DevOps

Introduction to DevOps

Chandara Chea

9/10/2026
Week I

Assessment

Attendance	10%
Quizzes	20%
Presentation	20%
Project	30%
Final Exam	20%

Contents

What is DevOps?

History

Key Benefits

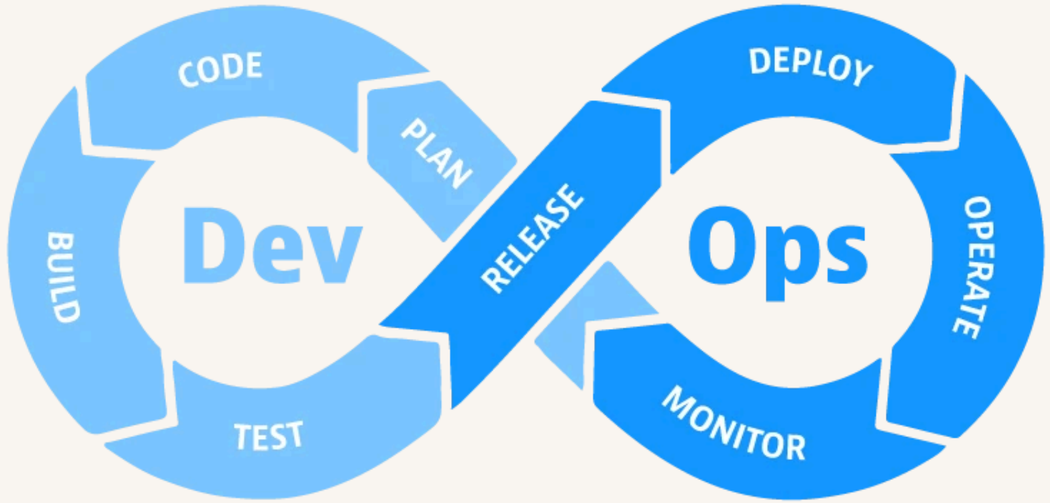
The Three Ways

DevOps Life Cycle

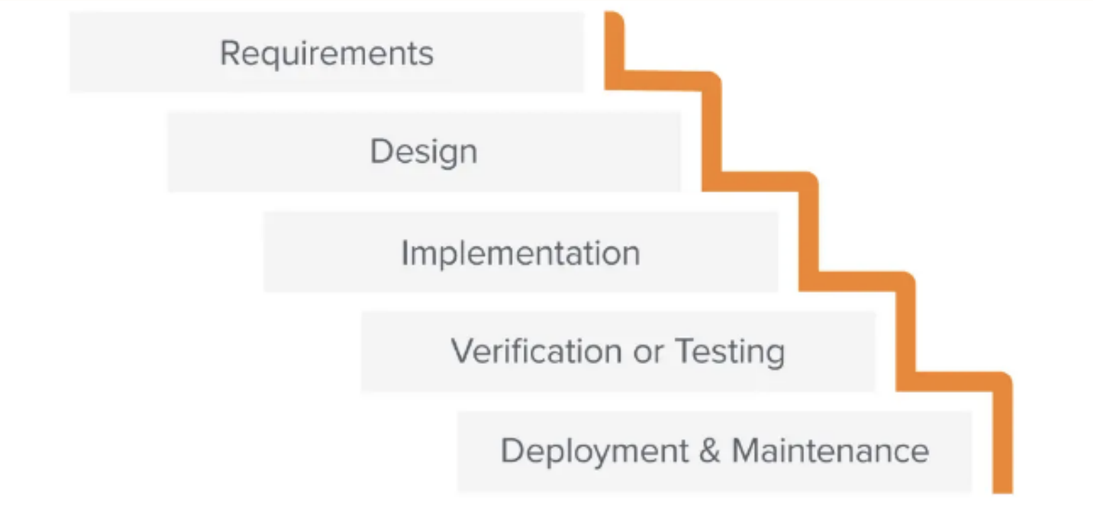
What is DevOps

DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

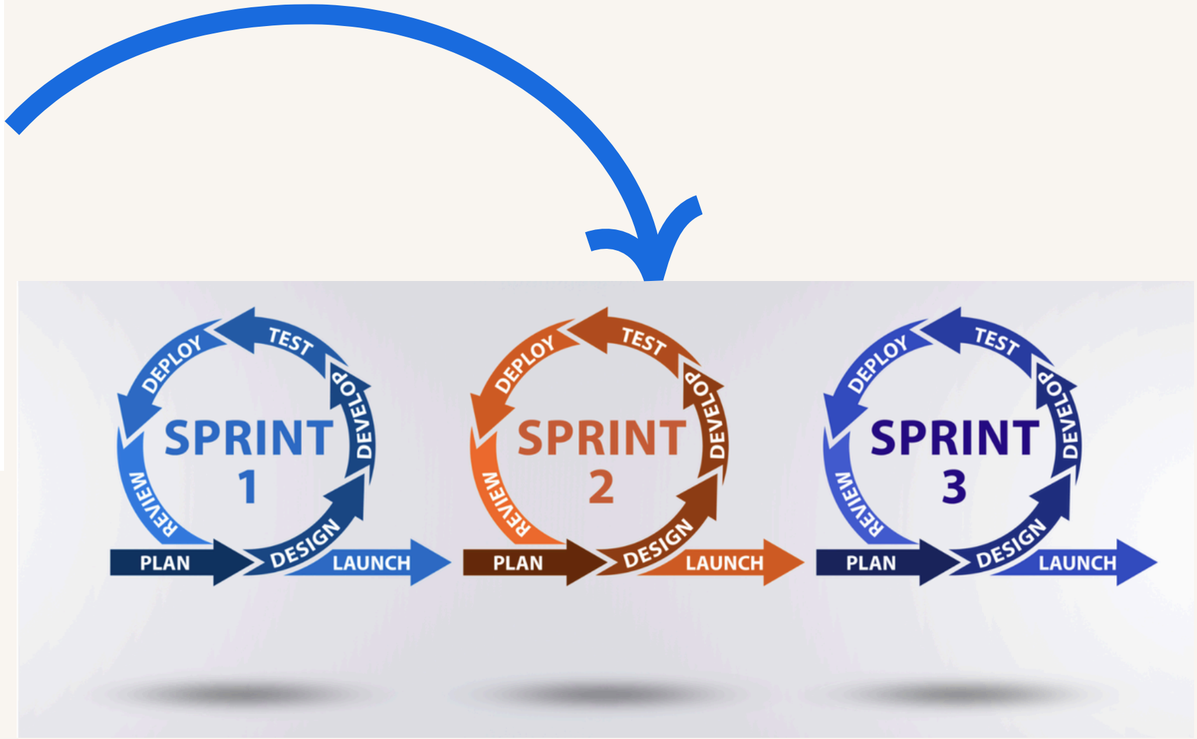
DevOps is more than just development and operations teams working together. It’s more than tools and practices. DevOps is a mindset, a cultural shift, where teams adopt new ways of working.



