DevOps

Atul Phad

Introduction

DevOps is a software development and delivery approach that aims to bridge the gap between development (Dev) and operations (Ops) teams. It fosters a culture of collaboration and communication, enabling faster delivery of high-quality software while maintaining operational stability and efficiency.

Contd..

In traditional software development models, the development and operations teams often worked in silos, resulting in delays and miscommunications. DevOps seeks to break down these barriers, allowing for a more streamlined process from ideation to production.

Core Principles

Collaboration

Automation

Continuous Integration / Continuous Delivery

Feedback and monitoring

Continuous Learning

Collaboration

DevOps encourages open communication and collaboration between development and operations teams, fostering a culture of shared responsibility and accountability.

CI / CD

CI/CD is the practice of automatically building, testing, and deploying software changes to production. This approach ensures a continuous flow of high-quality software releases.

Feedback and monitoring

DevOps emphasizes the importance of monitoring and gathering feedback from both technical and business stakeholders to improve processes and continuously iterate on products

Automation

Automation is a cornerstone of DevOps. It streamlines processes, reduces human error, and allows for faster delivery and improved consistency.

Continuous learning and improvement

A commitment to continuous learning and improvement is essential in a DevOps culture. Teams should regularly evaluate their processes and tools to identify areas for growth and optimization.

Why

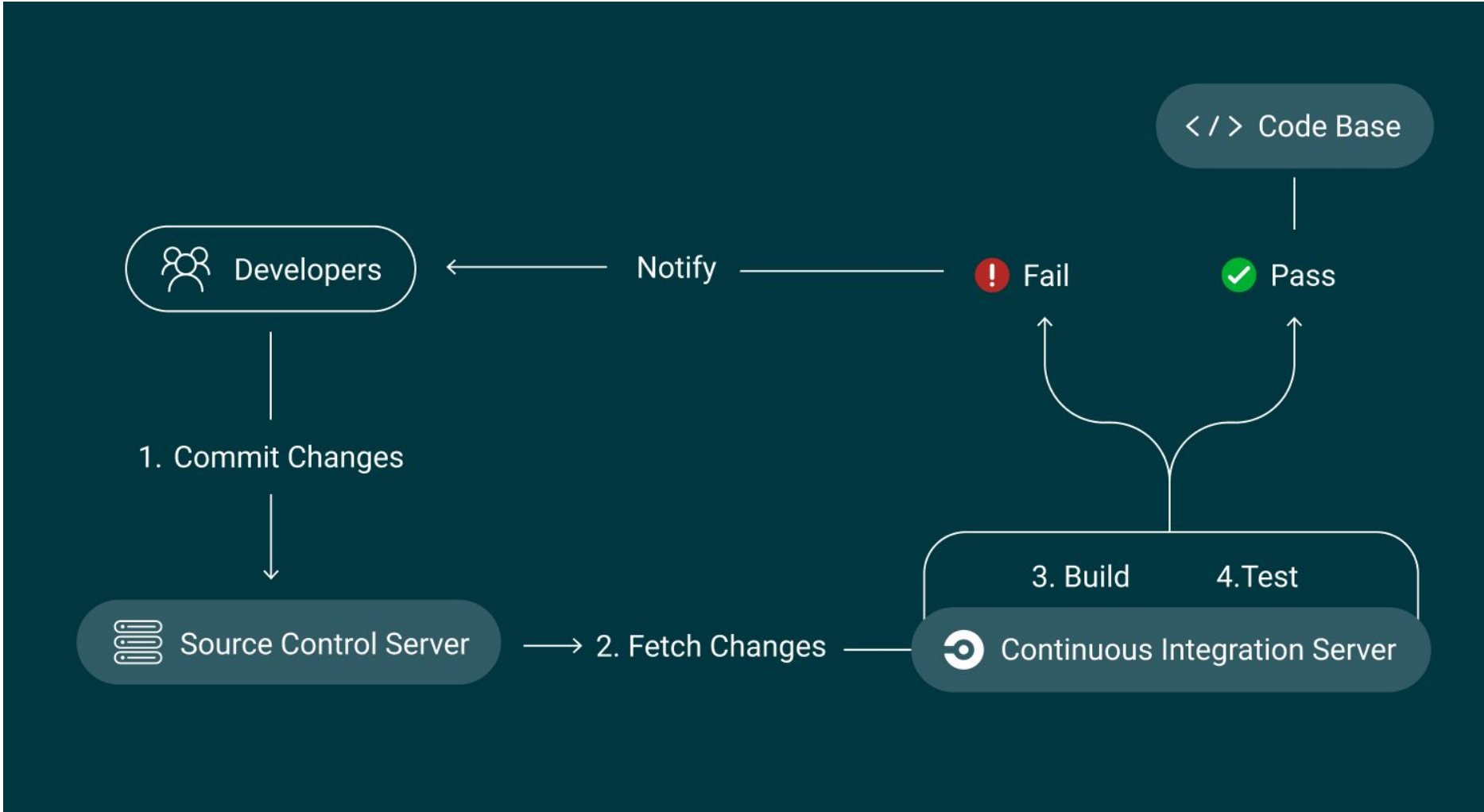
- Faster Delivery and Time-to-Market
- Improved Quality and Reliability
- Enhanced Collaboration and Communication
- Cost Savings
- Increased Customer Satisfaction

