

# DevOps Certification Training

Lesson 03: Continuous Integration, Continuous Deployment, and Build Tools



# Learning Objectives

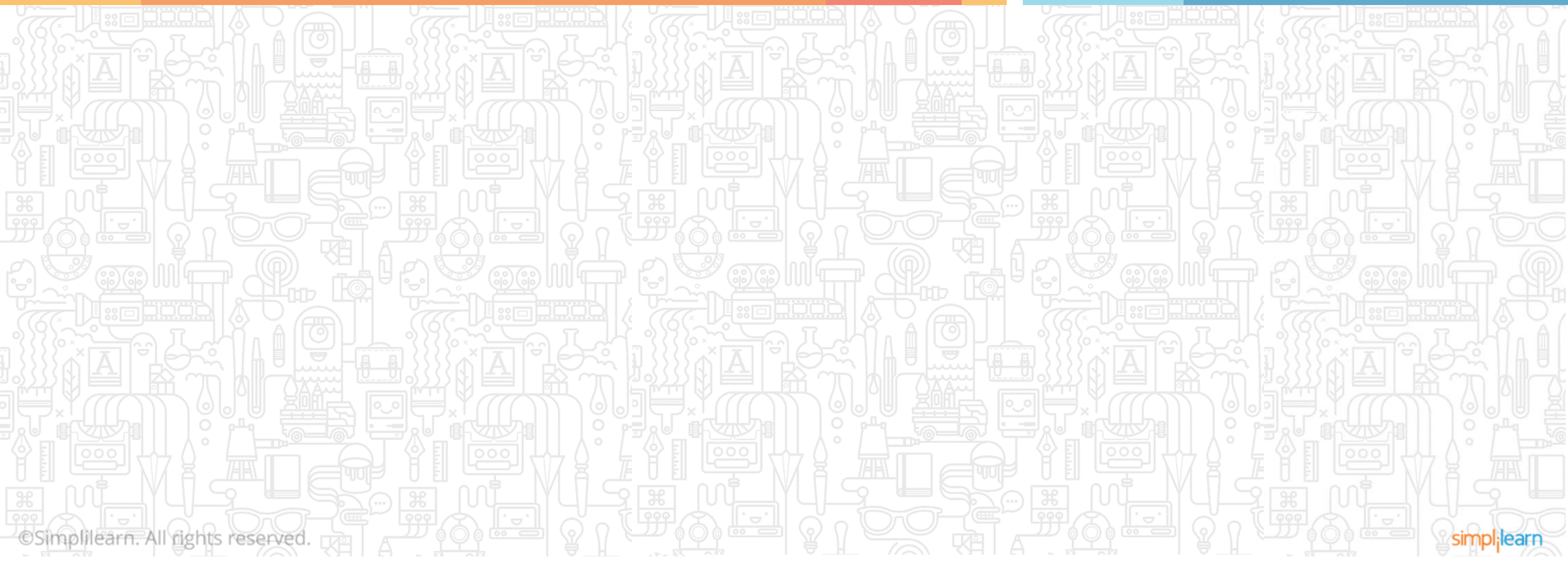
By the end of this lesson, you will be able to:

- ✓ Describe the importance of continuous integration and continuous deployment
- ✓ List the features of Jenkins and demonstrate their uses
- ✓ List the features of TeamCity and demonstrate their uses
- ✓ Select a suitable build tool for your organization



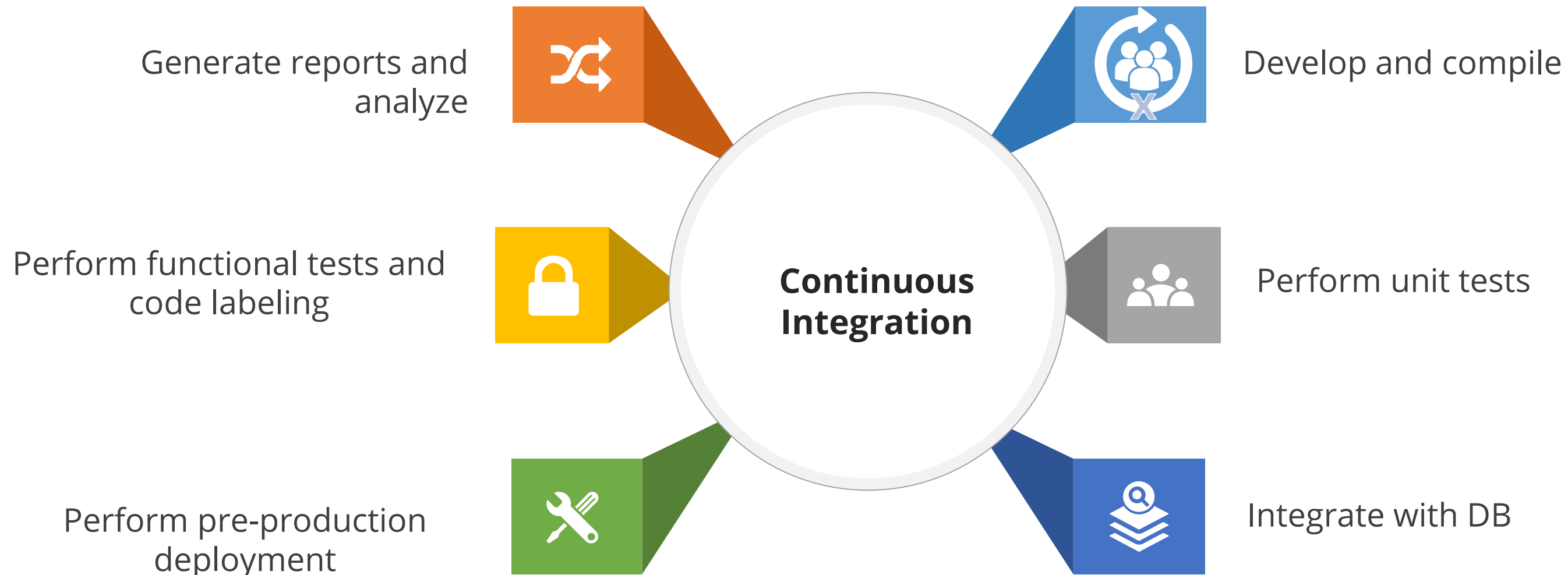
# Continuous Integration, Continuous Deployment, and Build Tools

Overview and Importance of Continuous Integration and Continuous Deployment



# Overview of Continuous Integration

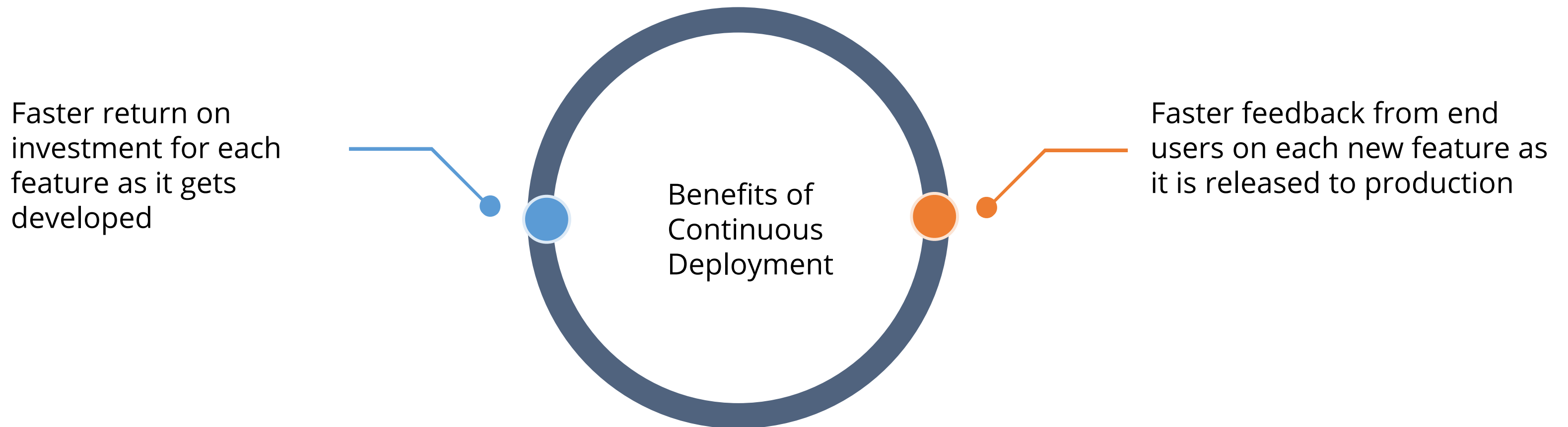
Continuous integration is a development practice of code integration into a shared repository. Each integration is verified by an automated build and automated tests.