History of DevOps

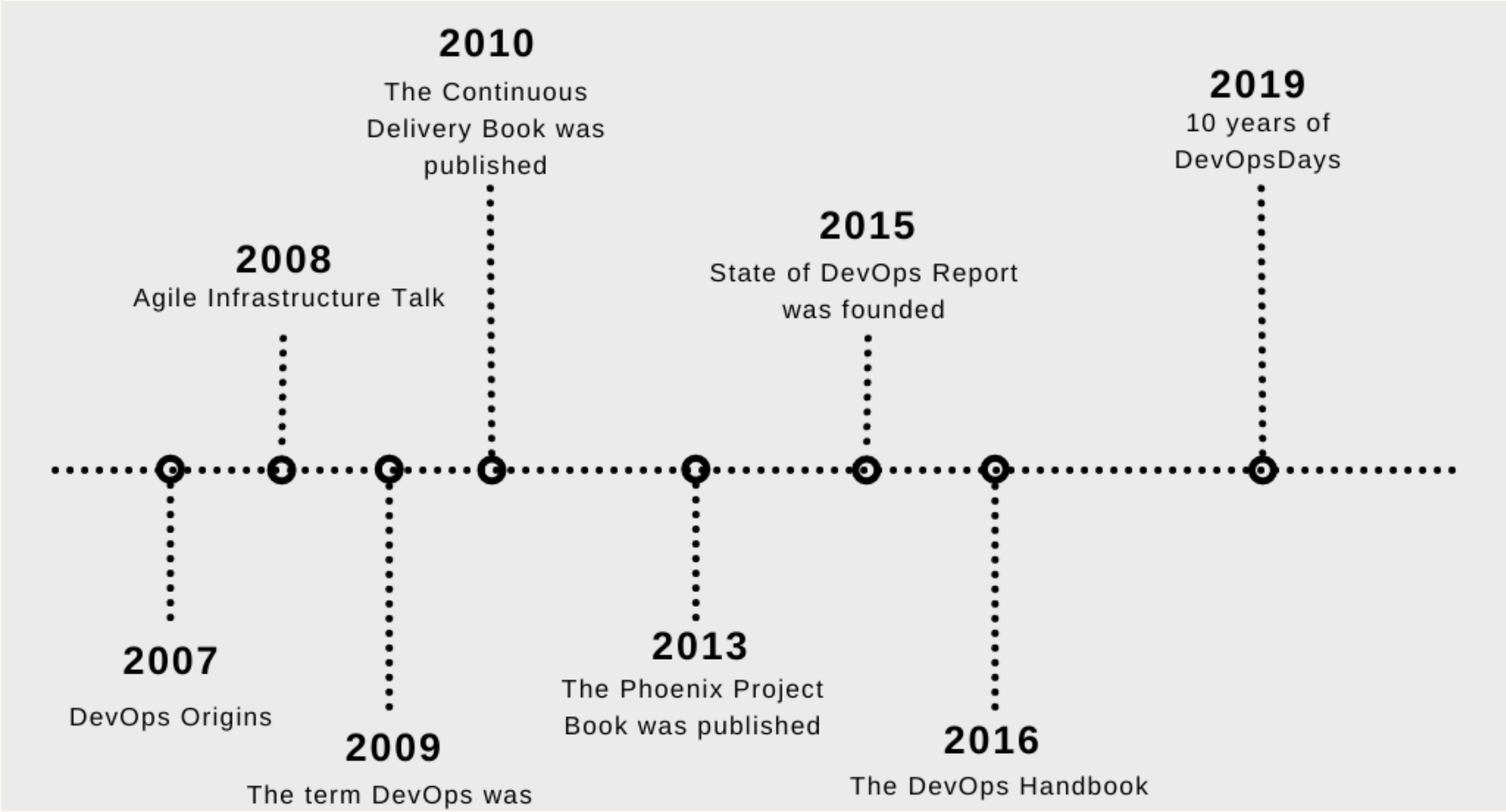


Waterfall Model



Agile Methodology

History of DevOps



Key Benefits of DevOps



Enhanced Collaboration



Increased Efficiency



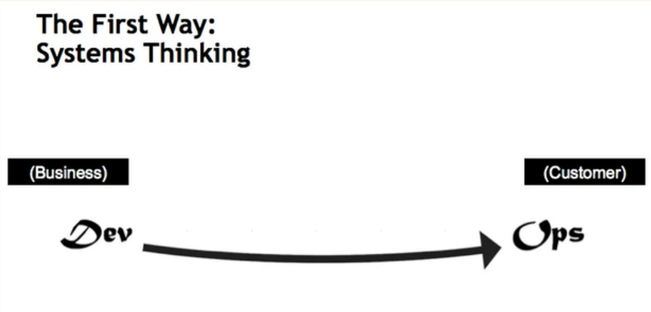
Faster Time to Market

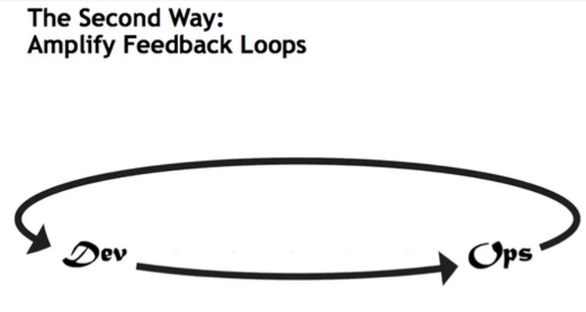


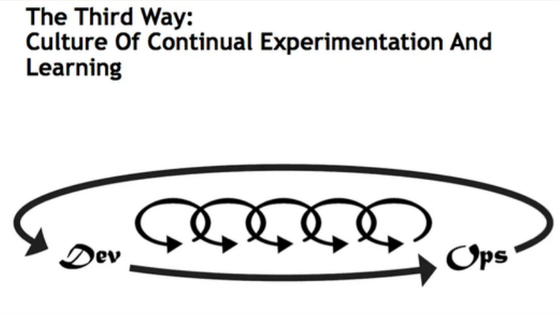
Higher Quality and Security

The Three Ways of DevOps

- The most respcted set principles
- Developed by Gene Kim, the author of The Phoenix Project and The DevOps Handbook