Tools

- Version Control Systems – Git, Mercurial, and Subversion
- CI/CD Tools – Jenkins, Travis CI, CircleCI, and GitLab CI/CD
- Configuration Management – Ansible, Chef, and Puppet
- Containerization – Docker, Kubernetes, and OpenShift
- Monitoring and Analytics – Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana), and Datadog
- Cloud Platforms – Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform

Continuous Integration

Continuous integration (CI) is a software development practice in which developers merge their changes to the main branch many times per day. Each merge triggers an automated code build and test sequence, which ideally runs in less than 10 minutes. A successful CI build may lead to further stages of continuous delivery.


Code Base
Developers
Notify
Fail
Pass
1. Commit Changes
3. Build
4.Test
Source Control Server
2. Fetch Changes
Continuous Integration Server

Contd..

Continuous integration is a software development process where developers integrate the new code they've written more frequently throughout the development cycle, adding it to the code base at least once a day. Automated testing is done against each iteration of the build to identify integration issues earlier, when they are easier to fix , which also helps avoid problems at the final merge for the release. Overall, continuous integration helps streamline the build process, resulting in higher-quality software and more predictable delivery schedules.

Continuous Delivery

Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, *safely* and *quickly* in a *sustainable* way.

JENKINS

Jenkins is an open source automation tool written in Java programming language that allows continuous integration.

Jenkins **builds** and **tests** our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

It also allows us to continuously **deliver** our software by integrating with a large number of testing and deployment technologies.

