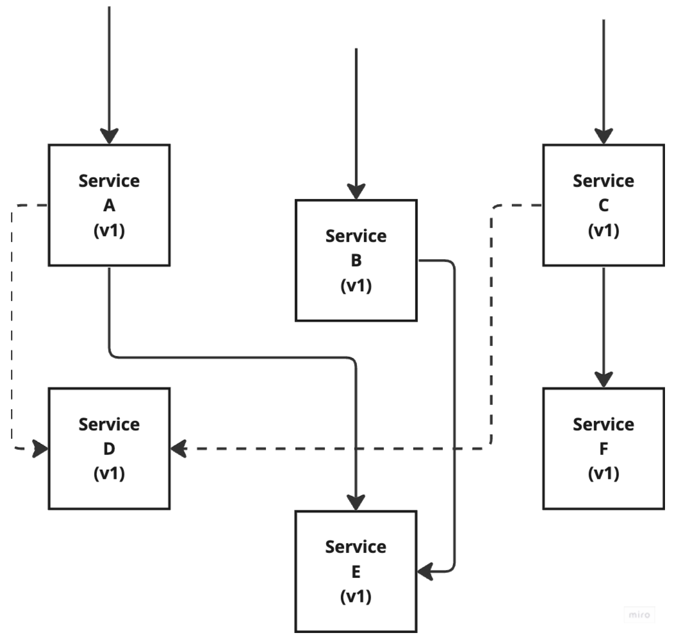


Feedback
Product Team
Requirements
Production Environment
Develop & Test
Feature
Use
Developers
End Users

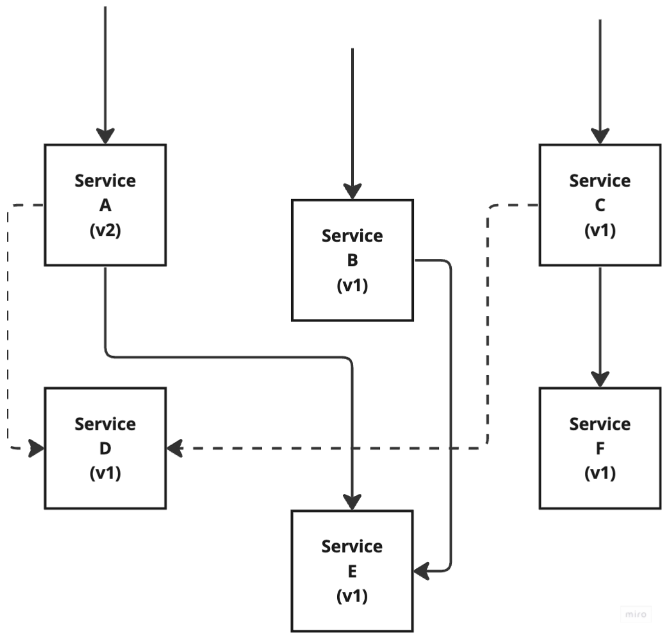
Continuous Delivery is the Software Development Process of getting the code changes deployed to production ***quickly***, ***safely*** and with higher ***quality***.

A Microservice is an ***independent deployable*** unit, ***modeled around a business domain*** and ***generally collaborate*** with other microservices to deliver a larger business use case.

Key Advantage - Team Autonomous

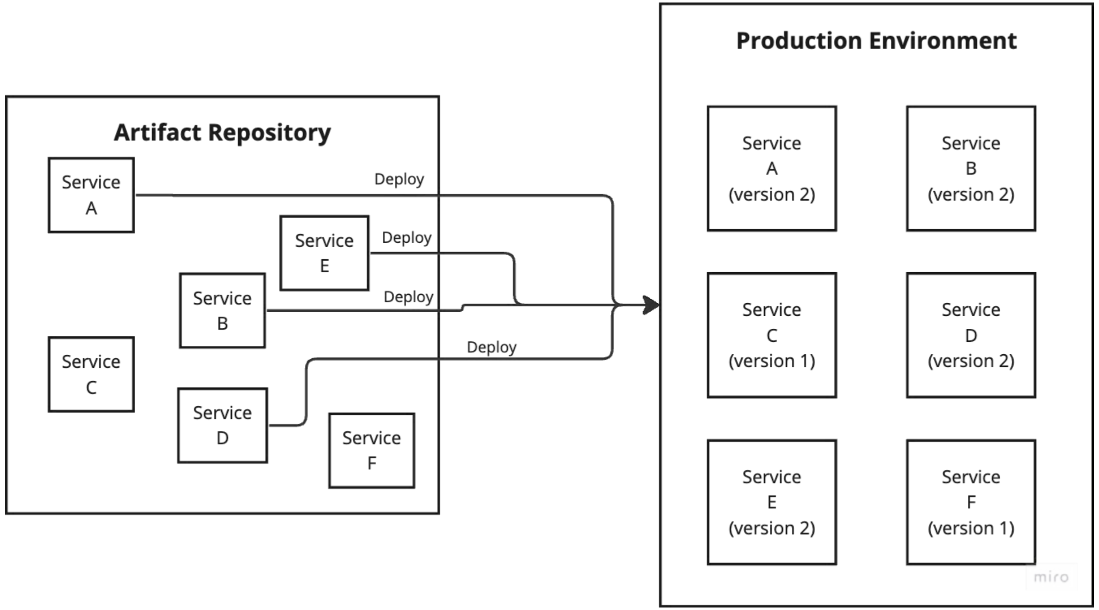


Key Advantage - Team Autonomous



But when it comes to shipping the Features, things go crazy

Deployment Pattern



Is that not Distributed Monolith ???