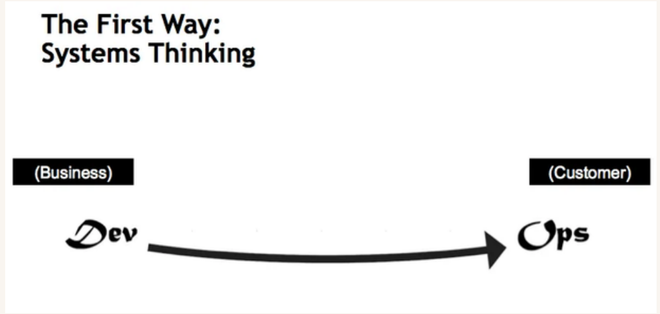


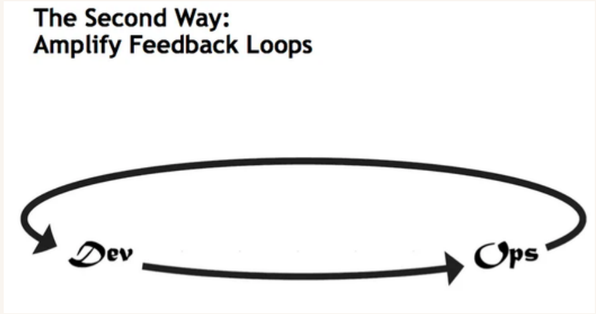


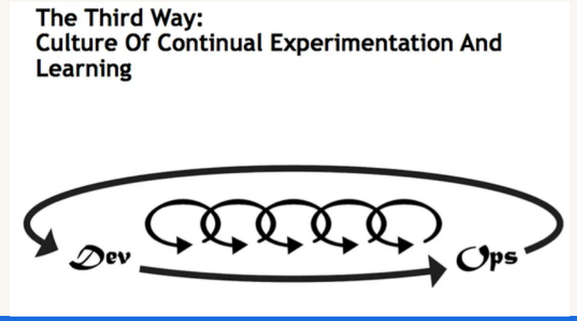


The Three Ways of DevOps

1. The First Way: Flow/Systems Thinking
2. The Second Way: Amplify Feedback Loops
3. The Third Way: Culture of Continual Experimentation and Learning

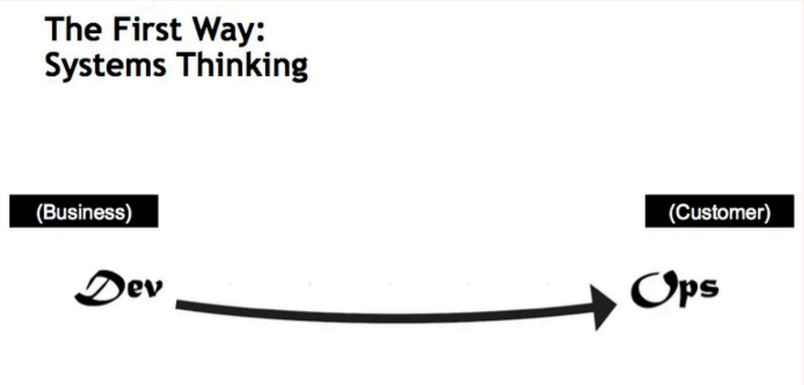






The First Way: Flow/Systems Thinking

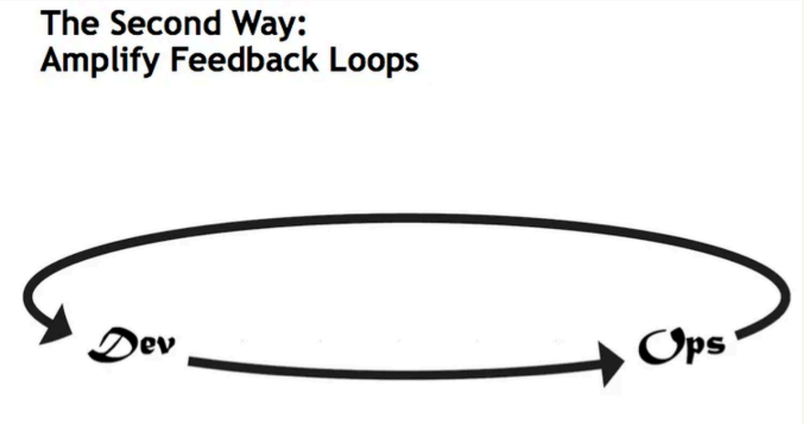
- Focus on the overall outcome of the entire system.
- You have to understand the whole system to optimize it well.
- Use systems thinking as guidance when defining success metrics and evaluating the outcome of changes.
- Overall flow from beginning to end is how you actually produce value.



Eliminate bottlenecks, reducing batch sizes, and accelerating workflow from development to production.

The Second Way: Amplify Feedback Loops

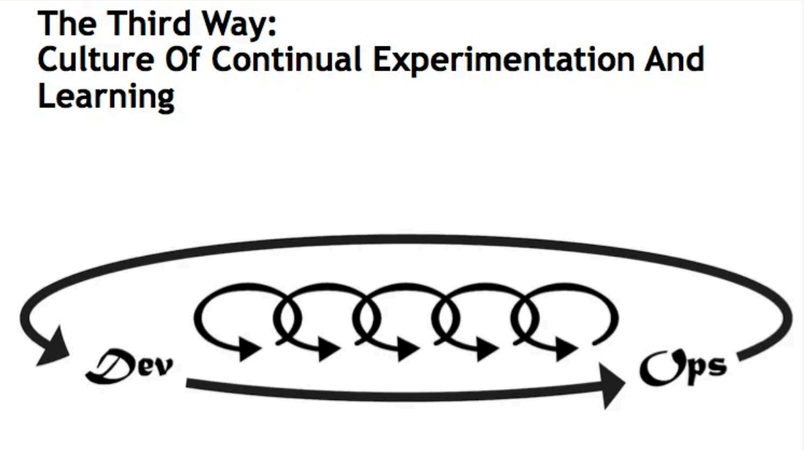
- Focus on creating, shortening, and amplifying feedback loops between the parts of the organization that are in the flow of the value chain.
- Effective feedback loops are the key to productive product development, software development and operations.
- Use amplifying feedback loops to help you when you're creating multi-team processes, visualizing metrics and designing delivery flows.



Quickly and accurately receive information about any issues or inefficiencies in the system.