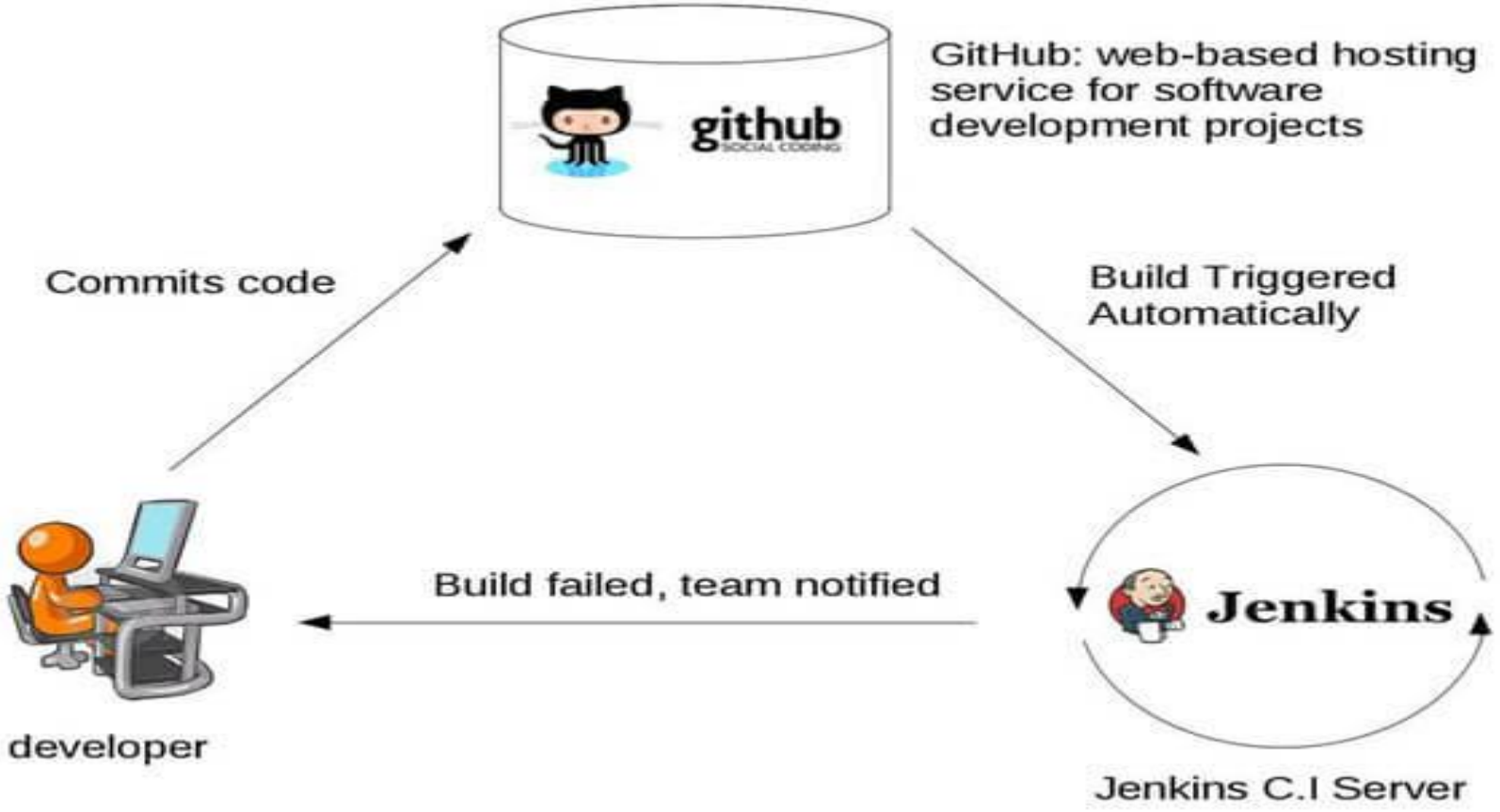
Contd..

Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.