What can lead to Distributed Monolith

- Inappropriate Service Boundary

What can lead to Distributed Monolith

- Inappropriate Service Boundary
- Shared Data Storage

What can lead to Distributed Monolith

- Inappropriate Service Boundary
- Shared Data Storage
- Too Many Shared Libraries

There is one more reason for such a Deployment Pattern

Pre-defined Feature Release Schedule

Some of the Non-Technical reasons

- You want to create a Market Buzz

Some of the Non-Technical reasons

- You want to create a Market Buzz
- Customers don’t really have an appetite to absorb so many features in a short time

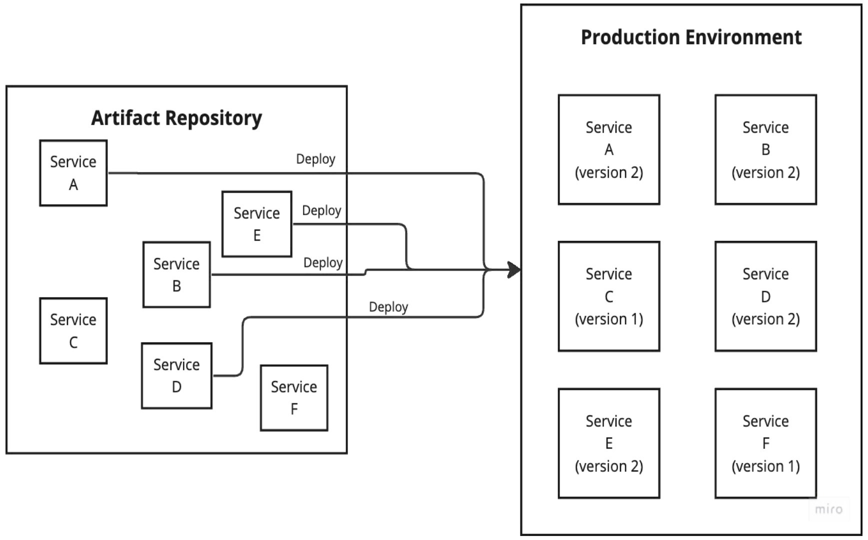
Some of the Non-Technical reasons

- You want to create a Market Buzz
- Customers don’t really have an appetite to absorb so many features in a short time
- You want to share Penetration testing report of your product

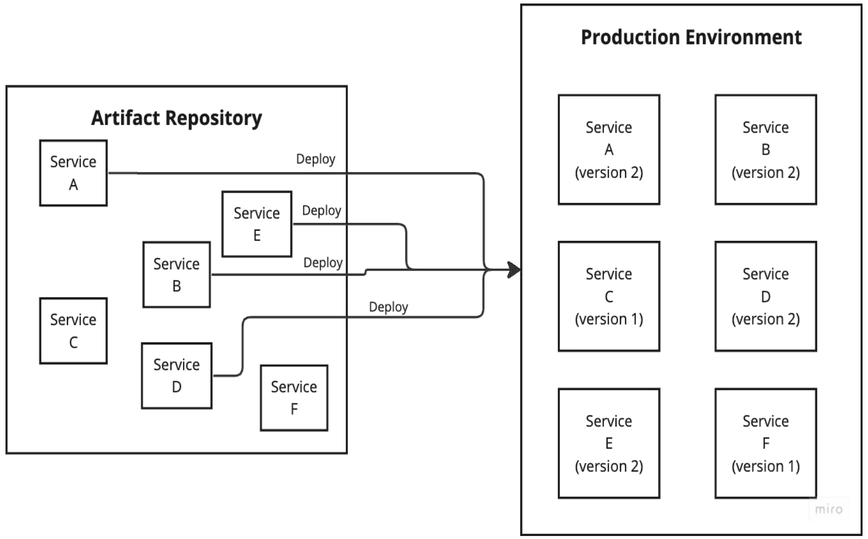
Some of the Non-Technical reasons

- You want to create a Market Buzz
- Customers don’t really have an appetite to absorb so many features in a short time
- You want to share Penetration testing report of your product
- You need to train your CS teams, publish user guides

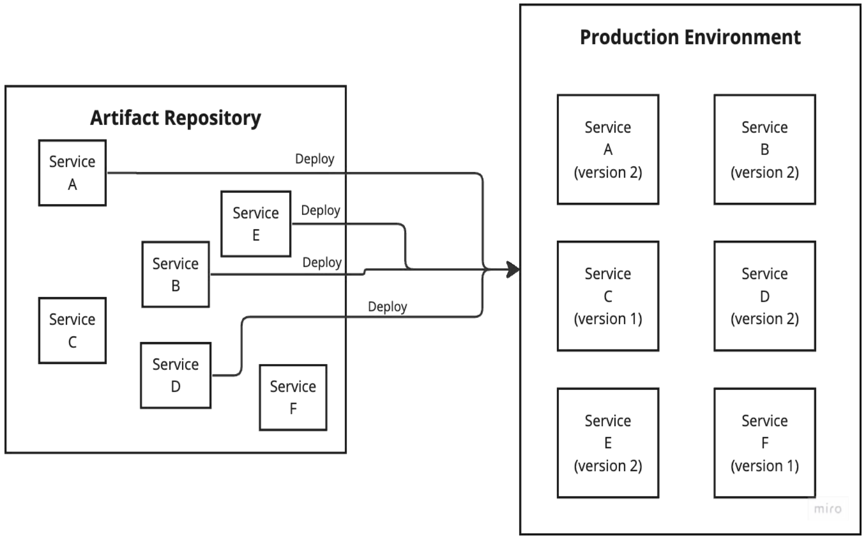
The Problem !!!