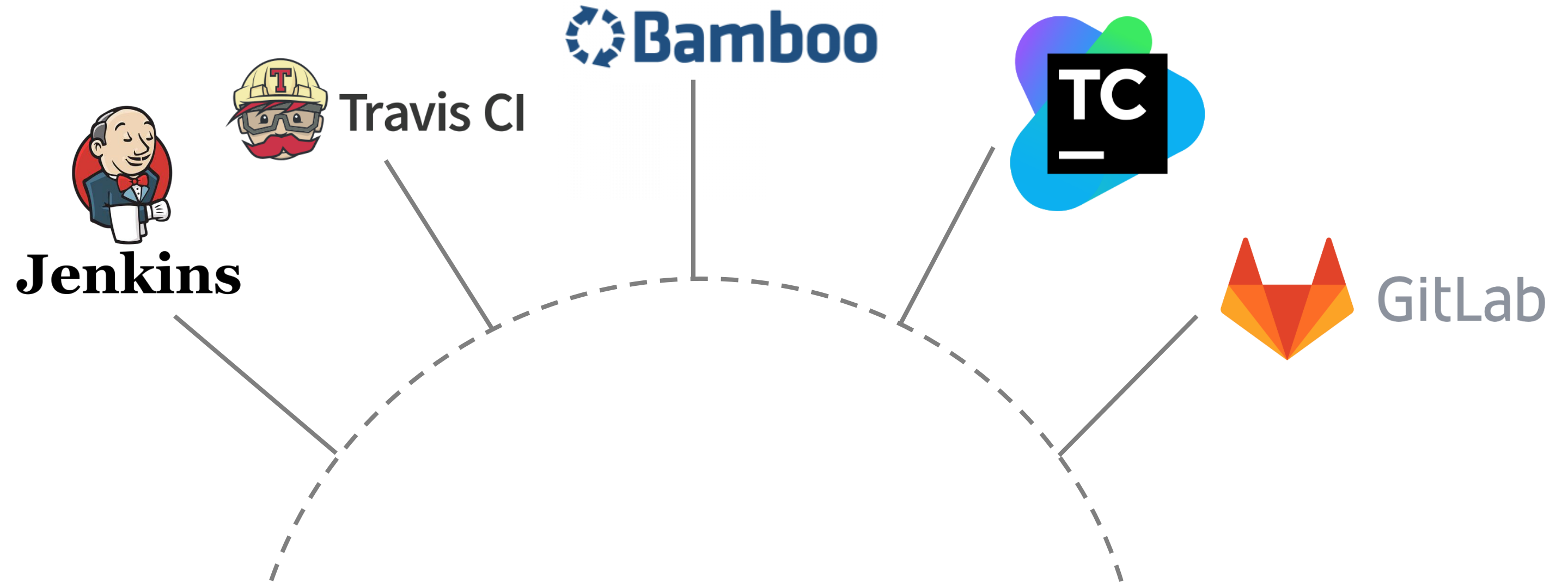
# Overview of Continuous Deployment

---

Continuous Deployment is an extension of continuous integration. It targets to reduce the time between development team writing one new line of code and using it in production.



