DevOps Fundamentals

Ganesh Palnitkar



@AutomationFact

Agenda

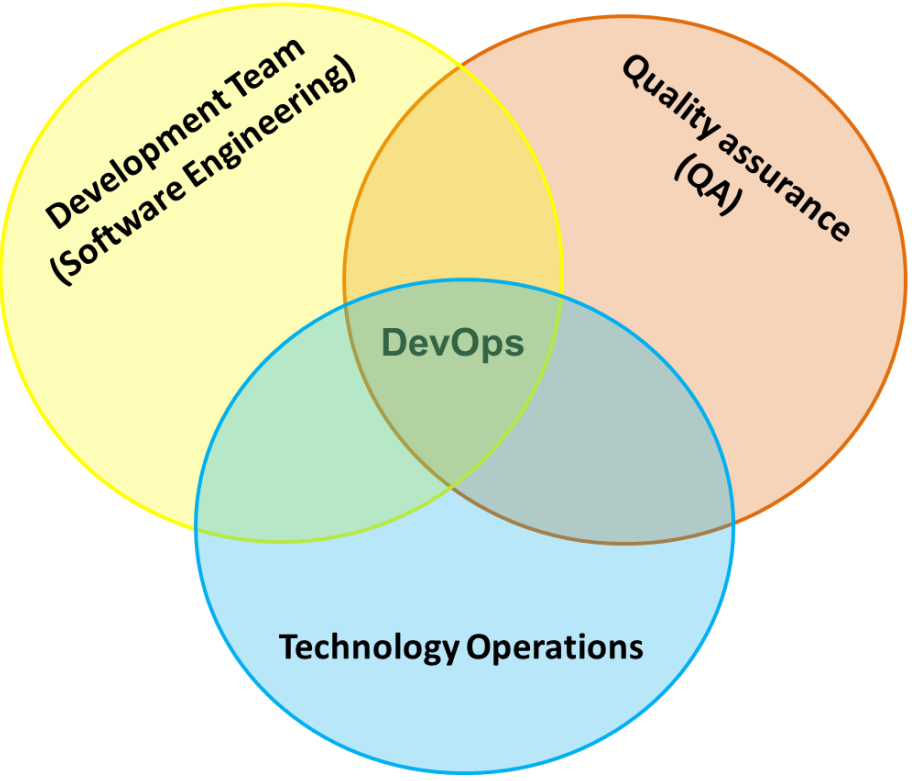
In this session, you will learn about:

- Introduction to DevOps
- Relationship between DevOps and Agile
- Best Practices – DevOps
- Maven and Gradle (Dependency Management)
- Source Control Repository (Git & Bit Bucket)
- Continuous Integration (Using Jenkins)

What’s DevOps

What’s DevOps?

- DevOps is a software development and delivery process that emphasizes communication and collaboration between Product Development Team, QA Team, Operations Team and Business owners to increase organization's ability to deliver application and services at high velocity.
- This speed enables organizations to better serve their customers and compete more effectively in the market.



What's DevOps?

DevOps building blocks:

- Code
- Build
- Test

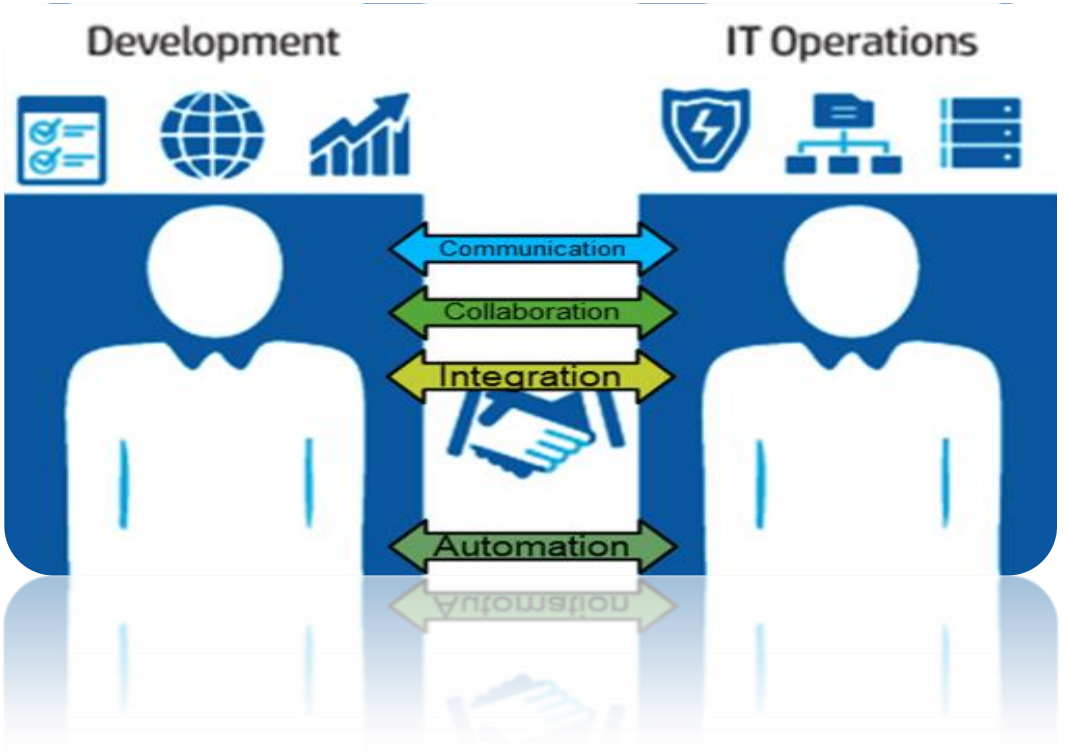
Development and QA Team

- Packaging
- Release management
- Configuration management
- Application and Infrastructure monitoring

Operations Team

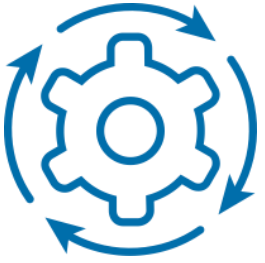
DevOps Culture

- DevOps is more than just a tool or a process change; it inherently requires an organizational culture shift.
- This cultural change is especially difficult, because of the conflicting nature of departmental roles:



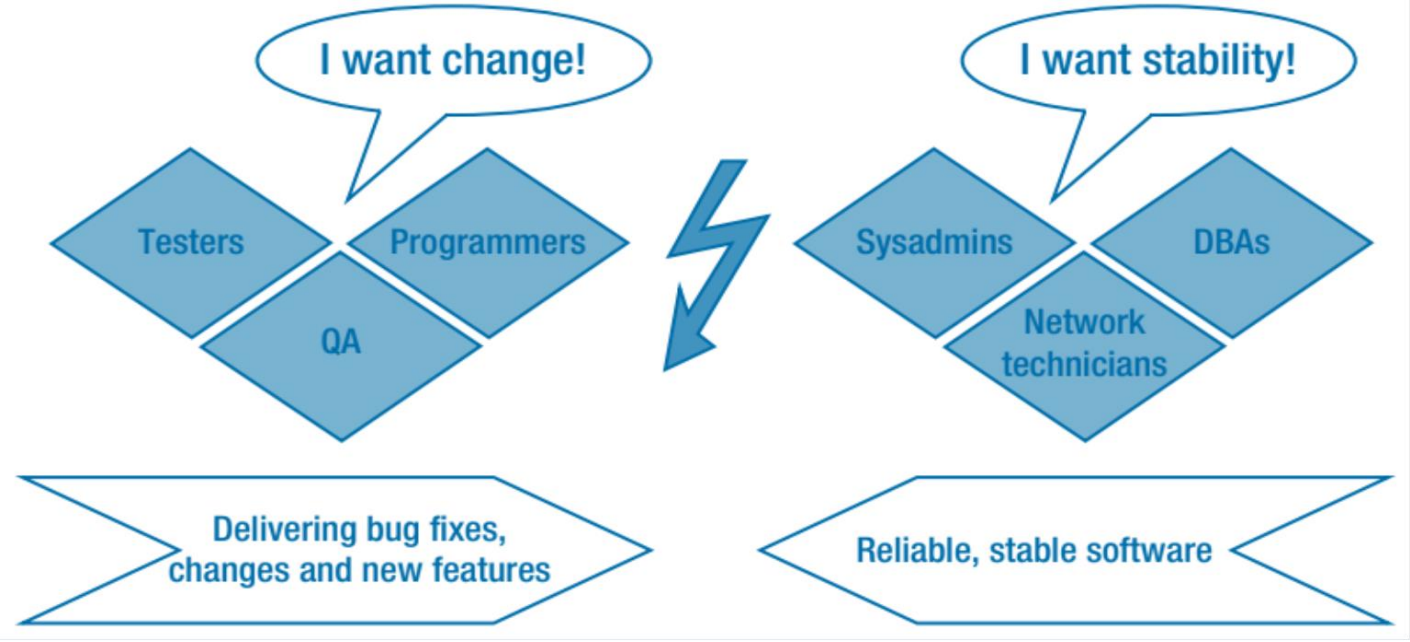
DevOps Culture

Operations	Developers	Testers
Seeks organizational stability	Seek change	Seek risk reduction