Artifact Repository
Service A
Service E
Service B
Service C
Service D
Service F
Deploy
Deploy
Deploy
Deploy
Production Environment
Service A (version 2)
Service B (version 2)
Service C (version 1)
Service D (version 2)
Service E (version 2)
Service F (version 1)
miro

High Risk



High Deployment Time



Burns out Engineering Team

The Solution !!!

Deploy the service as soon as a Feature is developed

But then...

What happens to the release cadence?

I still have half baked code for Feature X in the same branch?

Deployment != Feature Release

Feature Toggle to the rescue

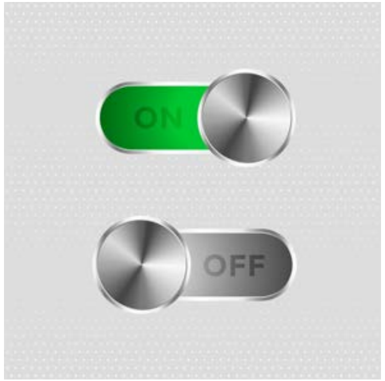


Image Credit - https://www.freepik.com/free-photos-vectors/on-off-toggle-switch

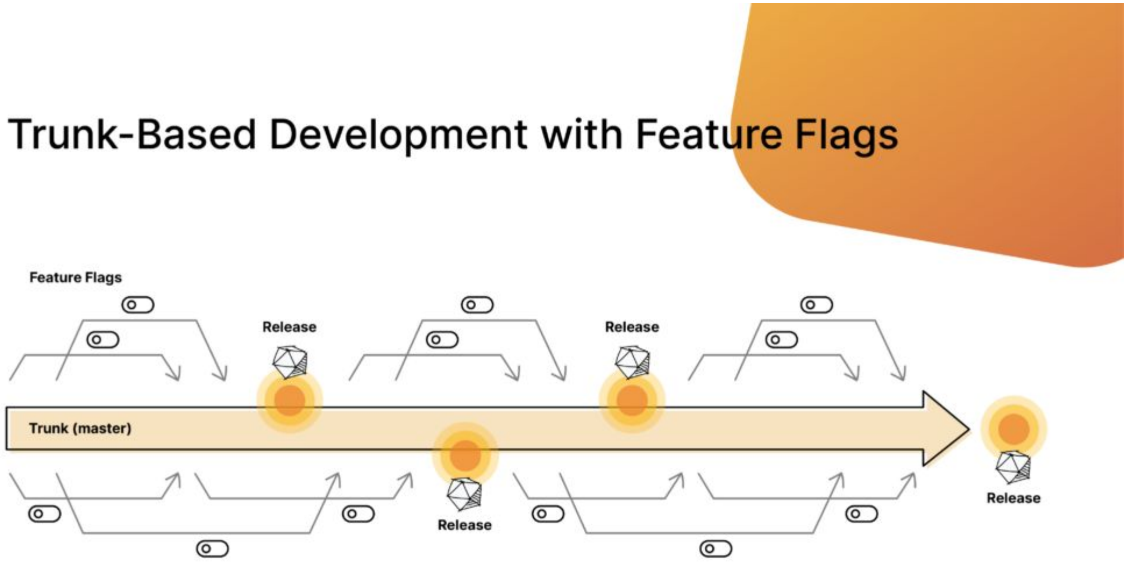
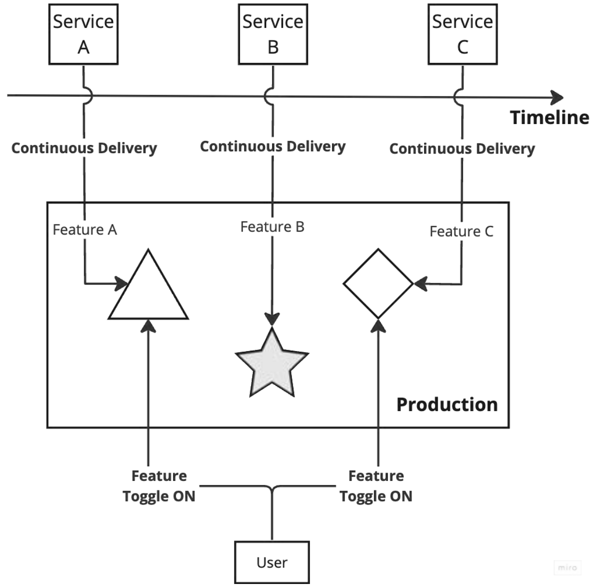


Image Credit - https://developer.harness.io/assets/images/overview-1f5470a387b20eafc3014f98233f200e.png

Continuous Delivery

Services shipping their features behind a feature flag as an when their features are developed and tested



Low Risk, Zero Downtime

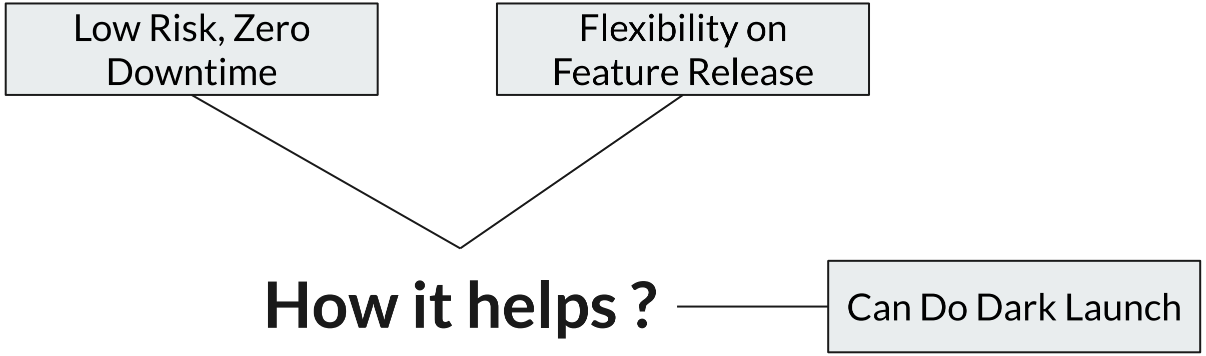
How it helps ?