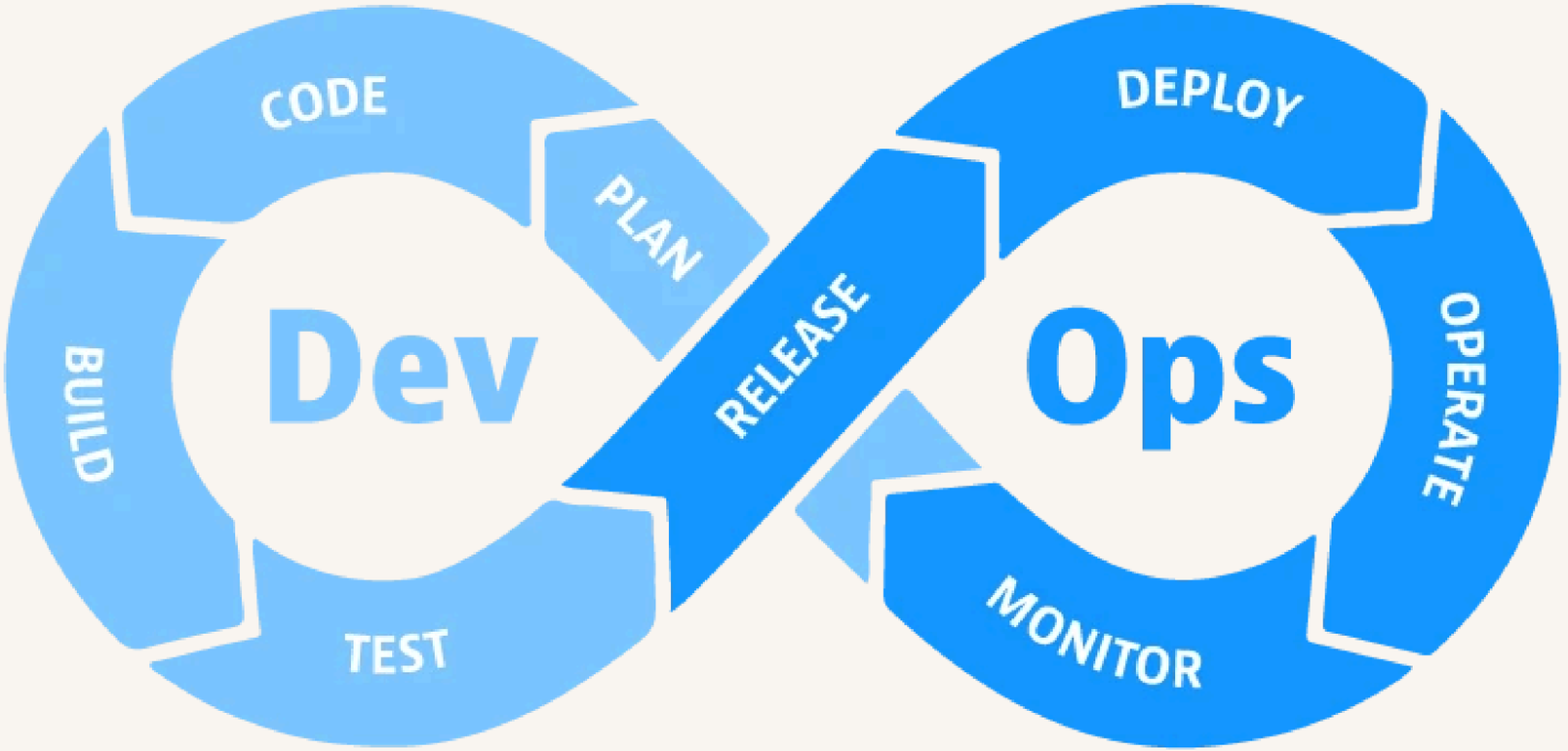
The Third Way: Culture of Continual Experimentation and Learning

- Focus on creating a work culture that allows for both continuous experimentation and learning.
- Use a culture of continuous experimentation and learning when creating team processes and standards and as part of your leadership style, encourage sharing and trying new ideas.



Encourage a culture of continuous learning, experimentation and taking risks.

DevOps Lifecycle



DevOps Lifecycle: Plan

- **What it is**: Define project requirements, features, tasks, and priorities.
- **Goal**: Ensure everyone knows what needs to be built.
- **Activities**:
  - Creating user stories, sprint plans
  - Task tracking and backlog management
- **Tools**: Jira, Trello, Asana, Azure DevOps



DevOps Lifecycle: Code

- **What it is:** Writing the actual software code.
- **Goal:** Developers collaborate efficiently and maintain quality.
- **Activities:**
  - Writing, reviewing, and merging code
  - Using version control for collaboration
- **Tools:** Git, GitHub, GitLab, Bitbucket



DevOps Lifecycle: Build

- **What it is:** Compile the code into executable packages or artifacts.
- **Goal:** Make sure code is packaged correctly for testing or deployment.
- **Activities:**
  - Compilation, dependency management
  - Building binaries or container images
- **Tools:** Maven, Gradle, npm, Docker



DevOps Lifecycle: Test

- **What it is:** Ensure the code works correctly and meets quality standards.
- **Goal:** Catch bugs early before deployment.
- **Activities:**
  - Unit tests, integration tests, automated regression tests
  - Performance testing
- **Tools:** Selenium, JUnit, TestNG, Postman, Cypress



DevOps Lifecycle: Release

- **What it is:** Prepare code to be delivered to production or staging environments.
- **Goal:** Ensure release is smooth and predictable.
- **Activities:**
  - Versioning and tagging releases
  - Approvals and release notes
- **Tools:** Jenkins, GitLab CI, Bamboo



DevOps Lifecycle: Deploy

- **What it is:** Move code to production or staging for use.
- **Goal:** Deliver features to users reliably and frequently.
- **Activities:**
  - Automated deployment scripts
  - Infrastructure setup and configuration
- **Tools:** Docker, Kubernetes, Ansible, Terraform, AWS/GCP/Azure



DevOps Lifecycle: Operate

- **What it is:** Maintain the application in production, ensuring it runs smoothly.
- **Goal:** Keep services available, scalable, and efficient.
- **Activities:**
  - Server management, scaling, load balancing
  - Deploying update & patches
  - Incident management
- **Tools:** Prometheus, Grafana, Nagios, ELK Stack