Development v/s Operations

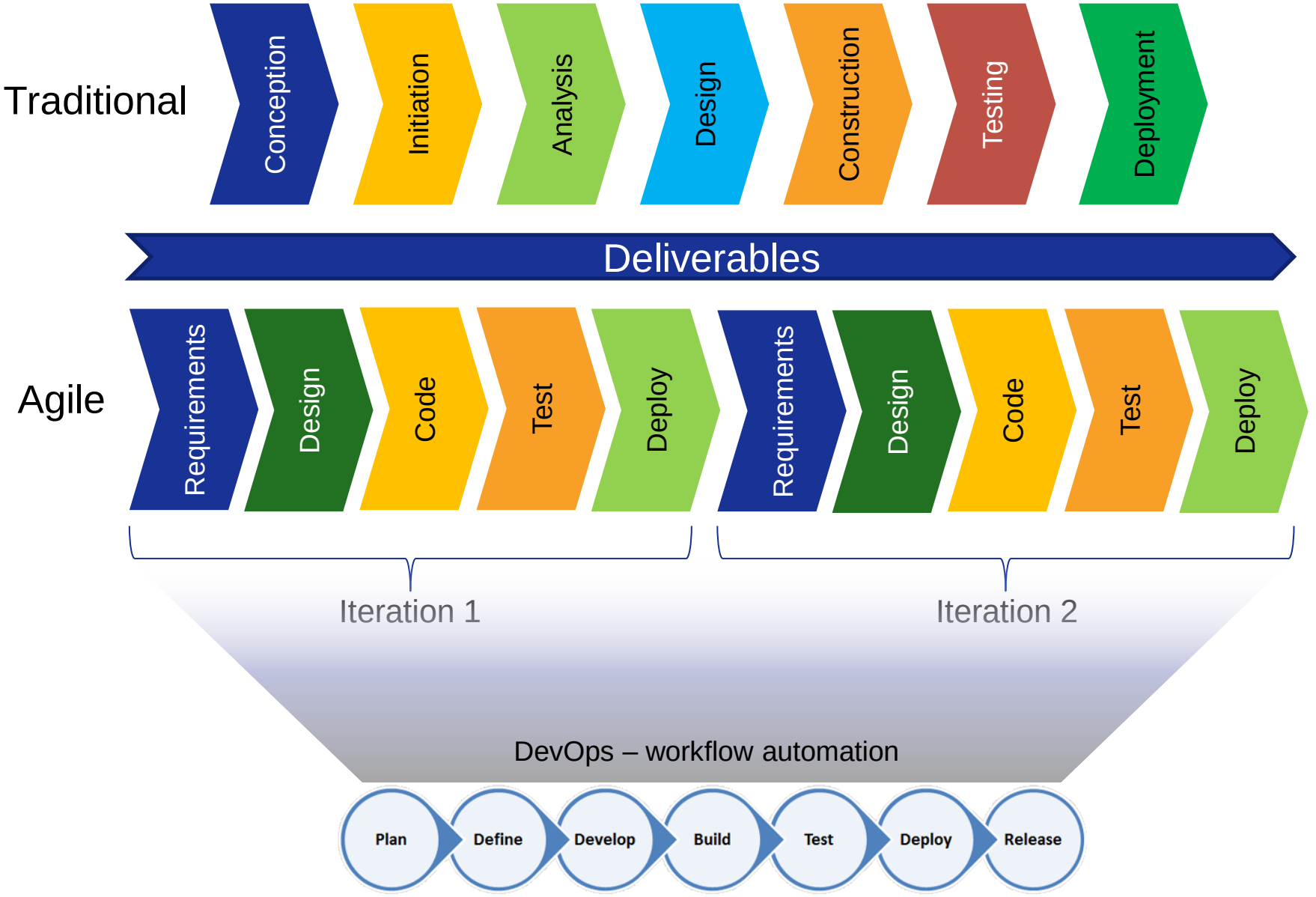
Developers want change



Deliver bug-fixes, changes and new features.

- The main task of the development team is to fulfil the customer's requirements, test the solution, and provide software updates in quick succession. New features that have been implemented and tested by the developers add potential value for the customer.
- On one hand, the development team wants change. On the other hand, the operations team is mainly interested in reliable and stable software environment. Every change forwarded by the development team can endanger the existing reliability and stability of production environment.

Development Methodologies - Comparison



DevOps in Business

DevOps Drivers

- **Market Competition**
- **Ever-changing Business Needs**
- **Quick to Market Requirement**
  - Tight delivery deadlines
  - “The code works on my machine” – blame game
  - Disconnect bet’n Development and Operations team.
- **Conflict Scenarios**
  - Conflict during deployment
  - Conflict after deployment
  - Conflict about performance

DevOps in Business

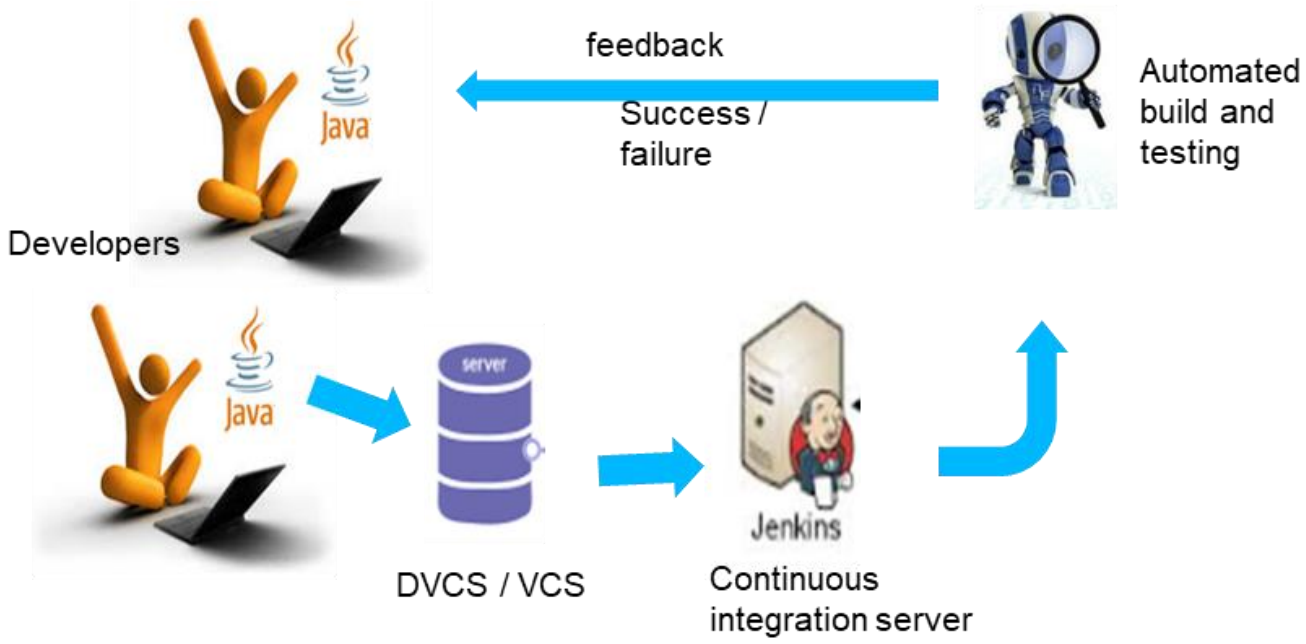
DevOps Drivers

- Advantages of agile processes like Scrum, Kanban are often nullified because of the obstacles to collaboration, processes, and tools that are build up in front of operations.
- Thus, achieving delivery timelines for a sprint becomes challenging.

DevOps for QA

Continuous Testing enabled with, continuous integration.

Continuous integration is software development practice in which **team members integrate their work frequently**, leading multiple integrations per day.



WHAT PURPOSE DOES THIS SERVE?

1. Each integration helps to reveals integrations errors in build success / failures as quickly as possible.

2. Helps in significantly reducing integration problems and delivery timeline.

DevOps for IT Operations

Integrated Environment Provisioning

- Dynamic Environment Provisioning
- Containerized App Deployment and Data Center Management

Continuous Application Deployment

- Single Click Deployment

Continuous Monitoring

- Performance Monitoring
- System and Application Monitoring
- Log Analysis

DevOps for Business Owners

Quick to Market

- Agility

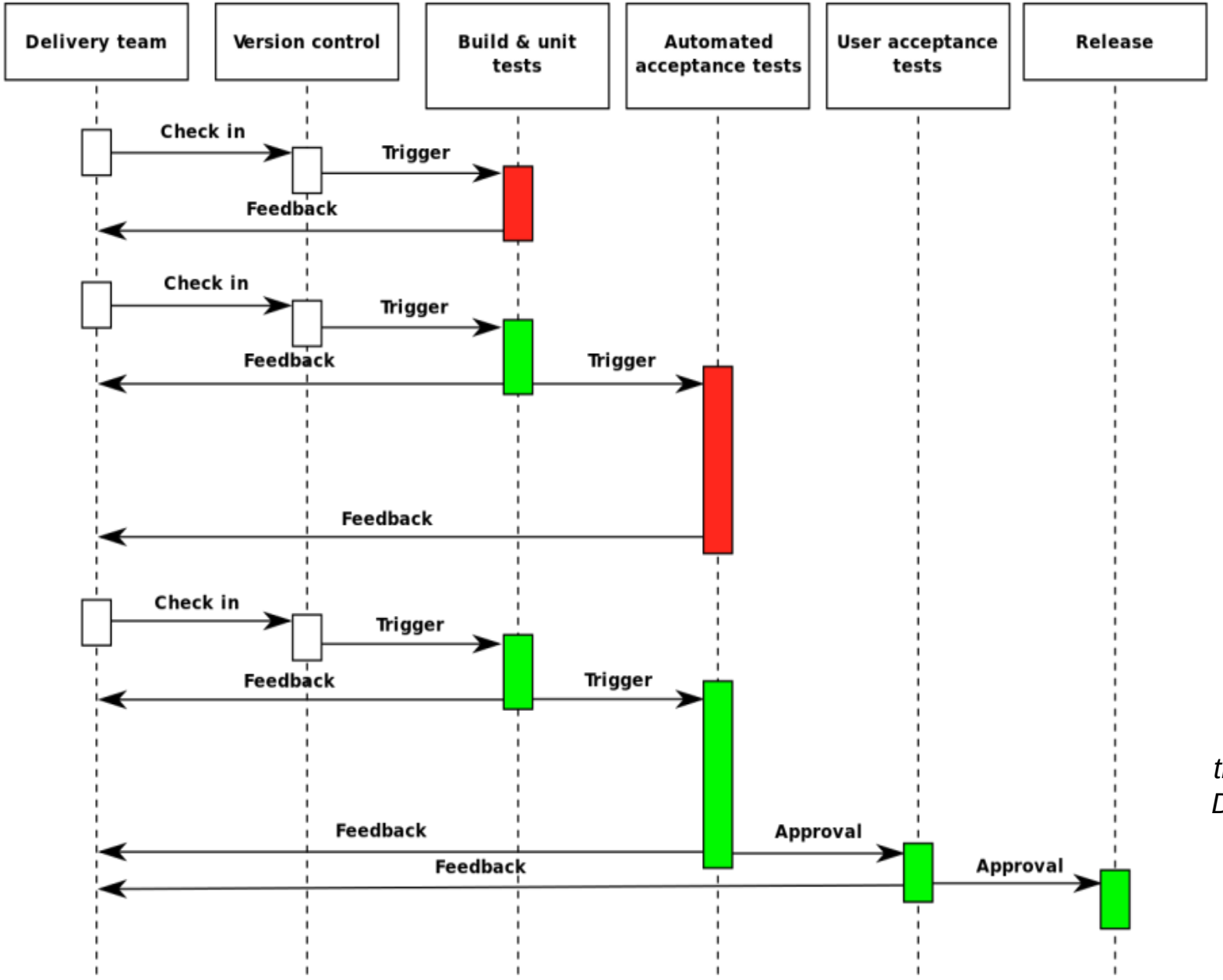
Environment stability

- Fast recovery
- Fully automated deployments

Customer satisfaction

- Improvement in product quality
- Quick turn around time

Continuous Delivery Pipeline



Source: *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*

Challenges in Implementing DevOps!

Establishing DevOps Culture

Implementing Change in Application Development Environment.

Environment Upgradation (standardization)

Application Complexity

Budget

Availability of Skillset

Best Practices in DevOps

Active partnership and close coordination among the stake holders in establishing DevOps culture.

Implement DevOps in totality. Avoid partial implementation, can become a reason for failure.

Choose right tool for each phase in DevOps implementation.

Best Practices

Options of substituting a exiting tools should be taken solicitously. No Fancy ideas.

Give **equal importance** to log analysis, report generation and circulation.

Mindset to adapt to changes.

DevOps Operational Benefits

Increased Agility

To enable instant change deployment

Improved Quality

To improve end user satisfaction

Improve Innovation

To increase innovation cycle

Reduced Outages

Less outages in production (about 80% outages are change related)

Maven and Gradle

Maven as App Build Automation

Apache Maven is a dependency management and a build automation tool, primarily used for Java applications.

Conventional Attributes

1. Maven continues to use XML files just like Ant but in a much more manageable way.
2. Maven follows the rule of convention over configuration.

Advantages of Maven

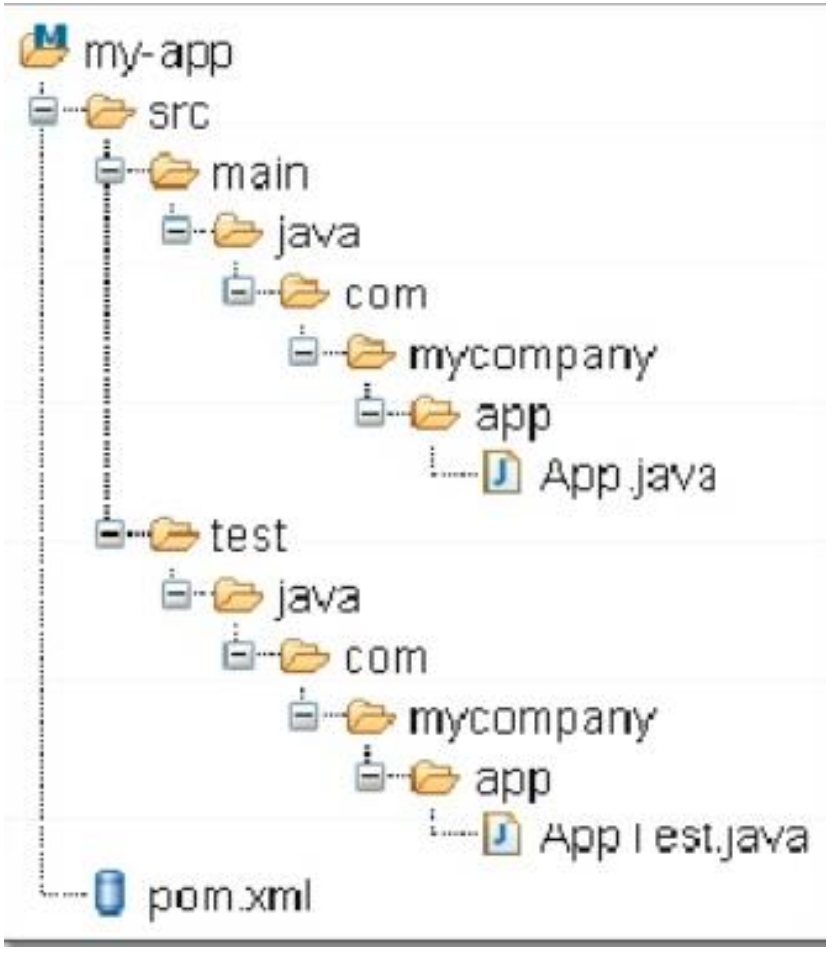
Maven allows us to **focus on what our build should do**, and gives us the framework to do it.

Maven provides **built-in support for dependency management**.

Maven's configuration file, containing build and dependency management instructions, is by convention called ***pom.xml***.

Maven also **prescribes strict project structure**, while **Ant provides flexibility** there as well.

Maven Project Folder structure



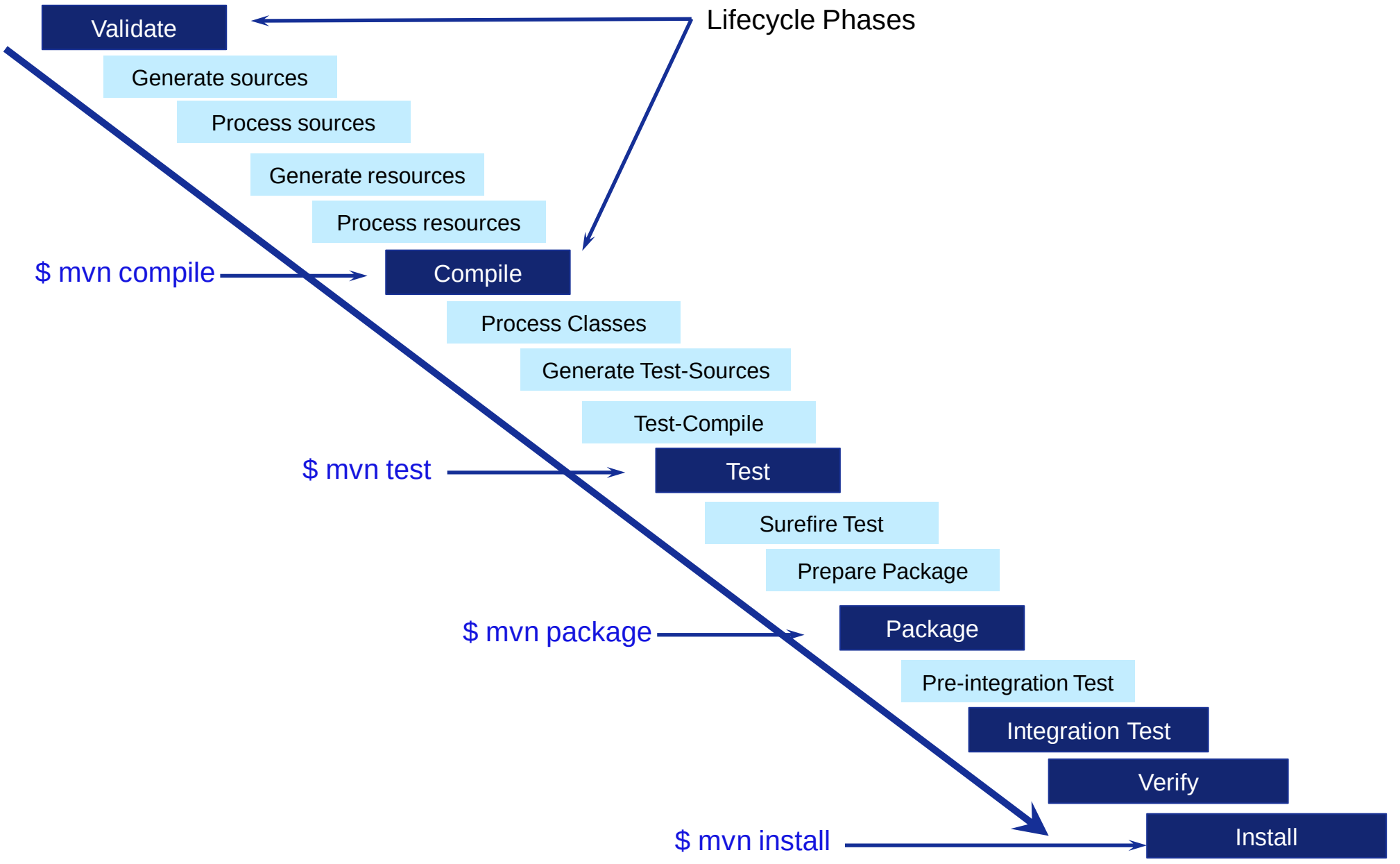
Every valid Maven project folder structure should comply to the requirement of conventions over description

Sample POM.xml File

Here's an example of a pom.xml file for the same simple Java project with the HelloWorld main class from before:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
      http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>baeldung</groupId>
    <artifactId>mavenExample</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <description>Maven example</description>
     <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Build Lifecycle



Gradle as Build Automation

Gradle is a dependency management and a build automation tool which was built upon the concepts of Ant and Maven.

This led to smaller configuration files with less clutter since the language was specifically designed to solve specific domain problems.

- Gradle's configuration file is by convention called *build.gradle.*
- Gradle gave its build steps name **"tasks".**

Sample build.gradle File

Here is an example of a build.gradle file for the same simple Java project with the HelloWorld main class from before:

```
apply plugin: 'java'
 repositories {
    mavenCentral()
}
 jar {
    baseName = 'gradleExample'
    version = '0.0.1-SNAPSHOT'
}
 dependencies {
    compile 'junit:junit:4.12'
}
```

Thank you!