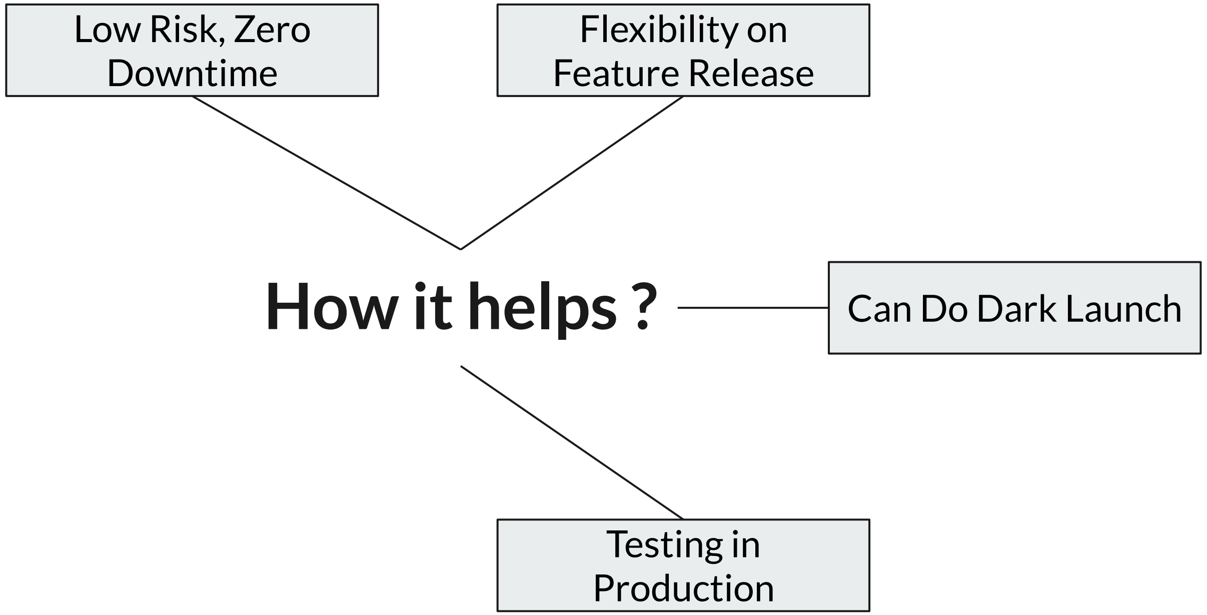
Low Risk, Zero Downtime

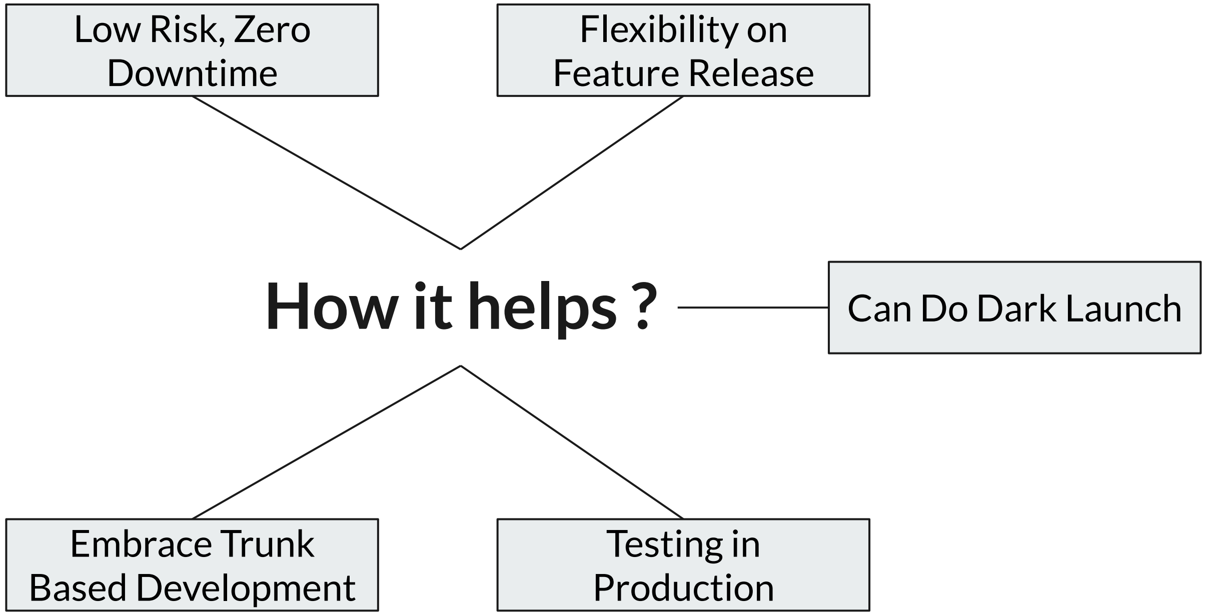
Flexibility on Feature Release

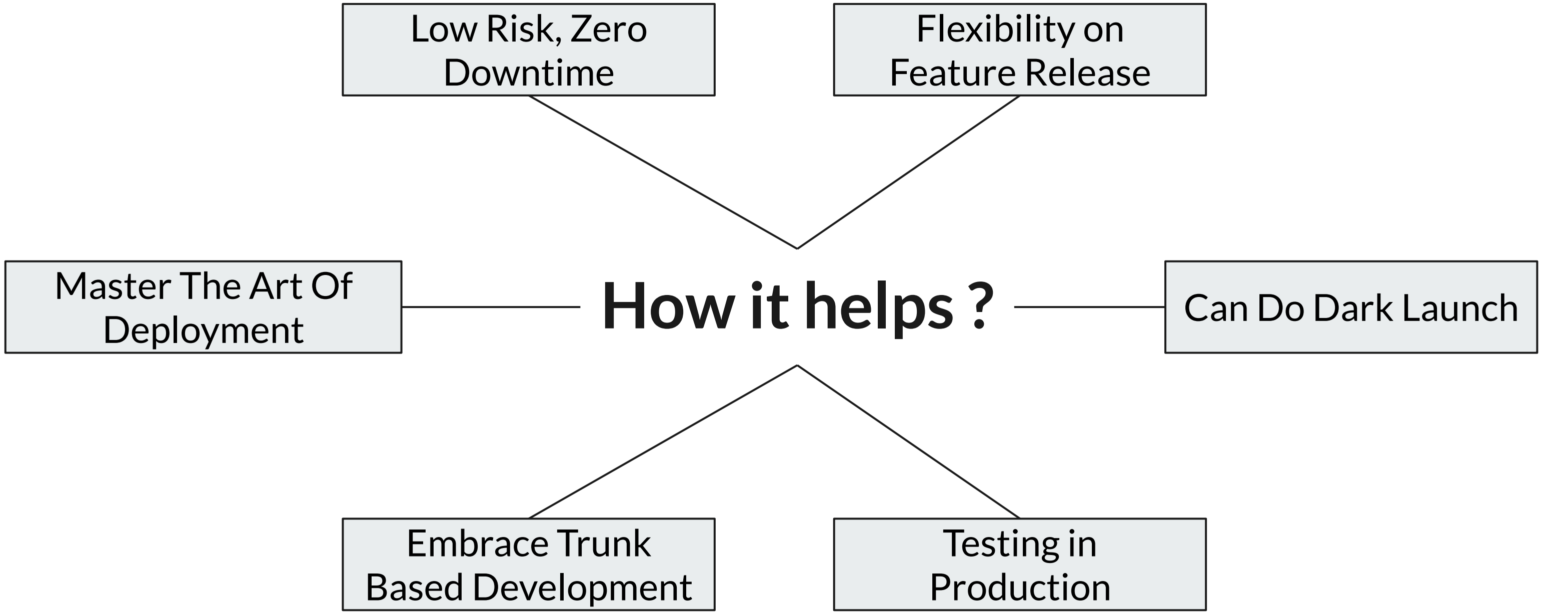
How it helps ?

How it helps ?


Low Risk, Zero Downtime
Flexibility on Feature Release
Can Do Dark Launch


Low Risk, Zero Downtime
Flexibility on Feature Release
How it helps ?
Can Do Dark Launch
Testing in Production

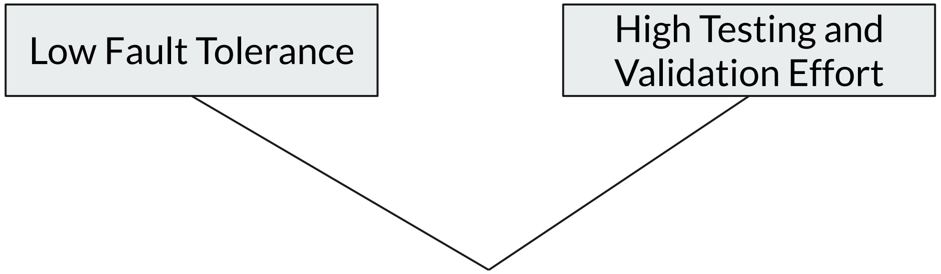

Low Risk, Zero Downtime
Flexibility on Feature Release
How it helps ?
Can Do Dark Launch
Embrace Trunk Based Development
Testing in Production


How it helps ?
Low Risk, Zero Downtime
Flexibility on Feature Release
Master The Art Of Deployment
Can Do Dark Launch
Embrace Trunk Based Development
Testing in Production

But as the saying goes, nothing comes for free !!!

Low Fault Tolerance

Challenges ?



Challenges ?