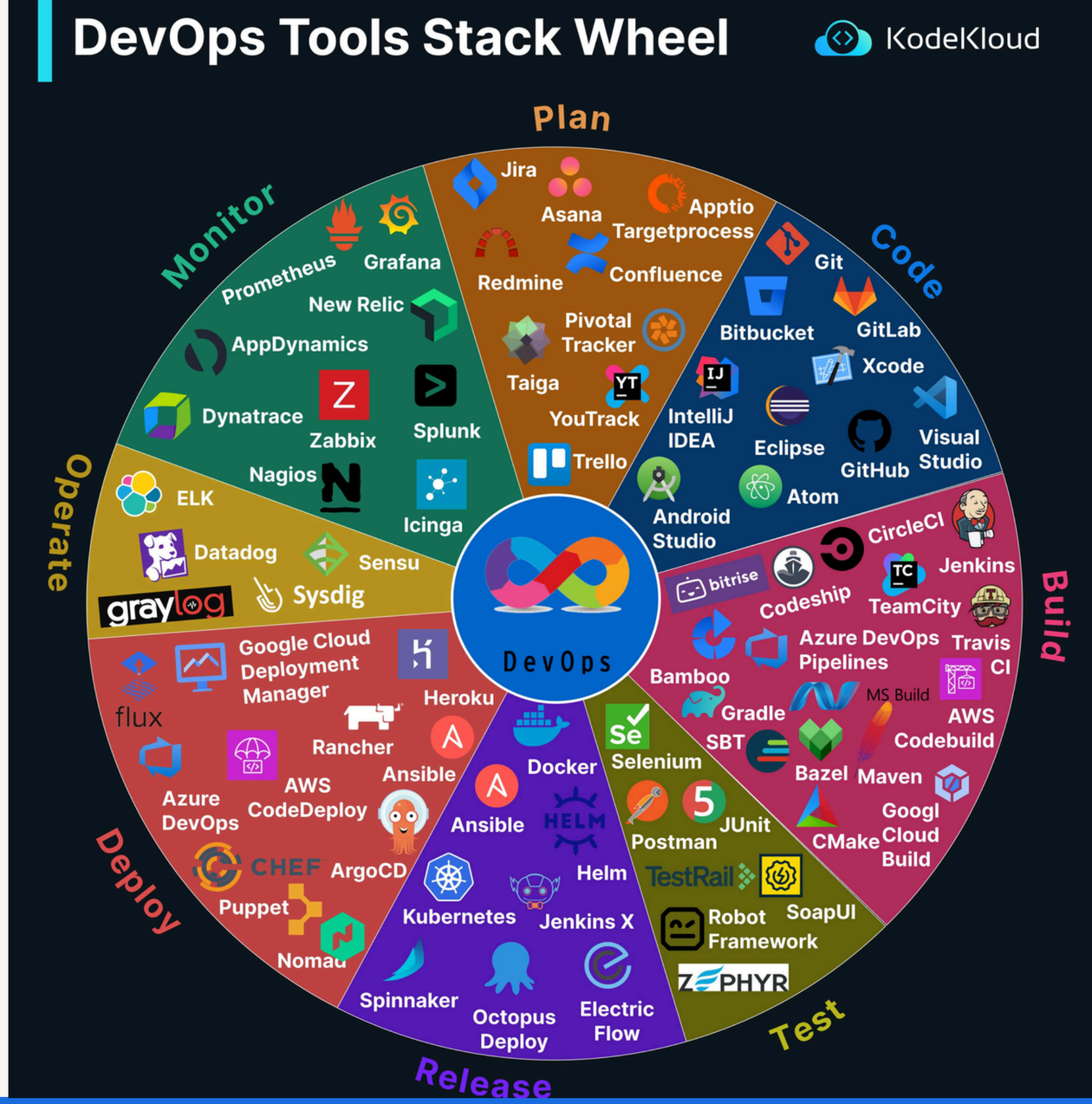
DevOps Lifecycle: Monitor

- **What it is:** Continuously observe application performance and user experience.
- **Goal:** Identify issues early and gather feedback for improvement.
- **Activities:**
  - Metrics collection, logging, alerting
  - Performance and error monitoring
- **Tools:** Prometheus, Grafana, ELK Stack, New Relic, Datadog



DevOps Lifecycle

Planning: Jira, Trello
Code/Version control: Git, GitHub, GitLab, Bitbucket
Build/CI: Jenkins, Maven, Gradle
Test: Selenium, JUnit, Postman
Deploy/Release: Docker, Kubernetes, Ansible, Spinnaker
Monitor/Operate: Prometheus, Grafana, ELK, Datadog, Nagios



DevOps

DevOps in Practice

Chandara Chea

9/10/2026

Week I

Contents

DevOps Roadmap

CI/CD vs DevOps

CI/CD

CI/CD workflow

Benefits of CI/CD Pipeline

Tools



DevOps Roadmap

Who is DevOps Engineer?



DevOps Engineer Job Description | Hiring on LinkedIn

Finding the right DevOps engineer for your company starts with creating a good job posting. This job description...