# Popular Tools in Continuous Integration and Continuous Deployment

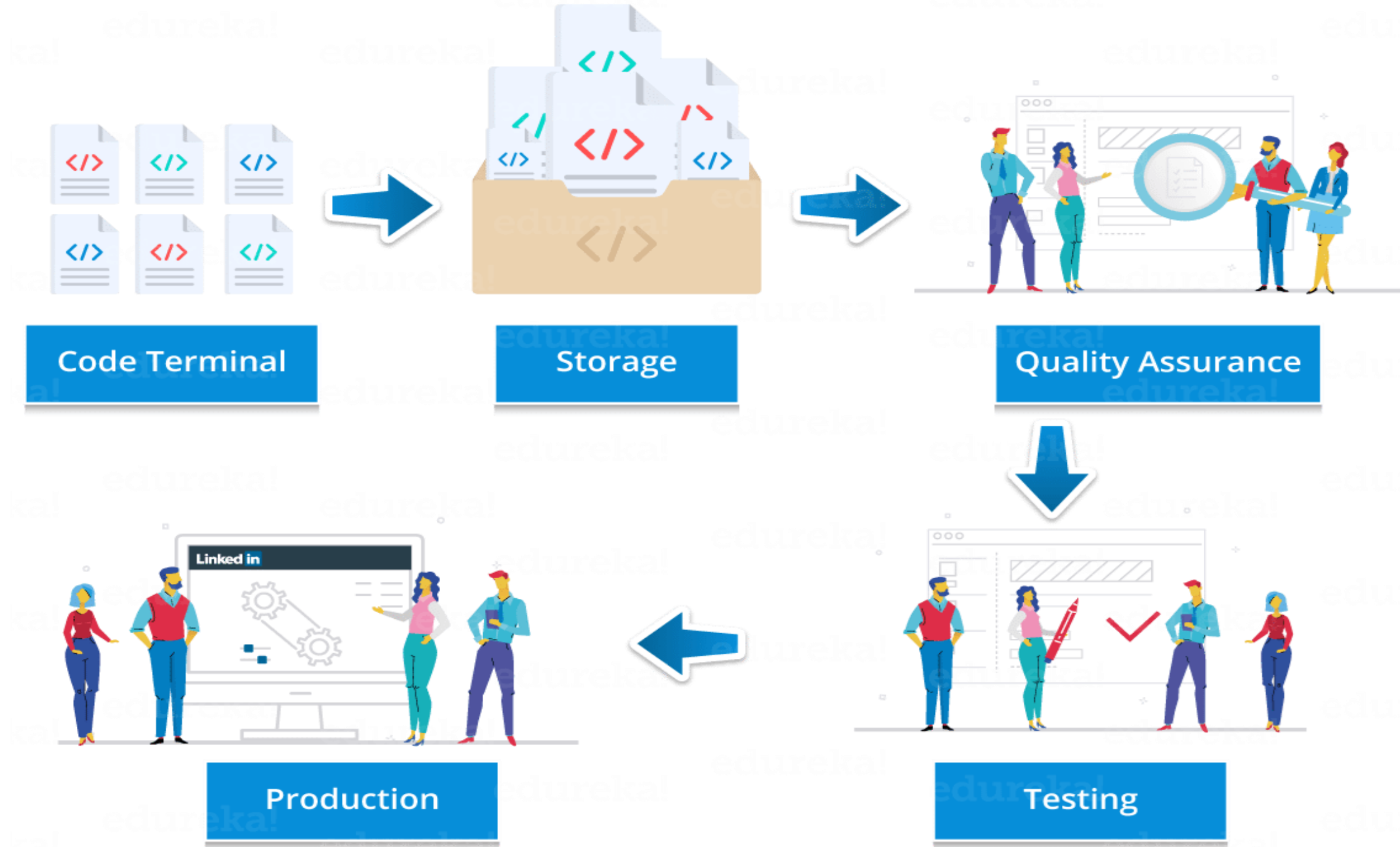




# Continuous Integration with Jenkins



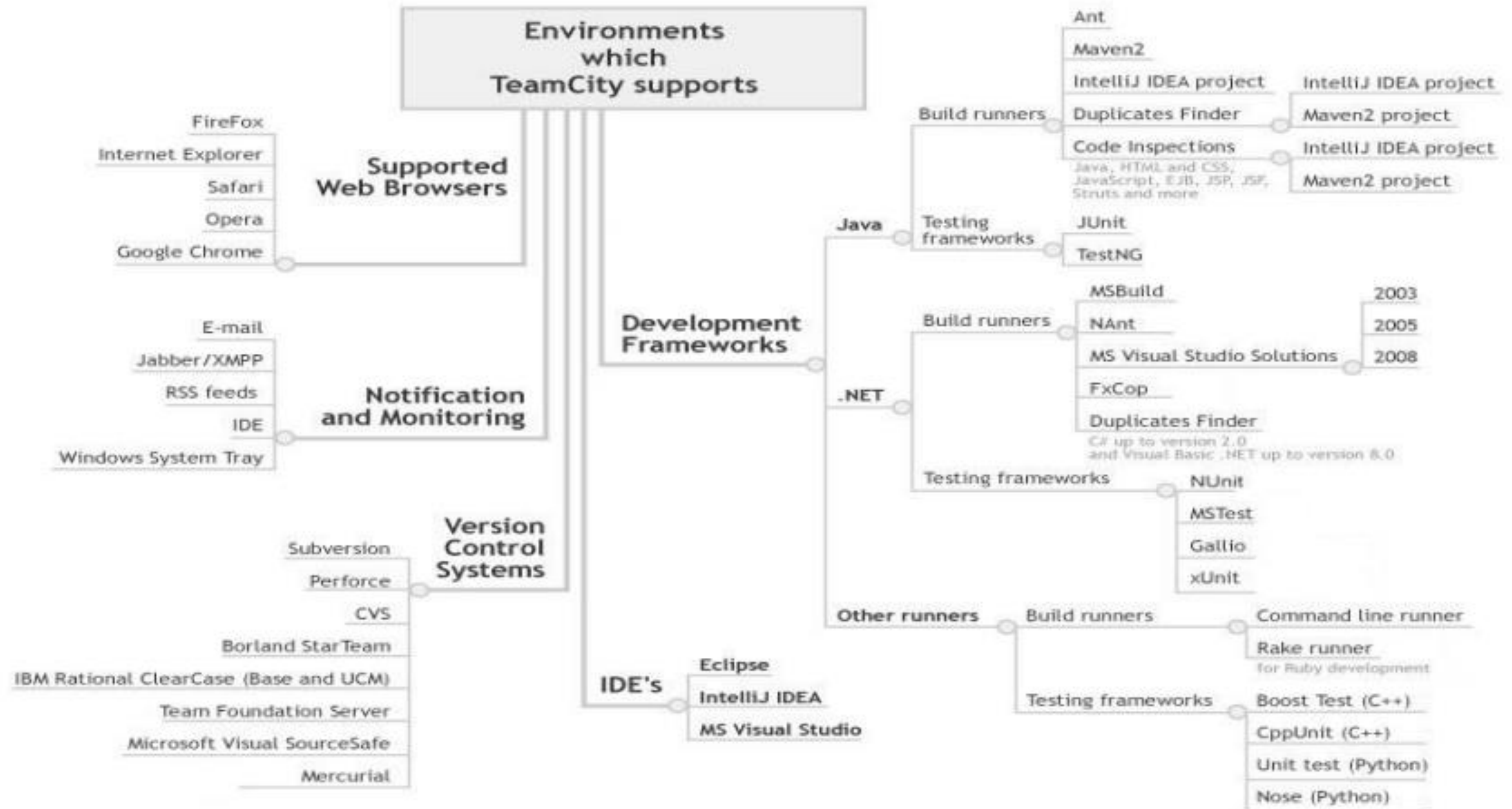
# Continuous Deployment with Jenkins



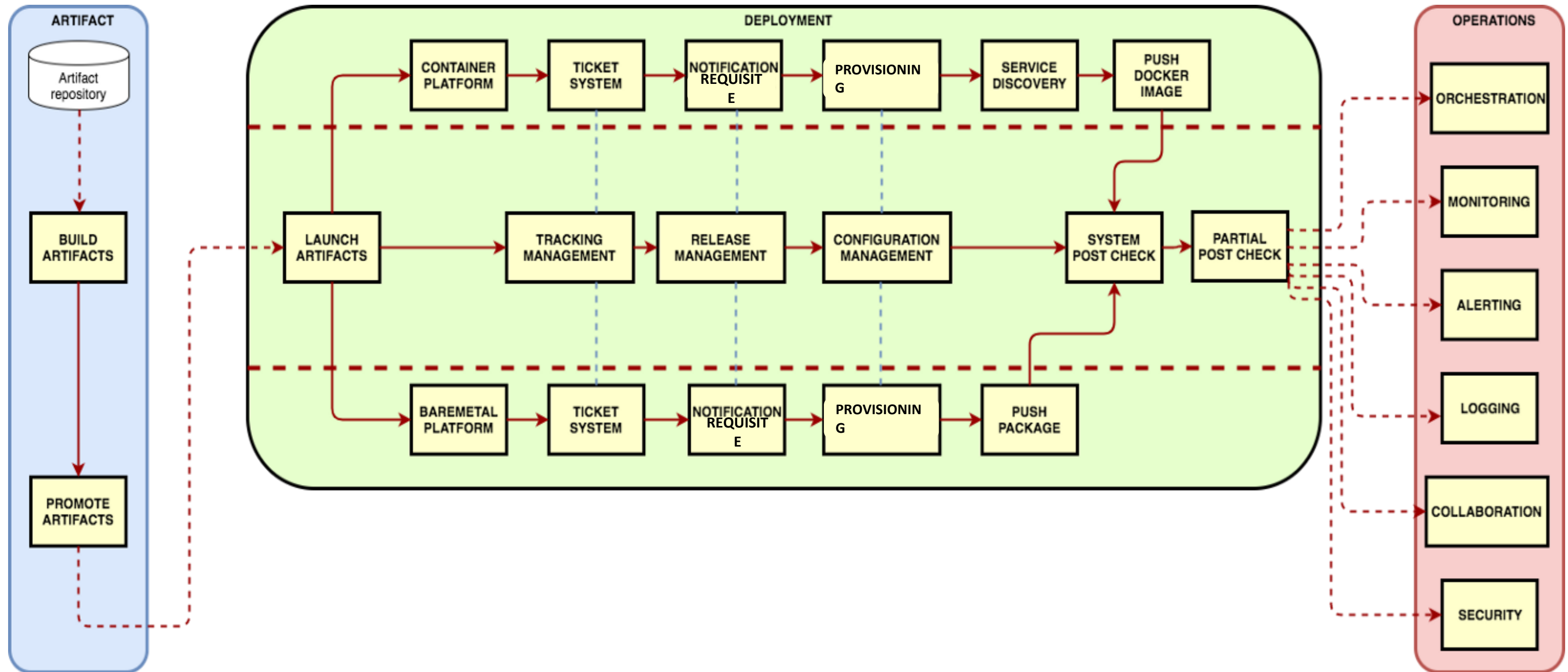
A case study by  
LinkedIn



# Continuous Integration with TeamCity

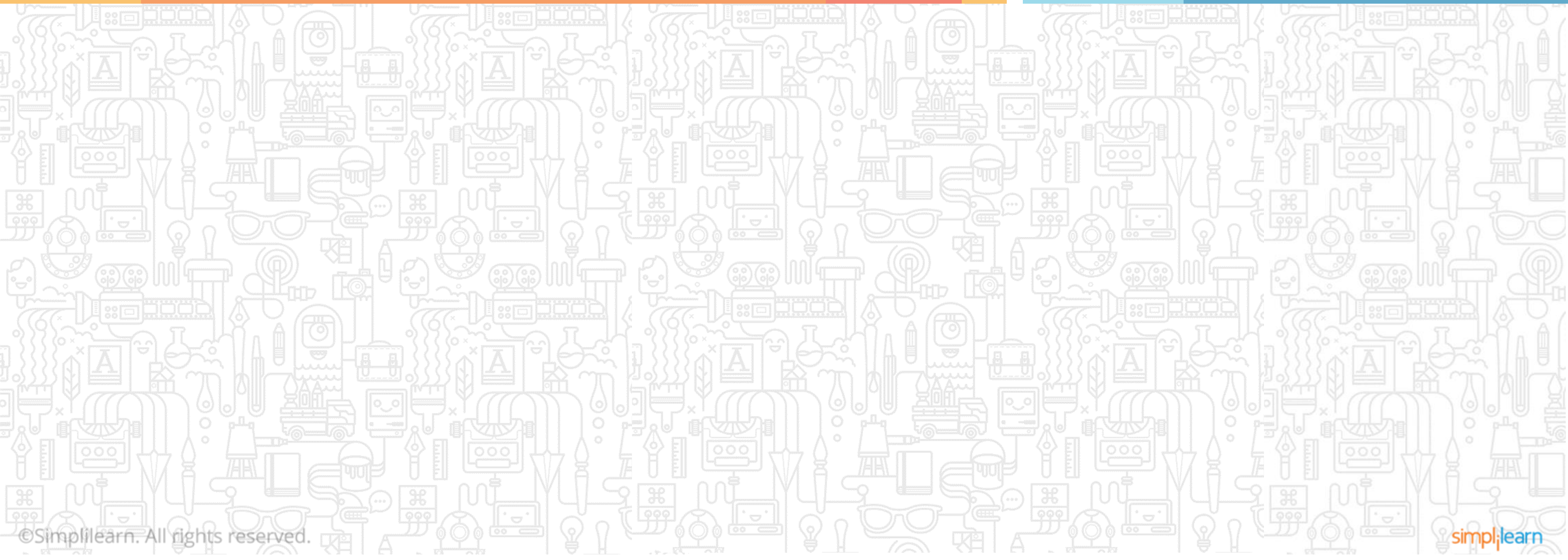


# Continuous Deployment with TeamCity



# Continuous Integration, Continuous Deployment, and Build Tools

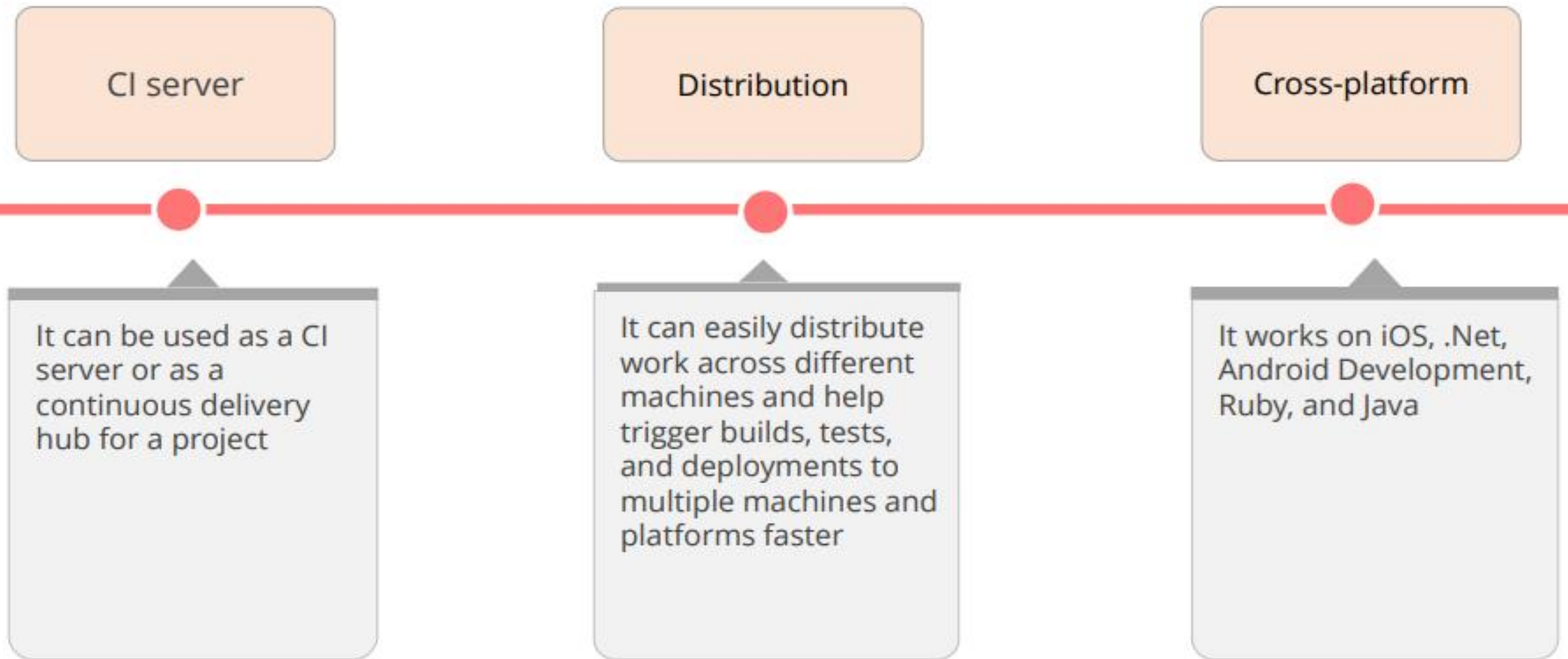
## Overview and Features of Jenkins





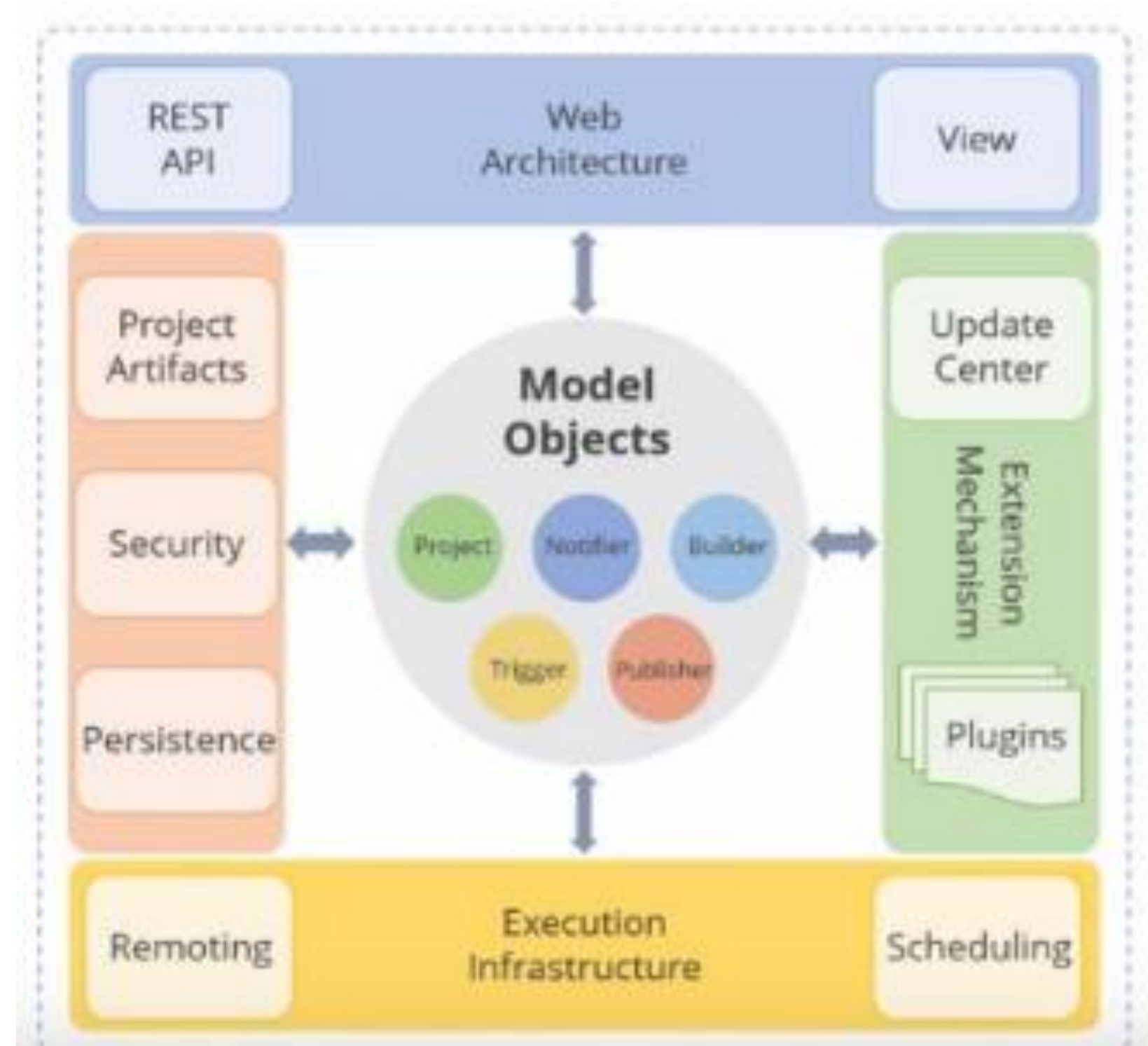
# Jenkins as a Continuous Integration Tool

Jenkins is a Java-based, open source automation tool. It functions as a server and is a software development and cross-platform tool used for continuous integration and continuous deployment.



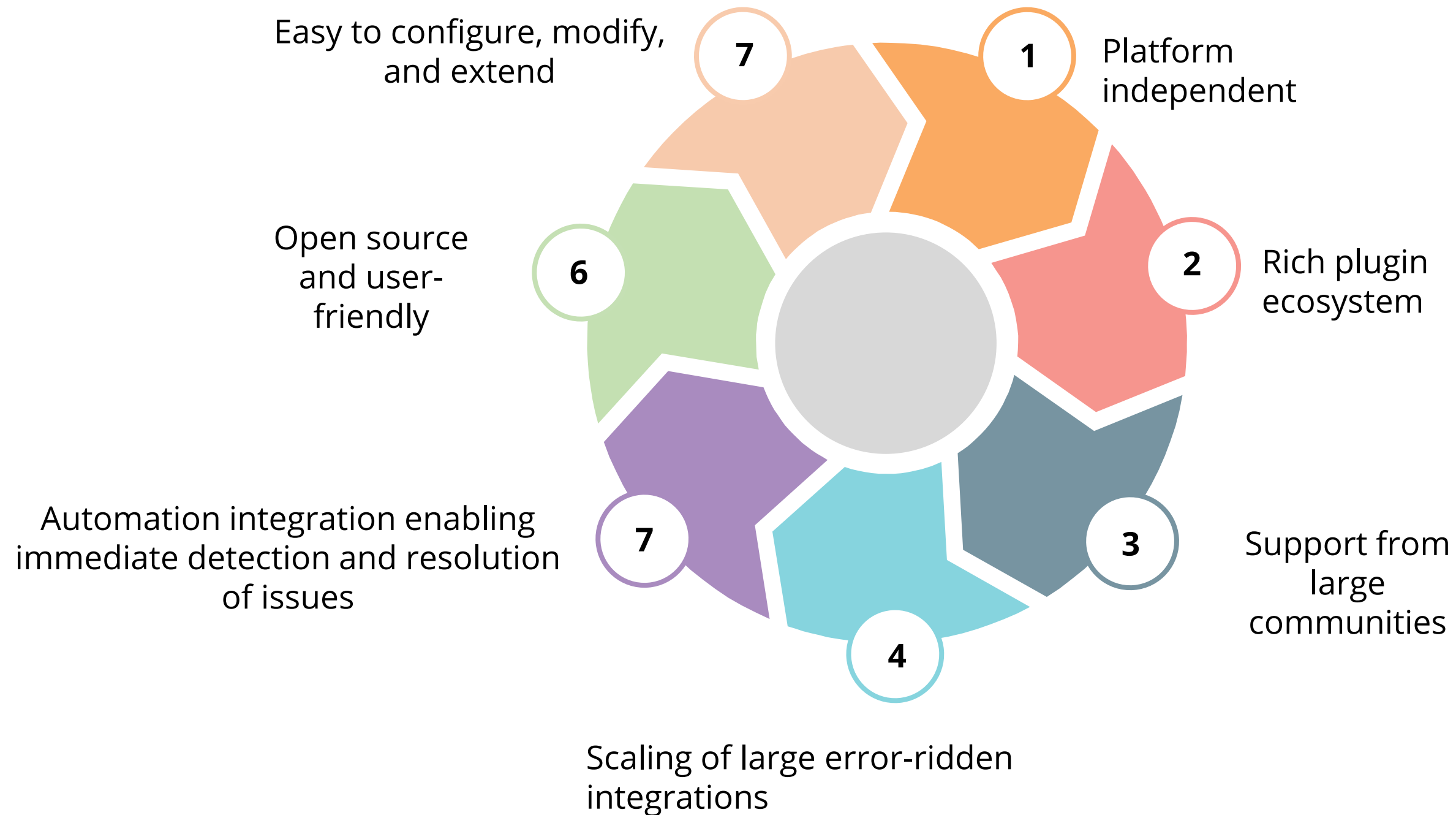
# Architecture of Jenkins

- Jenkins has classes like project and build.
- It uses Jelly as the view technology.
- It uses file system to store its data. Directories are created inside \$JENKINS\_HOME.
- It supports plugins, which can plug into those extension points and extend the capabilities of Jenkins.





# Popular Features of Jenkins



# Build Status and Job Health







Status of the build	Description
	Failed
	Unstable
	Success
	Pending
	Disabled
	Aborted

Figure a: Build status






Job health	Description
	No recent builds failed
	20-40% of recent builds failed
	40-60% of recent builds failed
	60-80% of recent builds failed
	All recent builds failed
	Unknown status

Figure b: Weather reports

# Assisted Practice

Duration: 30 mins

## Set up Jenkins

**Problem Statement:** You are given a project to install and configure Jenkins on your Ubuntu operating system.

**Access:** Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

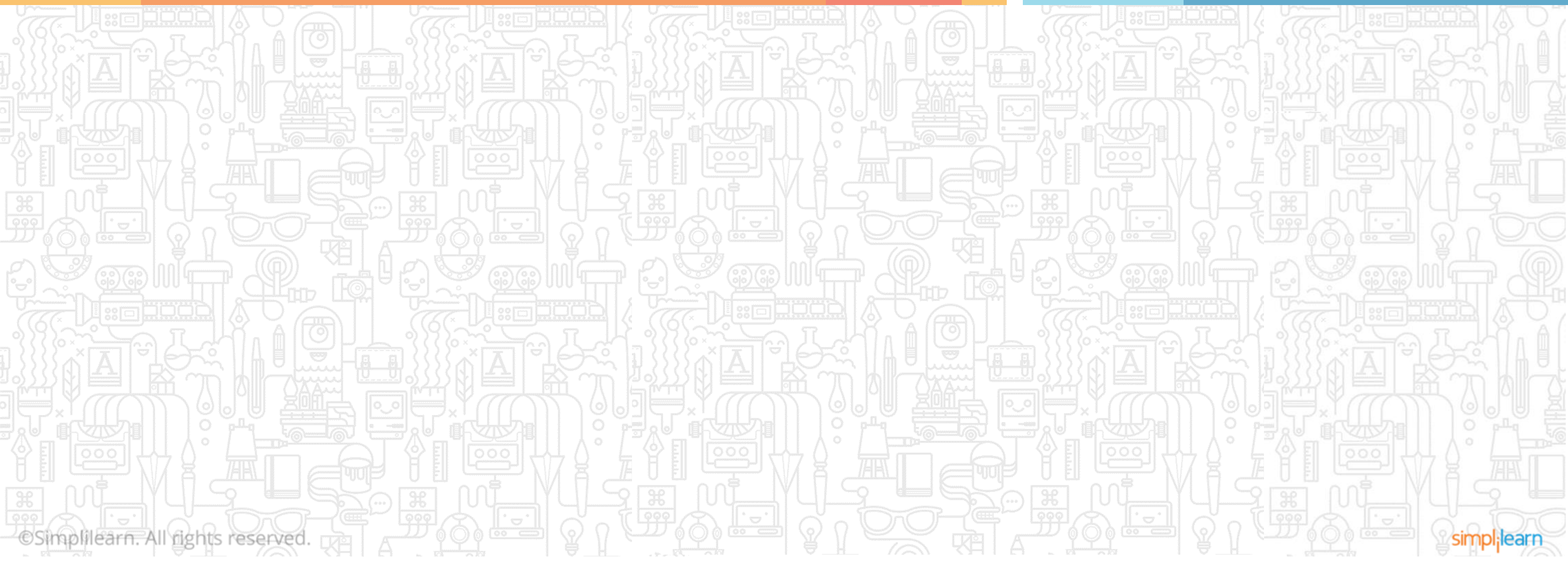
# Assisted Practice: Guidelines to Install and Configure Jenkins

---

1. Login to your Ubuntu lab provided with the course.
2. Open the terminal and execute the command available in the lab document 3.1 to add the key to the system.
3. Edit the sources.list file, add the command to the file, and save it.
4. Update the apt-get package.
5. Install JDK 8+ version.
6. Install Jenkins via apt-get package.
7. Navigate to x.x.x.x:8080 in the browser of your virtual Machine.
8. Get the password and enter it in the Jenkins window.
9. Create a new role/job in Jenkins.
10. Explore the free style project section and build section.

# Continuous Integration, Continuous Deployment, and Build Tools

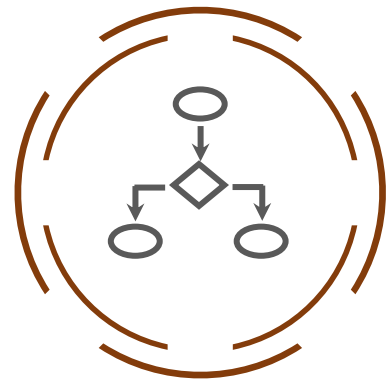
## Overview and the Features of TeamCity





# TeamCity as a Continuous Integration Tool

TeamCity is a Java-based, management and continuous integration server. It is a licensed commercial software used for continuous integration and continuous deployment.



Gated  
commits



Build grid



Integrated  
code

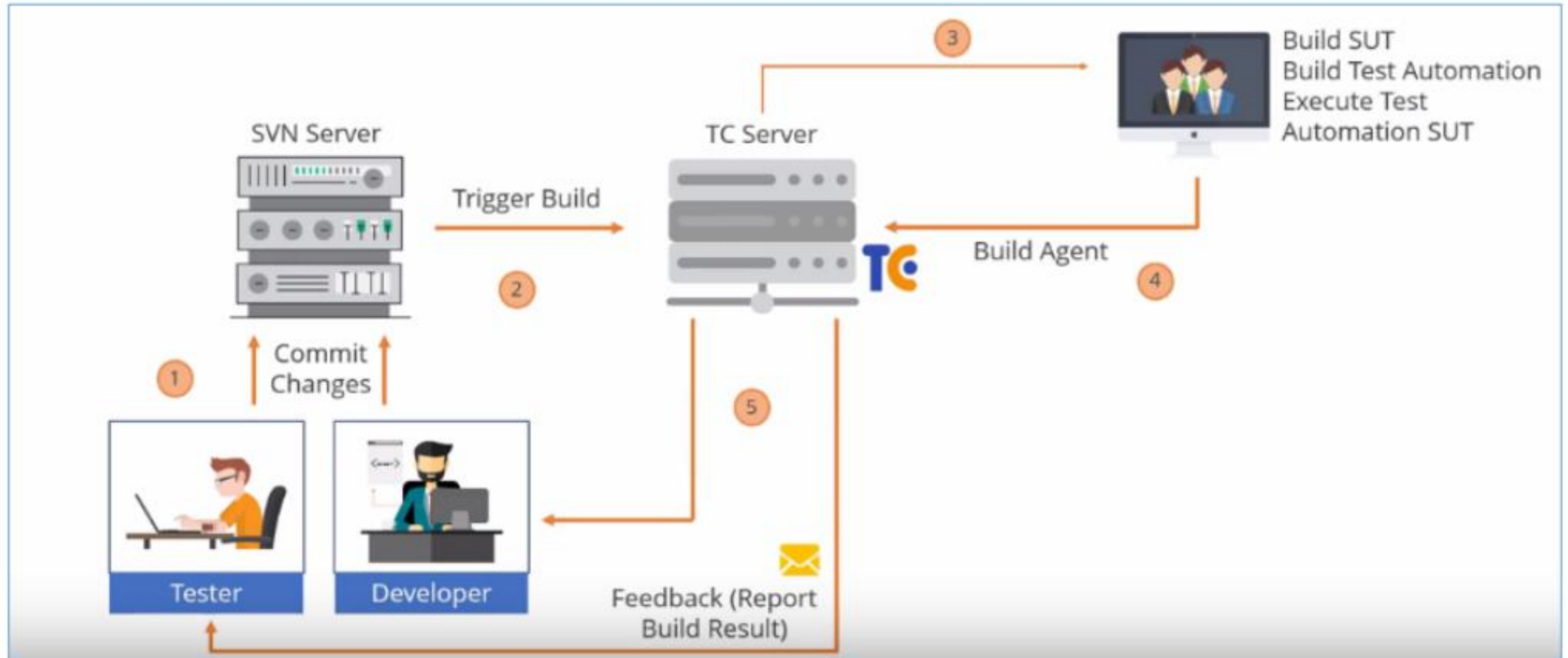


Integration with  
IDEs

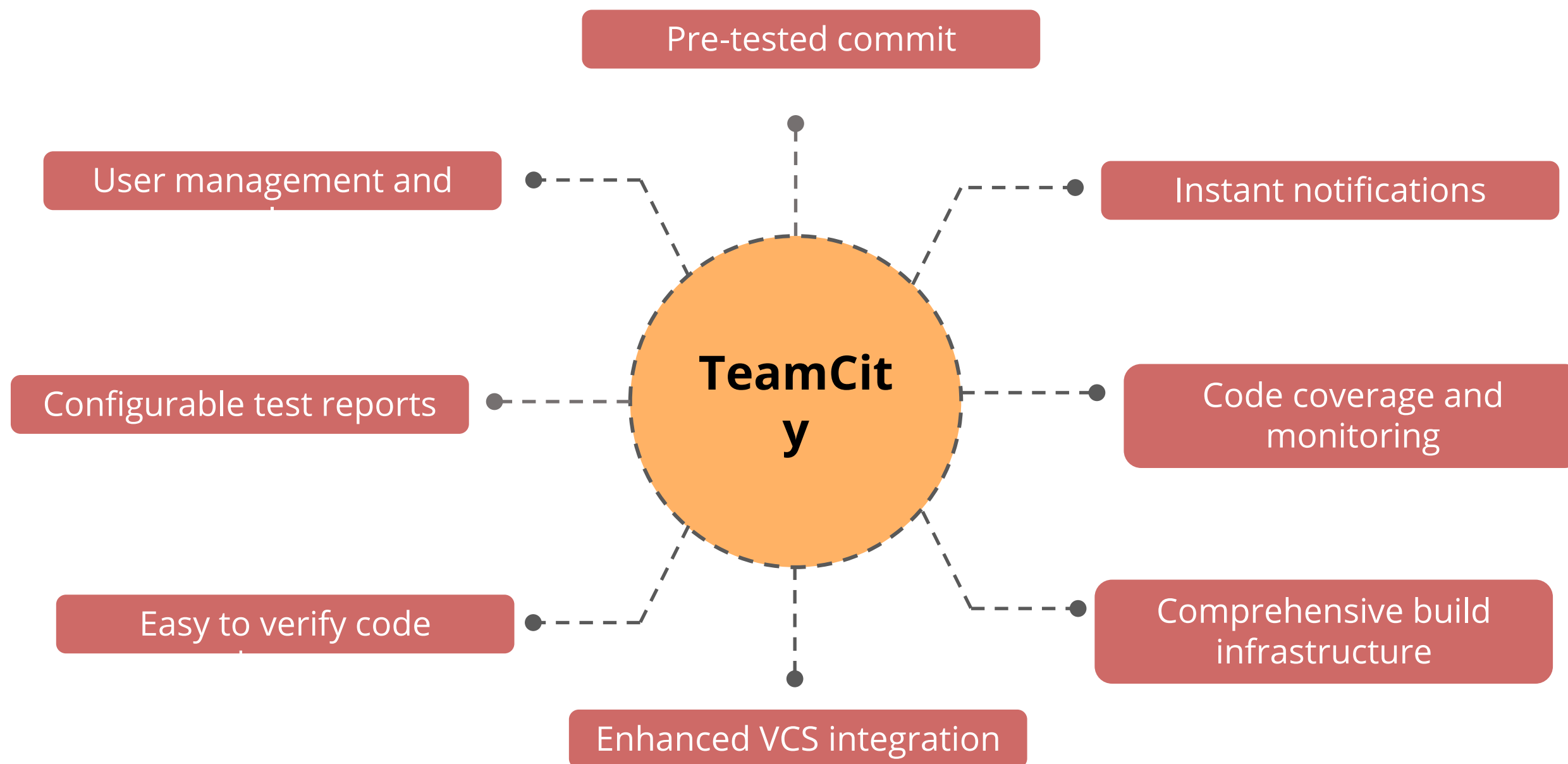


Cross-platform support

# TeamCity Workflow

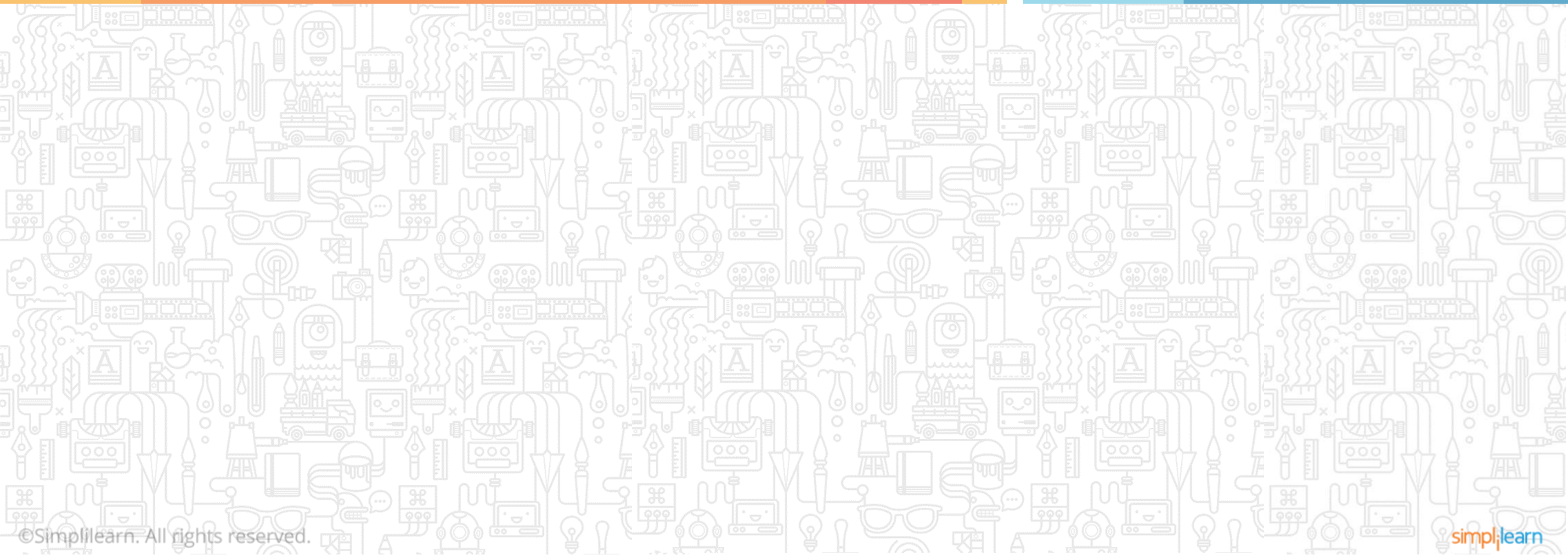


# Popular Features of TeamCity



# Continuous Integration, Continuous Deployment, and Build Tools

## Build Tools and Their Uses



# Build Tools

Build tools are programs that automate the creation of executable applications from the source code. Automation tools allow the build process to be more consistent.

1

**Make**  
Software

3



5



2



4



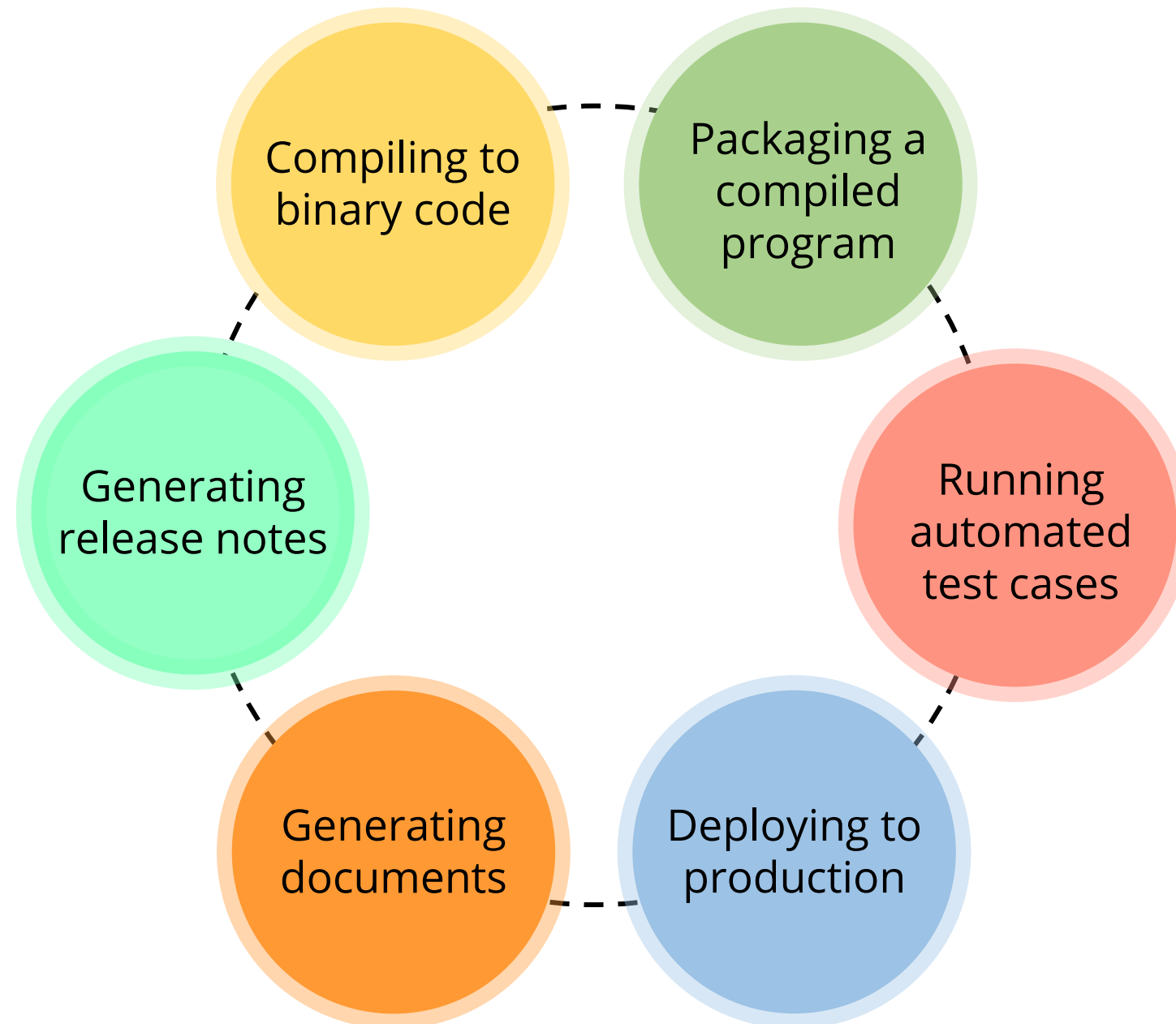
6





# Popular Features of Build Tools

---



# Overview of Apache Ant

---

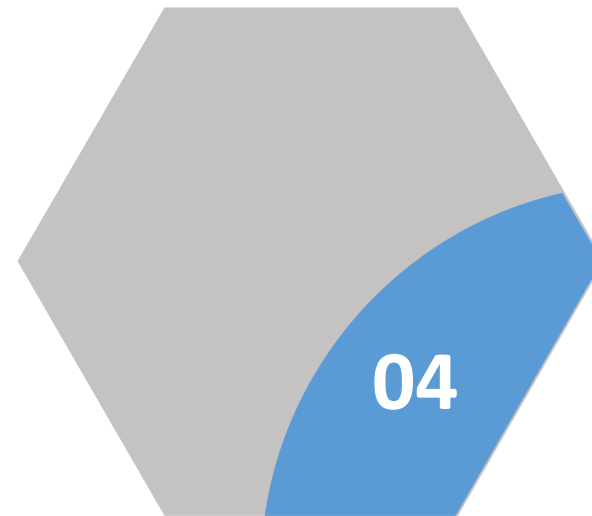
Apache Ant is a Java library and a command-line tool. It aims to drive processes described in build files as targets and extension points dependent on each other.



# Limitations of Apache Ant

---

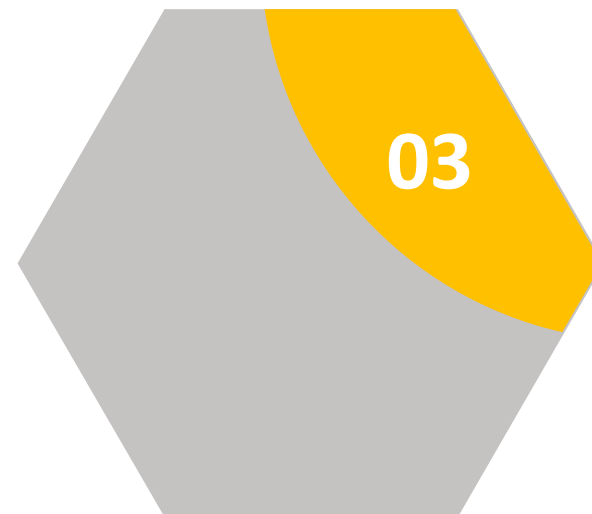
Ant build files are complex and verbose as they are hierarchical and partly ordered



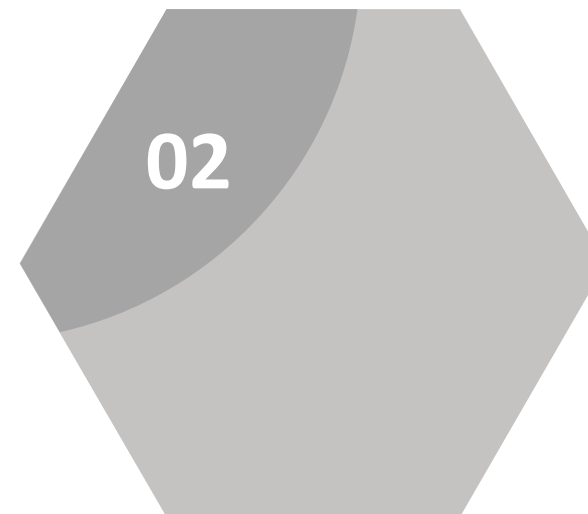
Undefined properties are not raised as errors but left as unexpanded reference



Older tasks use default value, which are not consistent, and changing defaults would break existing Ant scripts

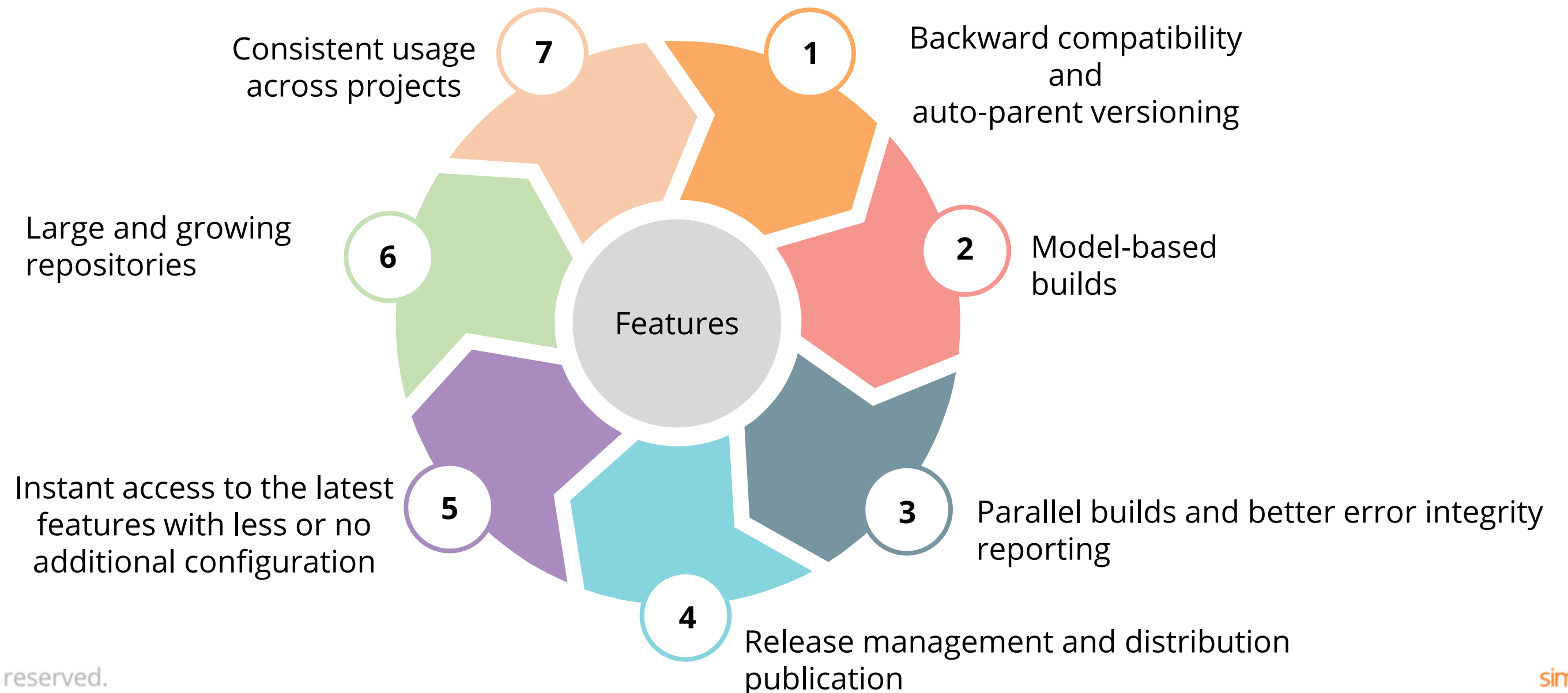


Ant has limited fault handling rules and lazy property evaluation is not supported



# Overview of Maven

Apache Maven is a software project management and comprehension tool. Based on Project Object Model (POM), Maven can manage a project's build reporting and documentation from a central piece of information.



# Drawbacks of Maven

Unreliable  
with  
Eclipse

Unable to depend  
on  
the outcome status

Slow and a  
partial black  
box

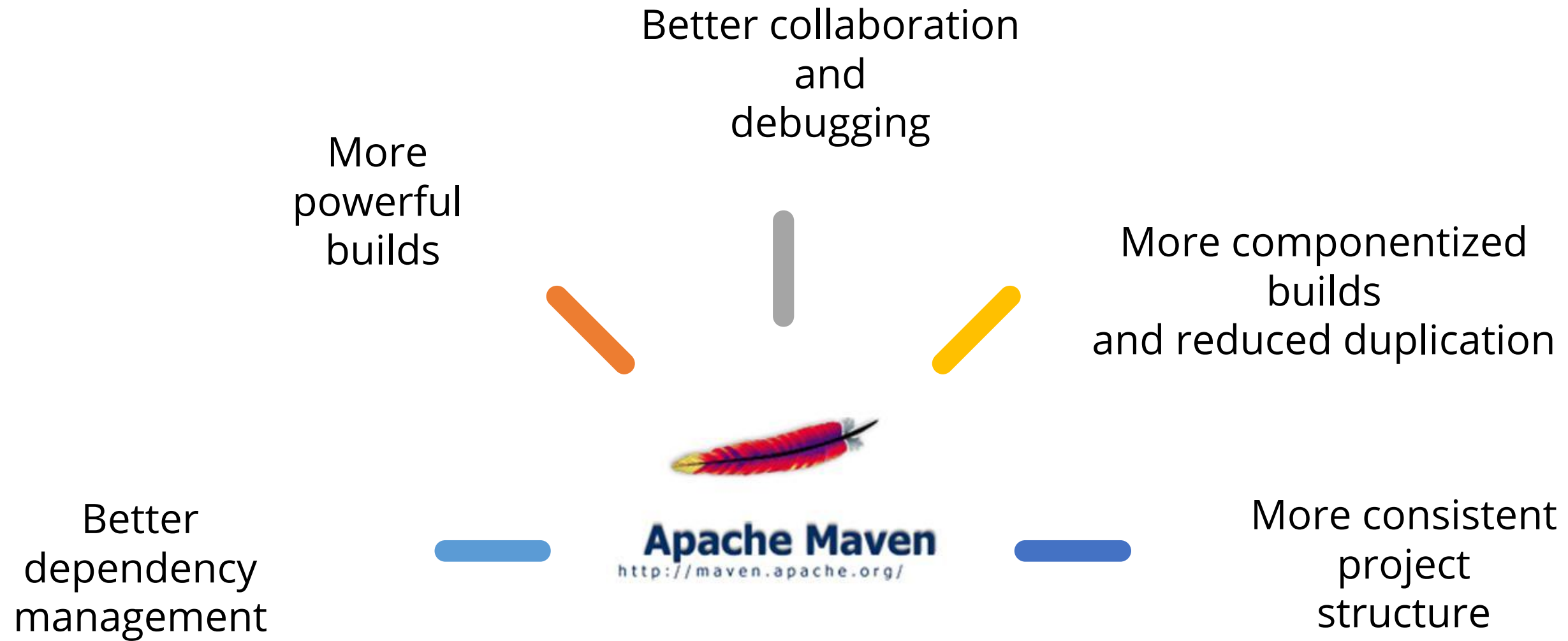
Verbose  
and  
complex





# Maven over Ant

---



# Project Object Model (POM)

---

Project Object Model is an XML representation of a Maven project, which provides general configurations such as project's name, its owner, and its dependencies on other projects.

- The POM needs to define the Group ID, Artifact ID, and Version
- The packaging should also be declared – the default is jar

```
<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>eu.totaleclipse</groupId>
  <artifactId>calculator</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
</project>
```

# Overview of Grunt

Grunt is a JavaScript-based task runner, which is used to automate repetitive tasks in a workflow. It can be used as a command-line tool for JavaScript objects.



Eases workflow such as writing a set up file



Helps in automation of repetitive task with less effort



Minifies the files such as .html, .CSS




Includes built-in tasks to extend functionality of plugins





Speeds the development flow and enhances performance




Supports small infrastructure which is a best fit for new codebase



Aims at reducing the chances of errors during repetitive tasks



Currently, it has over 4000 plugins and can be used in larger production sites



# Overview of Gulp

Gulp is an open-source JavaScript toolkit used as a streaming build system in front-end web development. It automates time-consuming and repetitive tasks involved in development.



## Features

- Code minification and concatenation
- Usage of pure JavaScript code
- Converts LESS or SASS to CSS compilation
- Manages file manipulation in the memory



## Advantages

- Easy to code
- Easy to test the web apps
- Plugins are simple to use



## Disadvantages

- More number of dependencies
- Multiple tasks cannot be performed
- Configuration is tedious

# Assisted Practice

Duration: 50 mins

## Continuous Integration with Jenkins and Maven

**Problem Statement:** You are given a project to configure Jenkins, poll Git commits and build the project code using Maven on your Ubuntu operating system.

**Access:** Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.



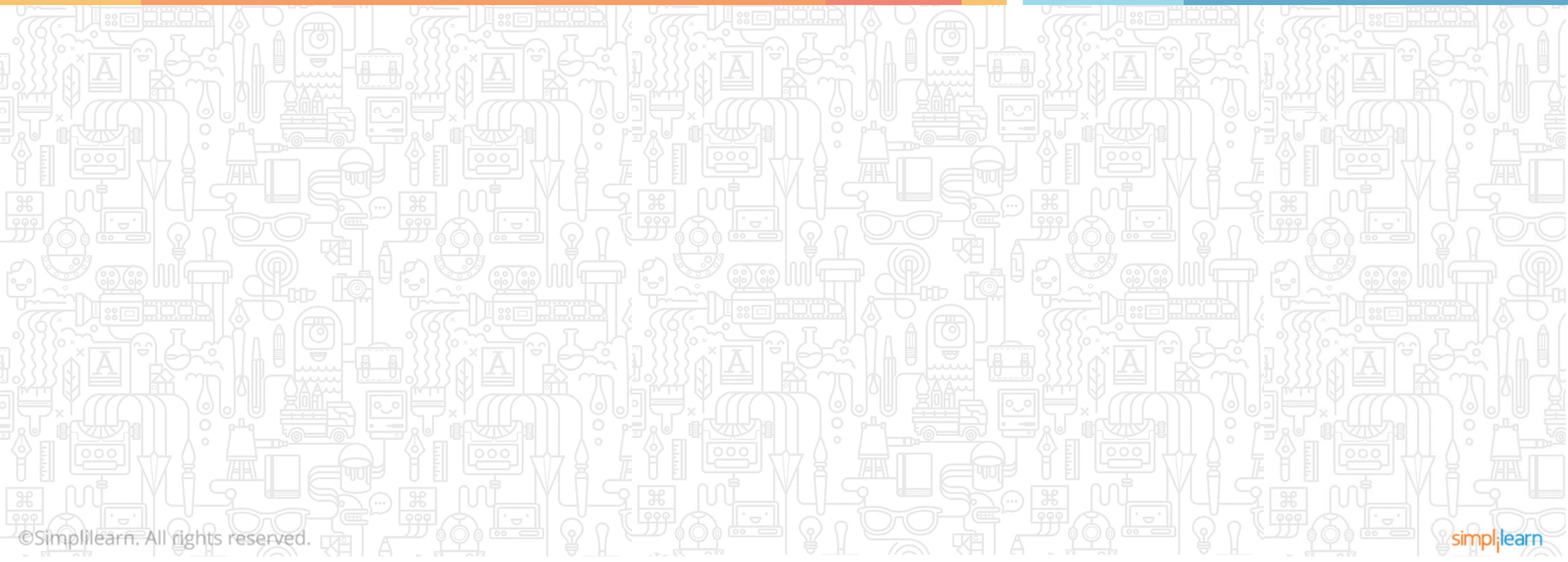
# Assisted Practice: Guidelines to Configure Jenkins and Maven

---

1. Login to your Ubuntu lab provided with the course.
2. Login to Jenkins and create the first Jenkins job.
3. Install and configure Maven.
4. Configure Jenkins with Java, Git, and Maven.
5. Create a Jenkins job for your maven build project and run the project.
6. Poll Git for commits and automatically trigger the build.
7. Build the trigger using Push mechanism instead of Pull.
8. Repeat steps 6 and 7 multiple times to observe the results at **console output** section.

# Continuous Integration, Continuous Deployment, and Build Tools

Working with Jenkins File



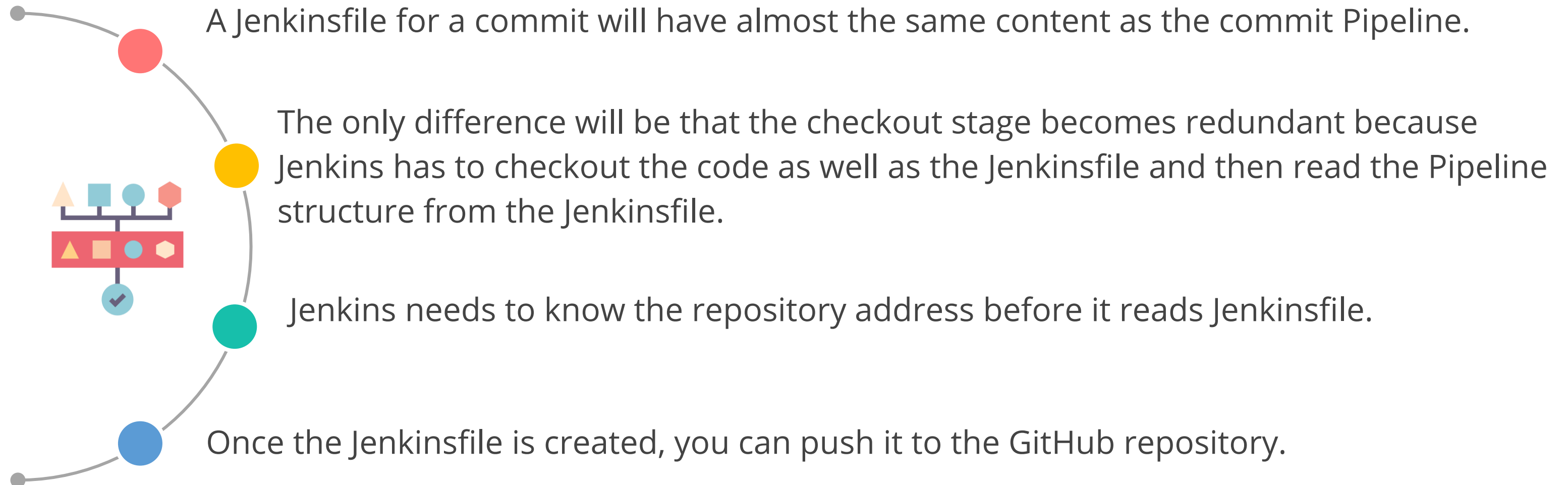
# Jenkinsfile

---

- The Pipeline definition in the **Jenkinsfile** can be committed to the repository along with the source code.
- This method is more consistent as the Pipeline structure becomes strongly tailored to the project.

For example: if you don't need the code compilation because your programming language is interpreted, then you won't have the Compile stage.

# Creating Jenkinsfile



# Creating Jenkinsfile

Here is a sample Jenkinsfile for a commit Pipeline:

```
Pipeline {  
  agent any  
  stages {  
    stage("Compile") {  
      steps {  
        sh "./gradlew compileJava"  
      }  
    }  
    stage("Unit test") {  
      steps {  
        sh "./gradlew test"  
      }  
    }  
  }  
}
```



# Running Jenkinsfile

---

After the Jenkinsfile is pushed, we can run the Pipeline as given below:

Open the Pipeline configuration.

Change *Definition* from *Pipeline script* to *Pipeline script from SCM*.

Select *Git* in SCM.

Enter Repository URL.

Save the Pipeline.

After saving, the build will always run from the current version of Jenkinsfile in the repository.

# Running Jenkinsfile

The screenshot below shows how to run a Pipeline from a Jenkinsfile:

The screenshot displays the Jenkins configuration interface for a Pipeline script sourced from a SCM. The 'SCM' dropdown is set to 'Git'. Under the 'Repositories' section, the 'Repository URL' is configured as 'https://github.com/leszko/calculator.git', and the 'Credentials' are set to '- none -'. There are buttons for 'Advanced...', 'Add Repository', and 'Add'. The 'Branches to build' section shows a 'Branch Specifier (blank for 'any')' set to '\*/master', with an 'Add Branch' button and a red 'X' icon. The 'Repository browser' is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. At the bottom, the 'Script Path' is set to 'Jenkinsfile'.

Pipeline script from SCM	
SCM	Git
Repositories	<div>Repository URL: <input type="text" value="https://github.com/leszko/calculator.git"/></div> <div>Credentials: <input type="text" value="- none -"/> <button>Add</button></div> <div><button>Advanced...</button></div> <div><button>Add Repository</button></div>
Branches to build	<div>Branch Specifier (blank for 'any'): <input type="text" value="*/master"/></div> <div><button>Add Branch</button></div>
Repository browser	<input type="text" value="(Auto)"/>
Additional Behaviours	<button>Add</button>
Script Path	Jenkinsfile

# Build with Jenkinsfile

---

- The Build stage of the Pipeline will be where source code is assembled, compiled, or packaged.
- The Jenkinsfile is not a replacement for an existing build tool such as GNU/Make, Maven, Gradle, etc.
- It can be considered a glue layer to bind the multiple phases of a project's development lifecycle like build, test, and deploy together.
- Jenkins has plugins for invoking practically any build tool.

# Test with Jenkinsfile

- Running automated tests is a crucial component of any successful continuous delivery process.
- Jenkins has a number of test recording, reporting, and visualization facilities provided by a number of plugins.
- The example below uses the JUnit step, provided by the JUnit plugin. If tests fail, the Pipeline is marked ***unstable***.

## Jenkinsfile (Declarative Pipeline)

```
pipeline {
  agent any

  stages {
    stage('Test') {
      steps {
        /* `make check` returns non-zero on test failures,
        * using `true` to allow the Pipeline to continue nonetheless
        */
        sh 'make check || true'
        junit '**/target/*.xml'
      }
    }
  }
}
```

# Test with Jenkinsfile

- Deployment can be anything from publishing built artifacts to an Artifactory server, to pushing code to a production system.
- The example below shows a Deploy Pipeline. At this stage of the example Pipeline, both the *Build* and *Test* stages have been successfully executed.

## Jenkinsfile (Declarative Pipeline)

```
pipeline {  
    agent any  
  
    stages {  
        stage('Deploy') {  
            when {  
                expression {  
                    currentBuild.result == null || currentBuild.result == 'SUCCESS'  
                }  
            }  
            steps {  
                sh 'make publish'  
            }  
        }  
    }  
}
```



# Key Takeaways

You are now able to:

- ✓ Describe the importance of continuous integration and continuous deployment
- ✓ List the features of Jenkins and demonstrate their uses
- ✓ List the features of TeamCity and demonstrate their uses
- ✓ Select a suitable build tool for your organization





**Knowledge  
Check**

**1**

**What advantage does Maven have over Ant?**

- a. There isn't one
- b. Ant only compiles code
- c. Maven is easier to configure
- d. It resolves dependencies



Knowledge  
Check

1

What advantage does Maven have over Ant?

- a. There isn't one
- b. Ant only compiles code
- c. Maven is easier to configure
- d. It resolves dependencies



The correct answer is **d. It resolves dependencies**

**It resolves dependencies on external jar files.**

**Knowledge  
Check  
2**

**What advantage does continuous integration provide?**

- a. It simplifies the build process
- b. It stops developers checking in bad code
- c. Build errors are quickly detected and reported
- d. There are no real advantages





Knowledge  
Check  
2

What advantage does continuous integration provide?

- a. It simplifies the build process
- b. It stops developers checking in bad code
- c. Build errors are quickly detected and reported
- d. There are no real advantages



The correct answer is **c. Build errors are quickly detected and reported**

**Continuous integration can also perform testing and can generate documentation.**

**Knowledge  
Check**

**3**

**What does POM stand for?**

- a. Project Object Model
- b. Project Oriented Model
- c. Project Operational Model
- d. Purpose Only Manufacturing



**Knowledge  
Check**

**3**

**What does POM stand for?**

- a. Project Object Model
- b. Project Oriented Model
- c. Project Operational Model
- d. Purpose Only Manufacturing



The correct answer is **a. Project Object Model**

**Knowledge  
Check**

**4**

## What is continuous deployment?

- a. A deployment server
- b. Deployment tool
- c. Open source deployment server for containers
- d. Minimizing the time elapsed between writing new code and using new code in production



Knowledge  
Check

4

## What is continuous deployment?

- a. A deployment server
- b. Deployment tool
- c. Open source deployment server for containers
- d. Minimizing the time elapsed between writing new code and using new code in production



The correct answer is **d. Minimizing the time elapsed between writing new code and using new code in production**

**Continuous deployment refers to automated, faster, and quicker deployments of code into production.**

**Knowledge  
Check**

**5**

**Which of the following is NOT a continuous deployment tool?**

- a. Microsoft Visual Studio
- b. GitHub
- c. ElectricFlow
- d. Bamboo





Knowledge  
Check

5

Which of the following is NOT a continuous deployment tool?

- a. Microsoft Visual Studio
- b. GitHub
- c. ElectricFlow
- d. Bamboo



The correct answer is **b. GitHub**

**GitHub is not a continuous deployment tool as it is a source code repository.**

# Lesson-End Project

Duration: 60 mins

## Continuous Integration with Jenkins, Git, and Maven

### Problem Statement:

Create a FreeStyle project in Jenkins and complete the following:

- Install “Email Extension plugin” in Jenkins.
- Configure Gmail in Jenkins.
- Receive an email when build fails and succeeds.

**Access:** Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.



# Thank You