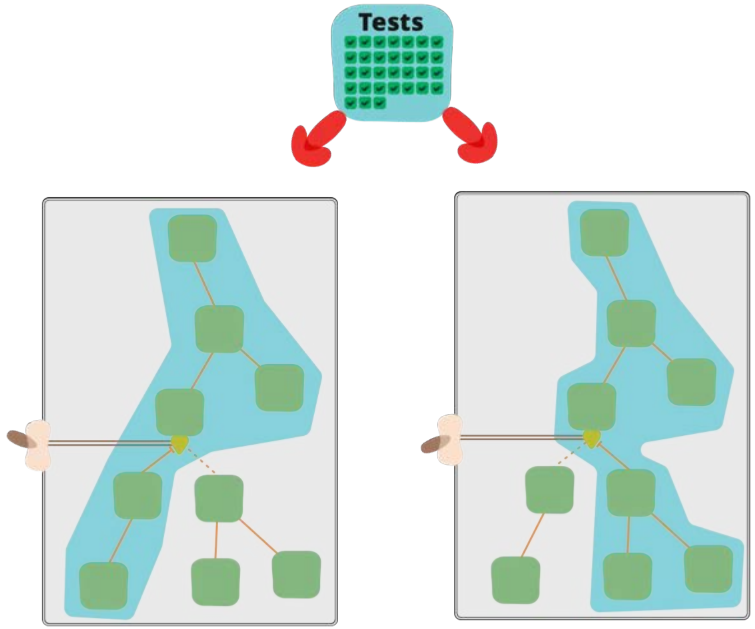
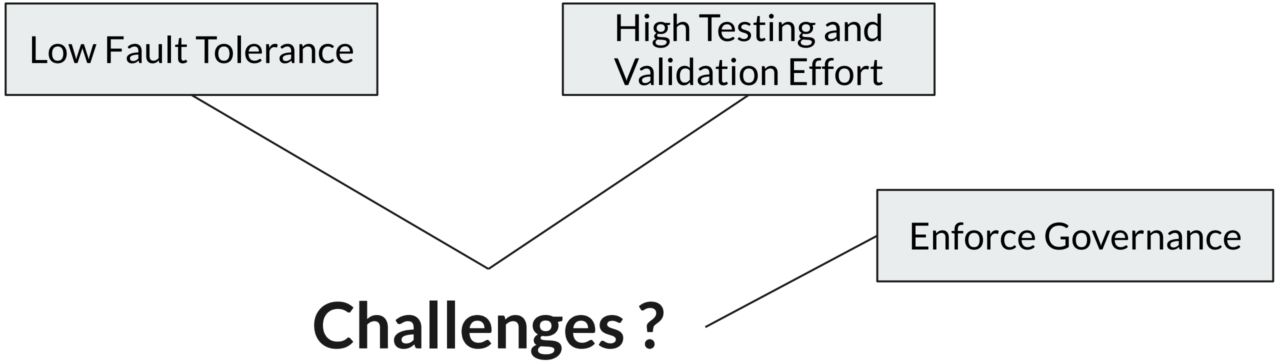
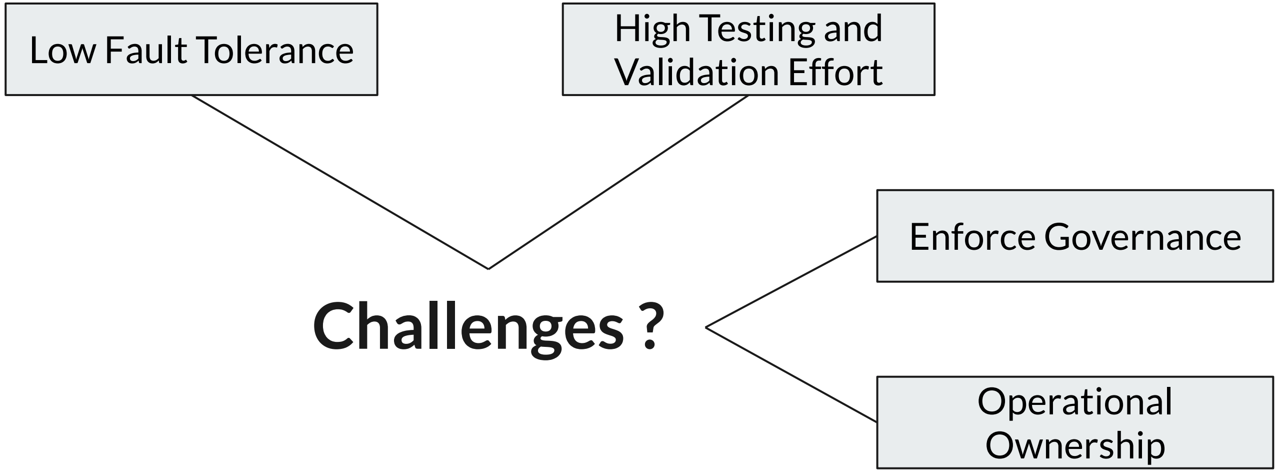
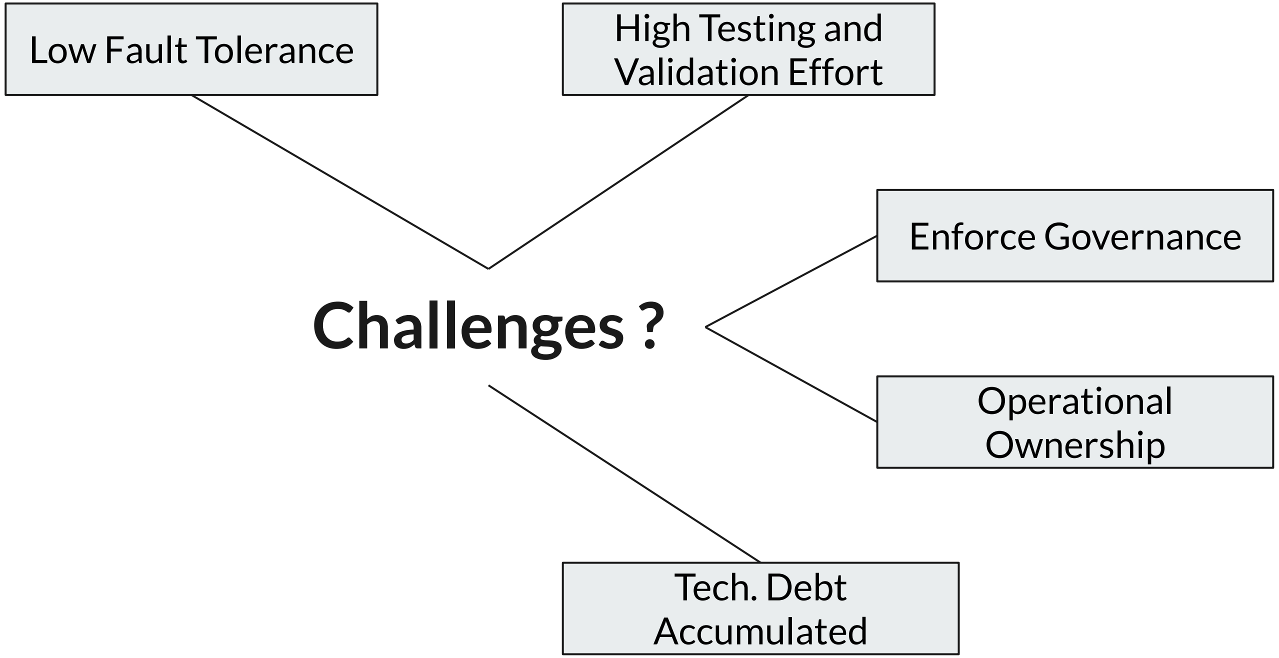
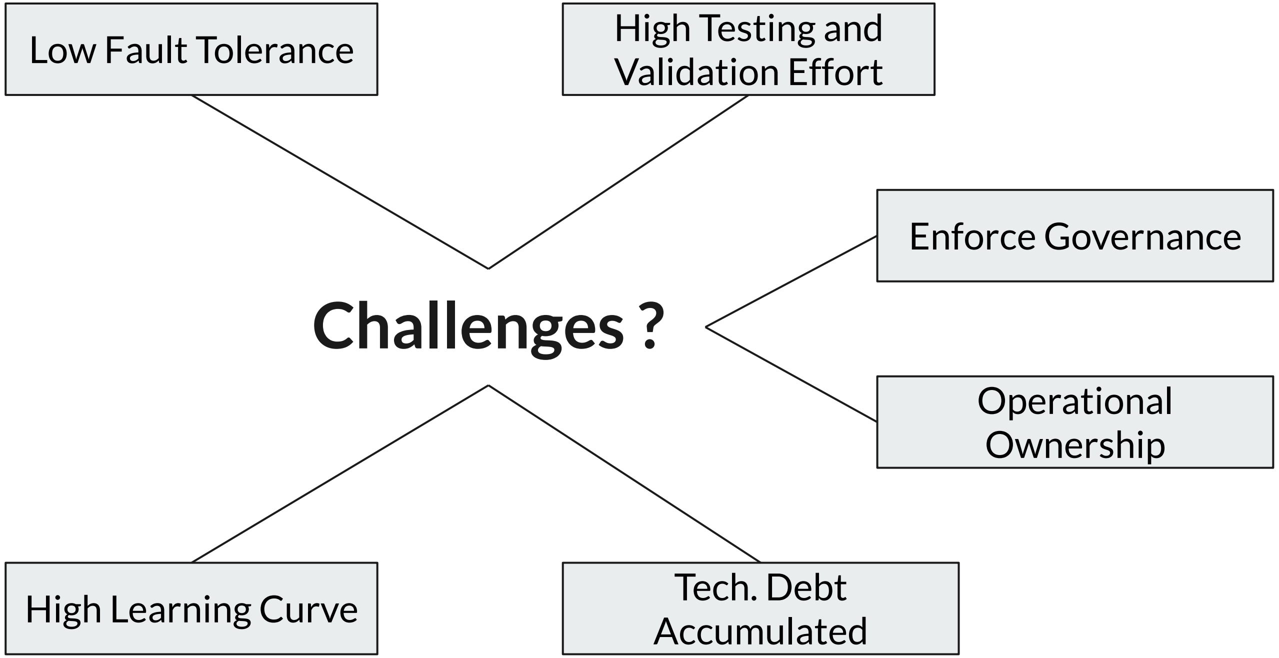


Image Credit - https://martinfowler.com/articles/feature-toggles/feature-toggles-testing.png


Low Fault Tolerance
High Testing and Validation Effort
Enforce Governance
Challenges ?

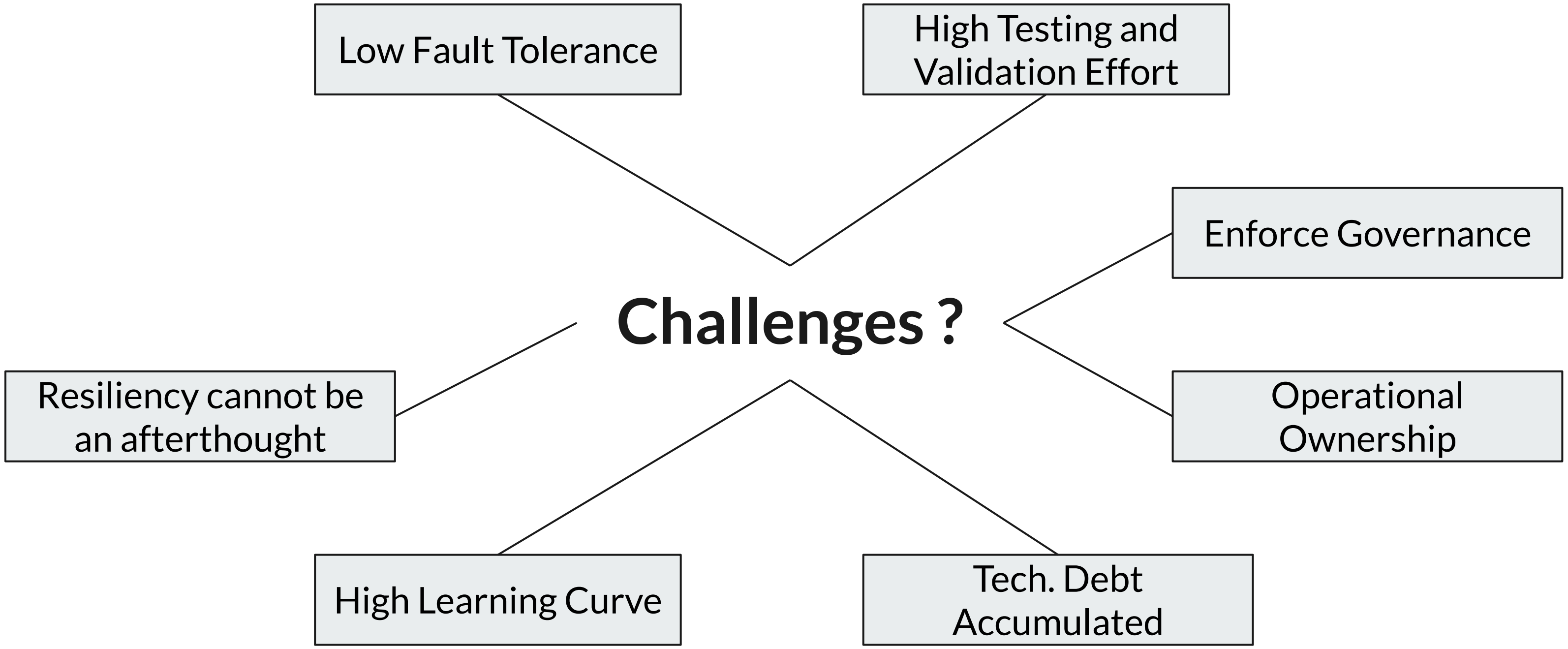

Low Fault Tolerance
High Testing and Validation Effort
Challenges ?
Enforce Governance
Operational Ownership

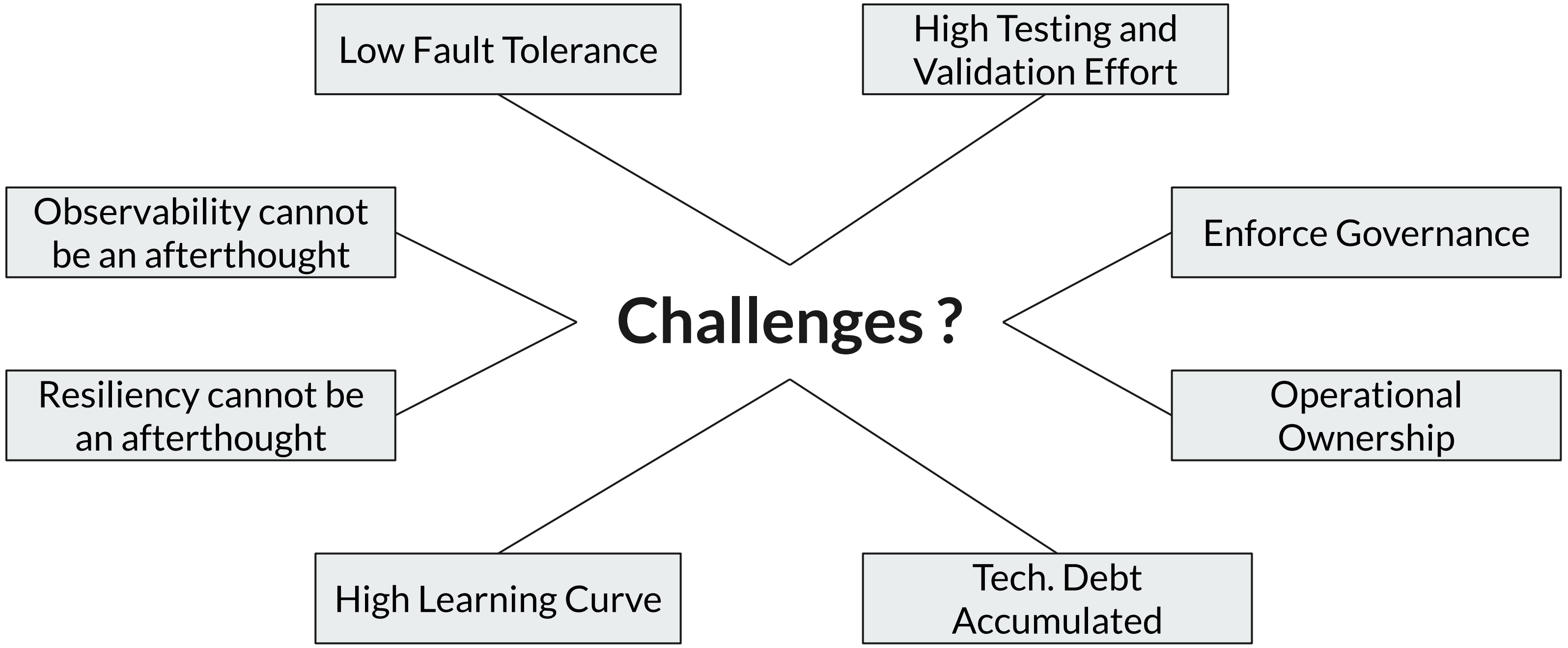

Low Fault Tolerance
High Testing and Validation Effort
Challenges ?
Enforce Governance
Operational Ownership
Tech. Debt Accumulated



Low Fault Tolerance
High Testing and Validation Effort
Enforce Governance
Operational Ownership
High Learning Curve
Tech. Debt Accumulated

Challenges ?


Challenges ?
Low Fault Tolerance
High Testing and Validation Effort
Enforce Governance
Operational Ownership
Resiliency cannot be an afterthought
High Learning Curve
Tech. Debt Accumulated


Low Fault Tolerance
High Testing and Validation Effort
Observability cannot be an afterthought
Enforce Governance
Challenges ?
Resiliency cannot be an afterthought
Operational Ownership
High Learning Curve
Tech. Debt Accumulated

Summary

Summary

Use Feature Flags and Define Operational & Governance model for it

Embrace Trunk Driven Development

Ensure Feature verification via Test Automation with Toggle Flags on and off

Summary

Use Feature Flags and Define Operational & Governance model for it

Embrace Trunk Driven Development

Ensure Feature verification via Test Automation with Toggle Flags on and off

And you are all set !!!