Hire

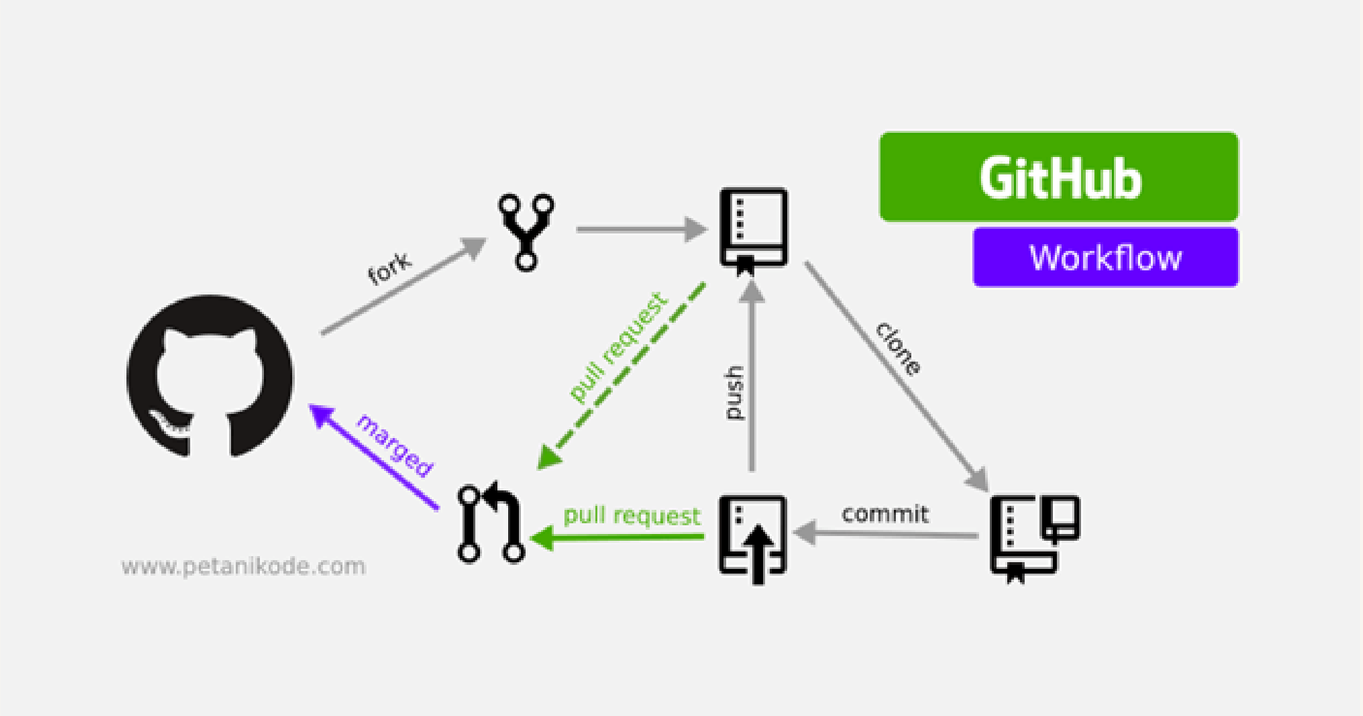
Version Control

In DevOps, Git is the single source of truth and foundation of collaboration.

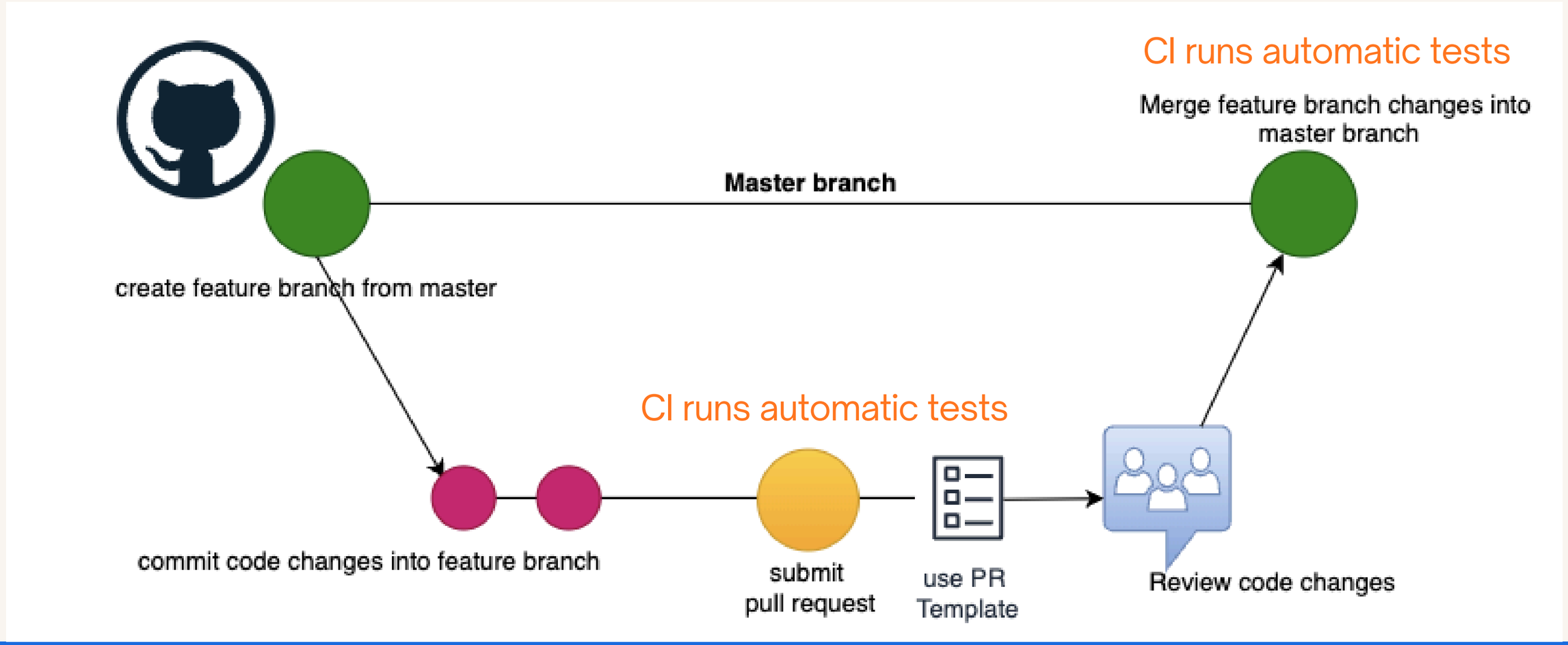
- tracks code
- enables teamwork, and
- triggers automation.

Git events trigger automation:

- push
- pull_request
- merge
- tag



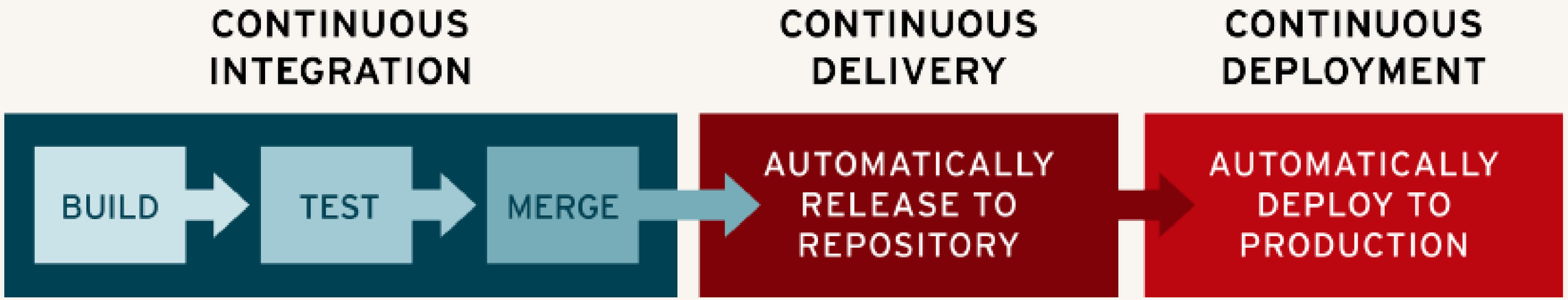
Pull Request Workflow



CI/CD

CI/CD stands for **Continuous Integration** and **Continuous Delivery (or Deployment)**. It's a DevOps practice that automates building, testing, and deploying code changes, enabling faster and more reliable software releases.