History

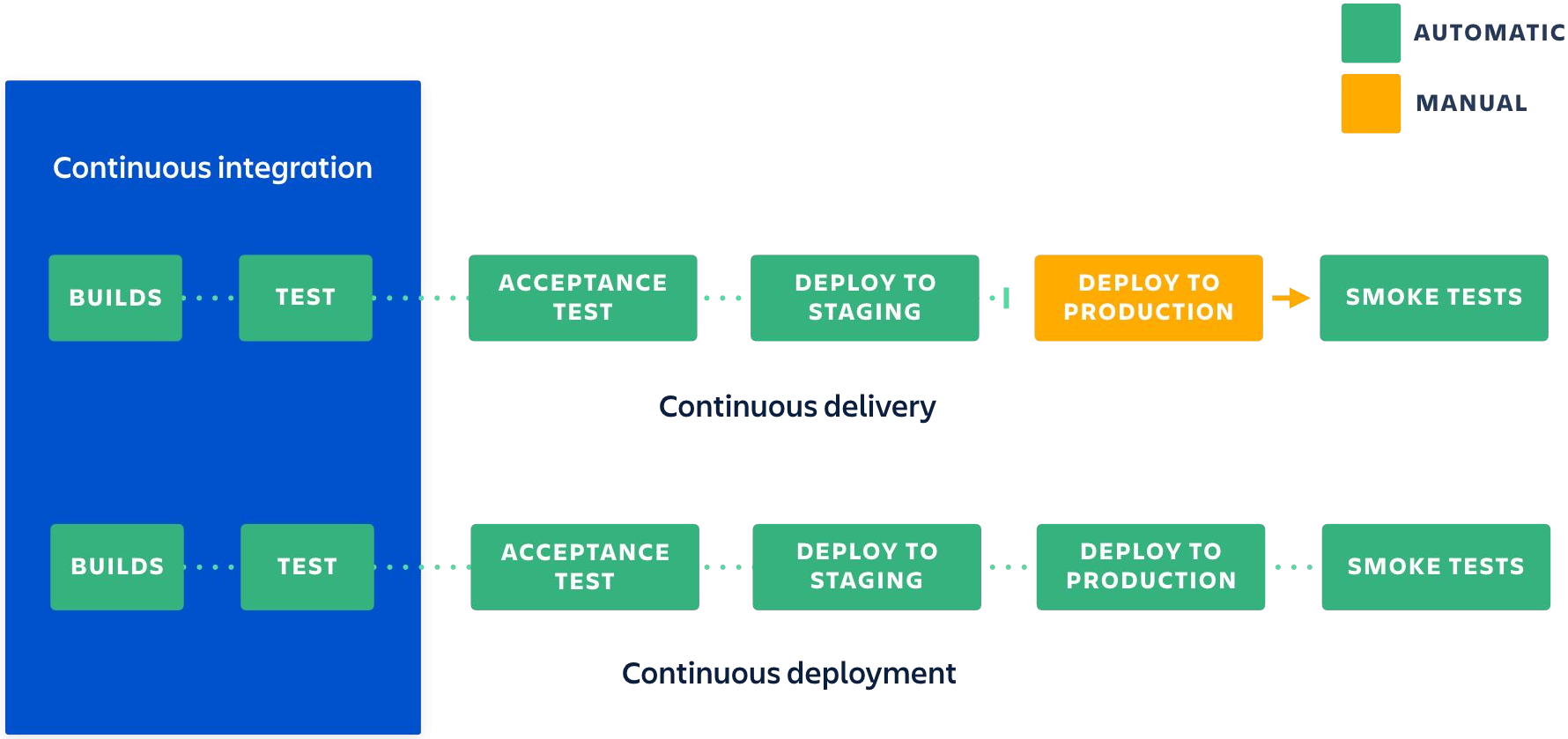
Kohsuke Kawaguchi, who is a Java developer, working at SUN Microsystems, was tired of building the code and fixing errors repetitively. In 2004, he created an automation server called **Hudson** that automates build and test task.

In 2011, Oracle who owned Sun Microsystems had a dispute with Hudson open source community, so they forked Hudson and renamed it as **Jenkins**.


GitHub: web-based hosting service for software development projects
github
Commits code
Build Triggered Automatically
Build failed, team notified
Jenkins
developer
Jenkins C.I Server

Continuous deployment

Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.


AUTOMATIC
MANUAL
Continuous integration
BUILDS
TEST
ACCEPTANCE TEST
DEPLOY TO STAGING
DEPLOY TO PRODUCTION
SMOKE TESTS
Continuous delivery
BUILDS
TEST
ACCEPTANCE TEST
DEPLOY TO STAGING
DEPLOY TO PRODUCTION
SMOKE TESTS
Continuous deployment