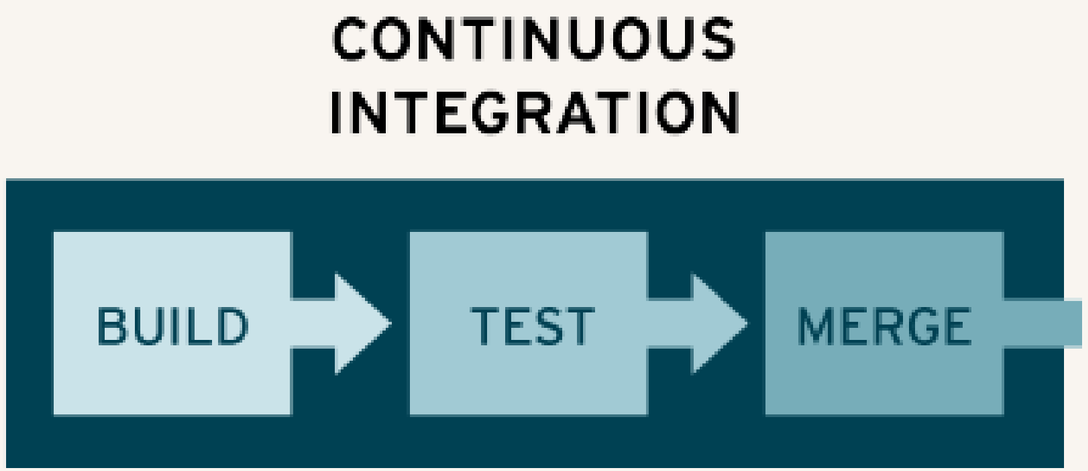
CI/CD is best explained as an automated workflow that replaces manual steps with pipelines that build, test, and deploy software reliably.



Continuous Integration (CI)

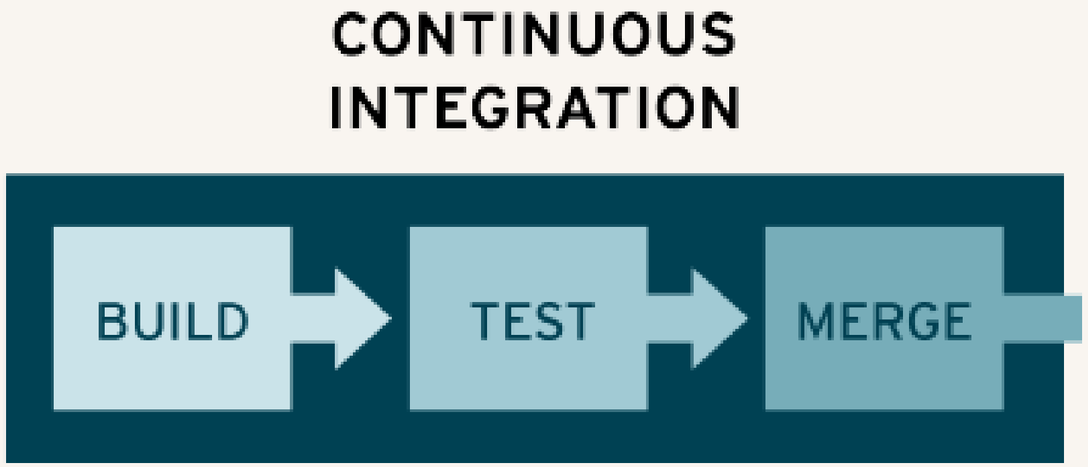
Continuous integration (CI) refers to the practice of automatically and frequently integrating code changes into a shared source code repository.

Successful CI means that once a developer's changes to an application are merged, those changes are validated by automatically building the application and running different levels of automated testing, typically unit and integration tests, to ensure the changes haven't broken the app.



Setup Continuous Integration

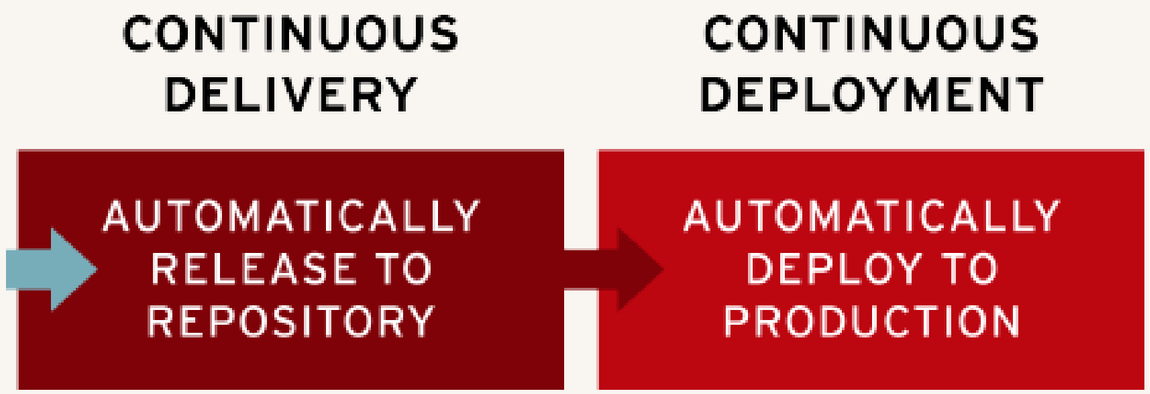
1. Start writing tests for the critical parts of your codebase.
2. Get a CI service to run those tests automatically on every push to the main repository.
3. Make sure that your team integrates their changes everyday.
4. Fix the build as soon as it’s broken.
5. Write tests for every new story that you implement.



Continuous Delivery/Deployment (CD)

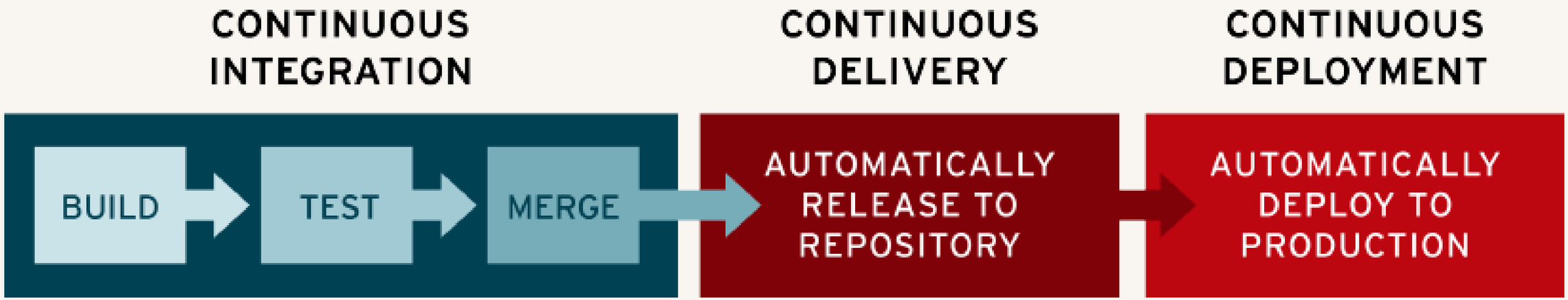
Continuous delivery and/or deployment (CD) is a 2 part process that refers to the integration, testing, and delivery of code changes.

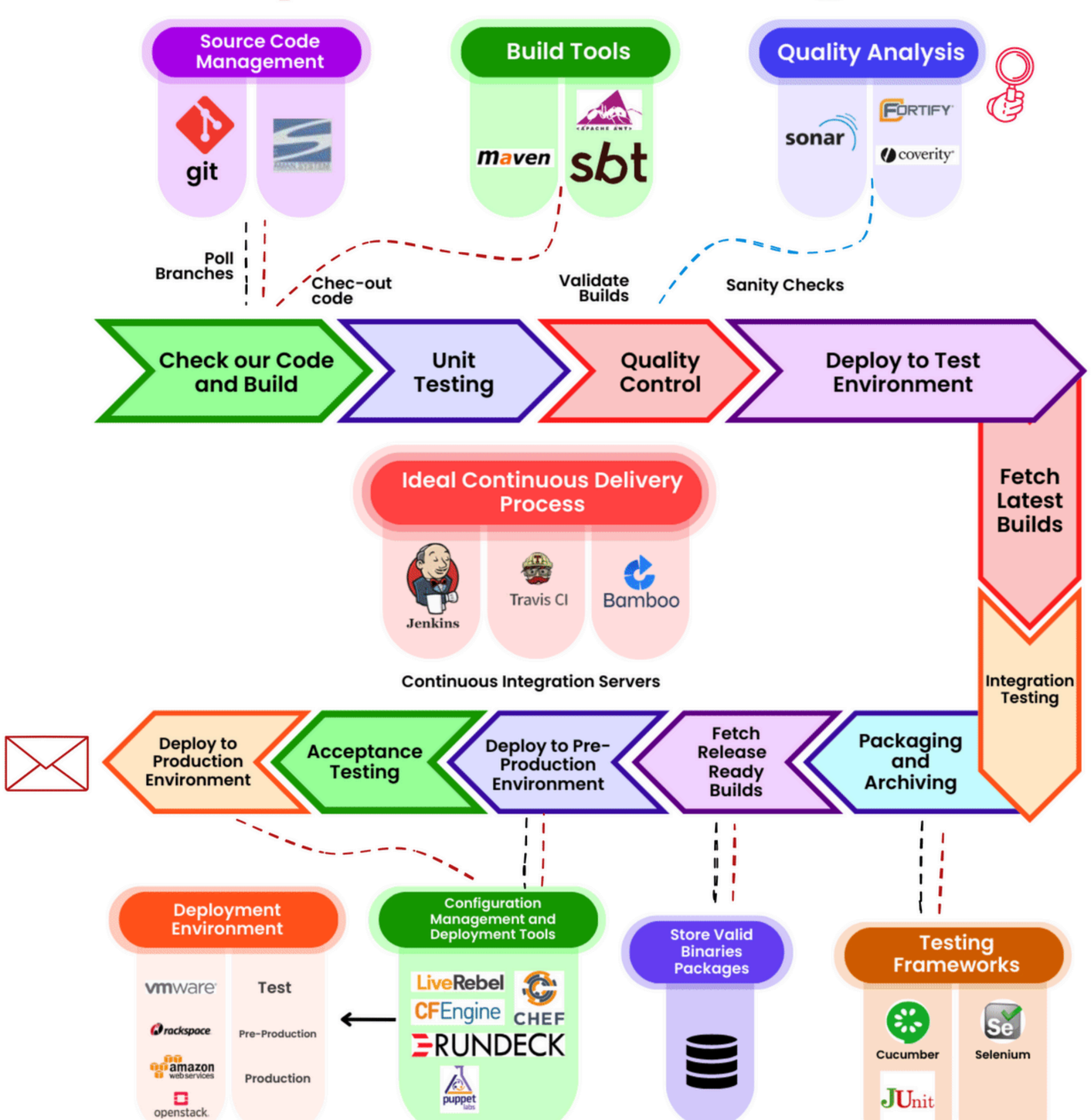
Continuous delivery stops short of automatic production deployment, while **continuous deployment** automatically releases the updates into the production environment.



CI/CD Pipeline

These connected practices are often referred to as a "CI/CD pipeline" and are supported by development and operations teams working together in an agile way within DevOps.





CI/CD Workflow

Benefits of CI/CD Pipeline

- Avoid bugs and code failures while maintaining a continuous cycle of software development and updates.
- Decrease complexity, increase efficiency, and streamline workflows.
- Automates the manual human intervention traditionally needed to get new code from a commit into production, downtime is minimized and code releases happen faster.
- Incorporate users’ feedbacks more frequently and effectively, meaning positive outcomes for end users and more satisfied customers overall.

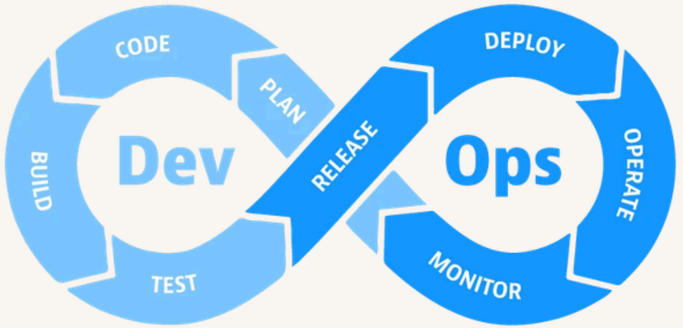
Tools


Best CI/CD Tools
GitHub Actions
CI/CD
Jenkins
argo
Bamboo
circleci
axify
Buddy
Spinnaker
TeamCity
Azure
Concourse

CI/CD vs DevOps

The scope of CI/CD is primarily focused on software development and delivery. It involves practices like version control, code integration, testing, and deployment. CI/CD is a part of the build-and-release process in the software development life cycle.

DevOps has a broader scope. It encompasses the entire software development life cycle, from planning, coding, building, testing, deployment, operation, and monitoring.



References

https://www.redhat.com/en/topics/devops/what-is-ci-cd
https://kodekloud.com/blog/the-ultimate-devops-roadmap-for-2025-a-step-by-step-guide-to-mastering-devops/
https://www.atlassian.com/continuous-delivery/continuous-integration/how-to-get-to-continuous-integration
https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing