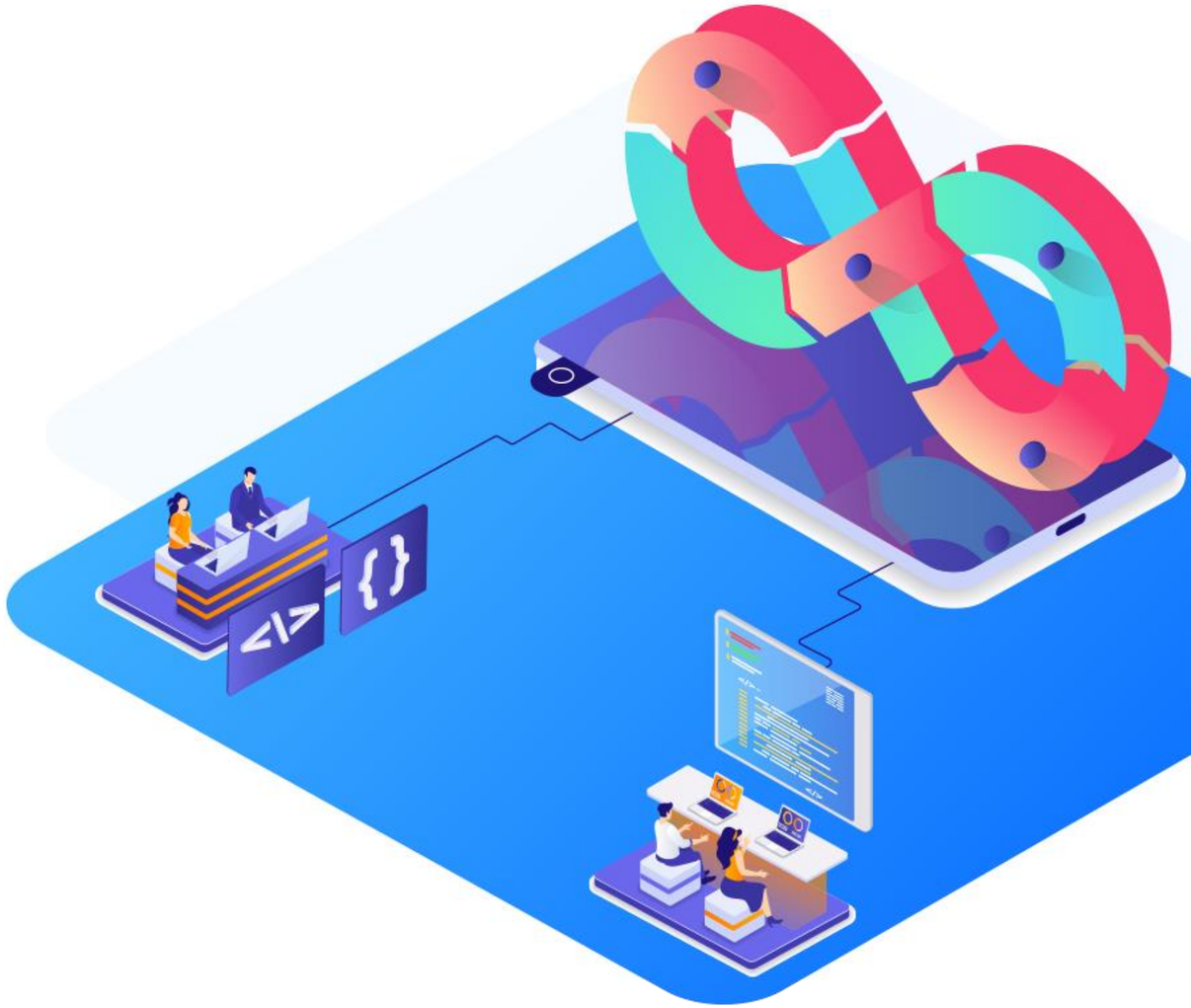
DevOps Foundations: Version Control and CI/CD with Jenkins



Jenkins Integrations

Learning Objectives

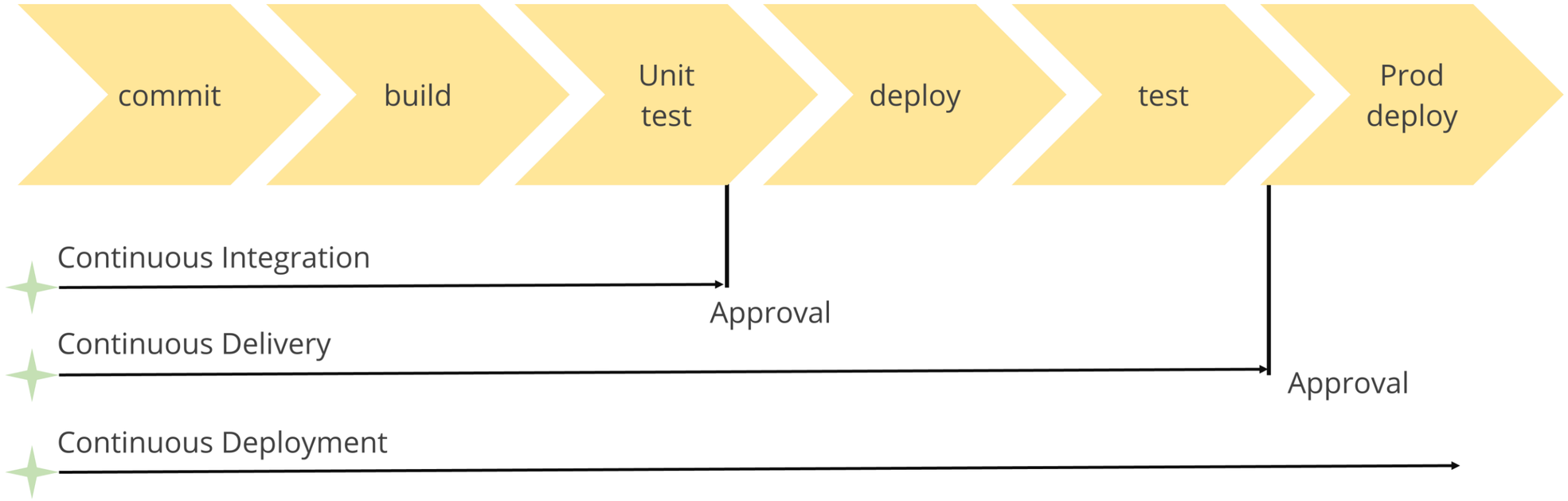
By the end of this lesson, you will be able to:

- Create Jenkins pipeline for continuous deployment
- Implement Apache Tomcat setup for deployment environments
- Determine code scanning metrics with Jenkins tool for analysis and continuous improvement
- List and compare different code scanning tools integrated with Jenkins for optimal code quality
- Integrate Jenkins with Slack for collaborative communication

Getting Started with Continuous Deployment in Jenkins

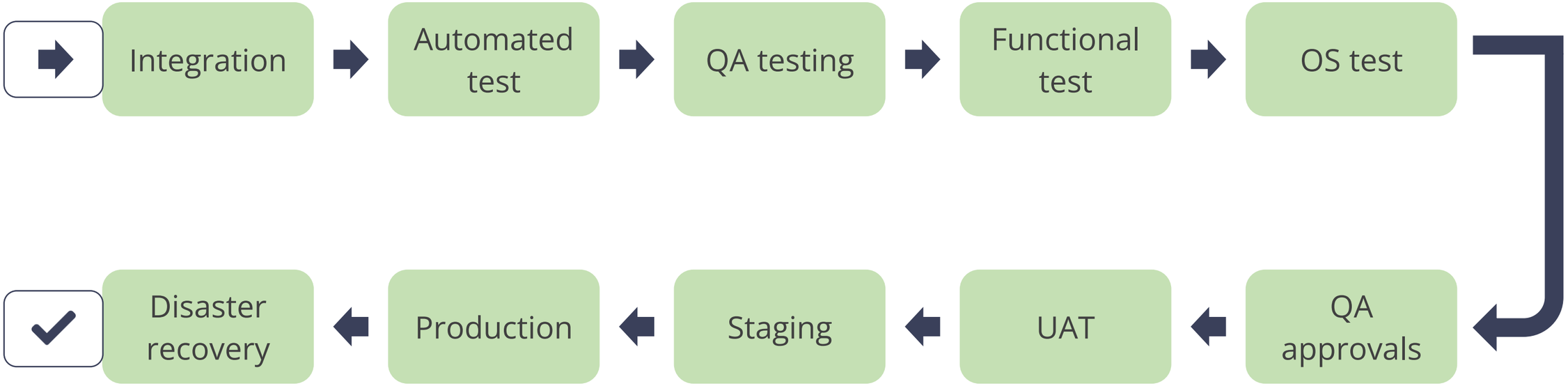
Continuous Deployment Using Jenkins Pipeline

It refers to automating the entire software delivery process, from code acquisition to deployment in production, using Jenkins and defining the workflow stages in a Jenkinsfile.



Continuous Deployment Using Jenkins Pipeline

The continuous deployment workflow using Jenkins pipeline is shown below:



Continuous Deployment Workflow

Continuous Deployment Using Jenkins Pipeline

Integration	Push code to version control system
Automated test	Run automated tests to catch bugs early
Build	Build and package the application for deployment
Functional test	Execute functional tests to verify the working of a new feature
OS test	Run test to evaluate operating system compatibility

Continuous Deployment Using Jenkins Pipeline

UAT	Conduct user acceptance testing (UAT) with real users
QA approvals	Gain QA approval for production deployment
Production deployment	Deploy application to production environment
Disaster recovery	Maintain disaster recovery plan for quick restoration

Continuous Deployment Using Jenkins Pipeline

Advantages of employing Jenkins pipeline for continuous deployment are:

Faster deployments	Automates deployments, reducing manual steps and getting features to users quicker
Improved software quality	Deployments with automated testing leads to faster feedback and higher quality code
Reduced risk of errors	Minimizes human error during deployments, leading to more reliable releases
Increased developer productivity	Spends less time on deployments, freeing them to focus on core development tasks
Simplified workflow	Provides a clear and repeatable deployment process for the team

Installing Deploy Plugin to Implement Continuous Deployment

Jenkins provides several deployment plugins that can be used to implement continuous deployment. Some of the available plugins are:

Deploy to Container

- Deployment plugin for Tomcat, JBoss, and Glassfish webservers
- Supports **Tomcat 4.x/5.x/6.x/7.x/8.x/9.x**, **JBoss 3.x/4.x/5.x/6.x/7.x**, and **Glassfish 2.x/3.x/4.x**

Deploy WebLogic

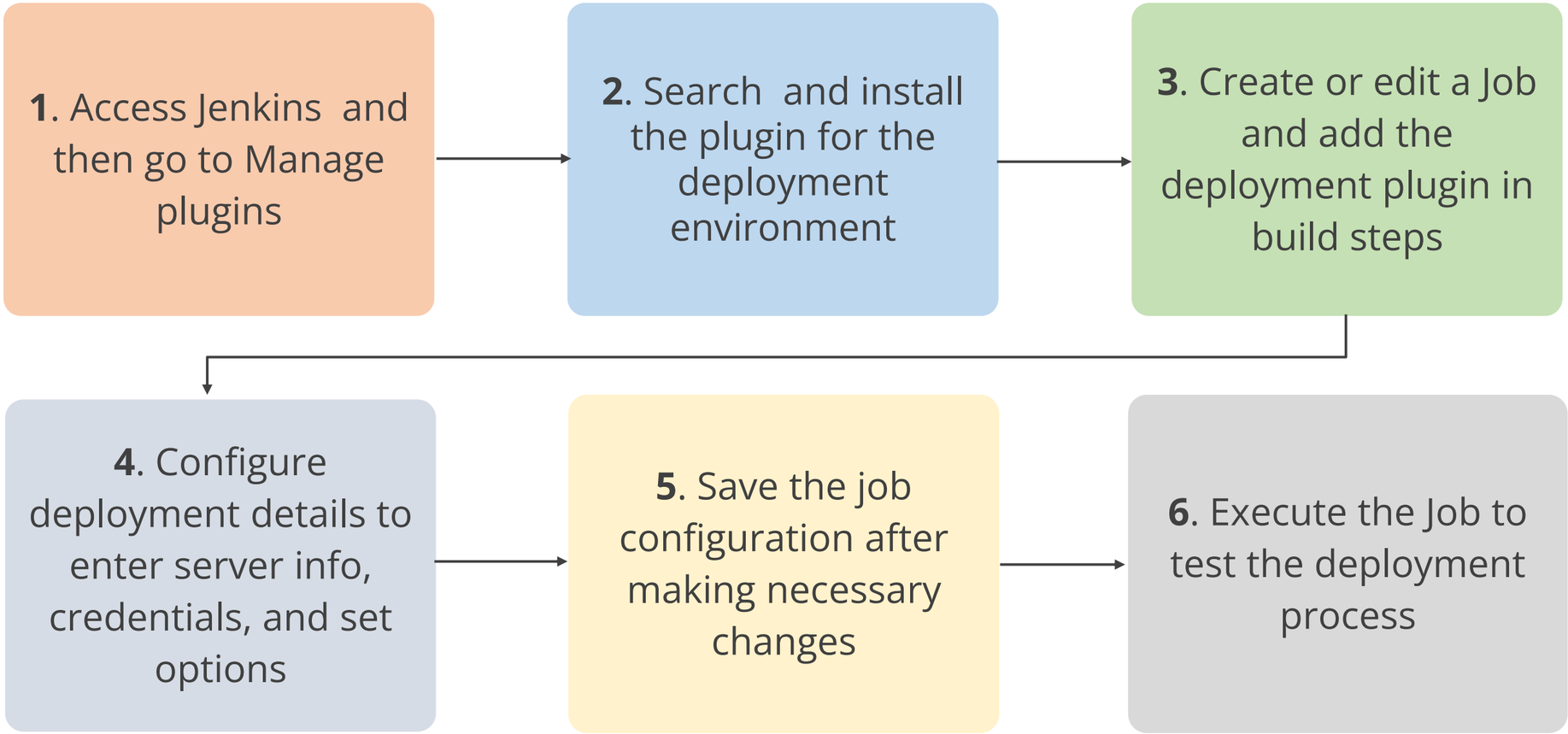
- Deployment plugin for WebLogic webserver

Deploy WebSphere

- Deployment plugin for WebSphere webserver

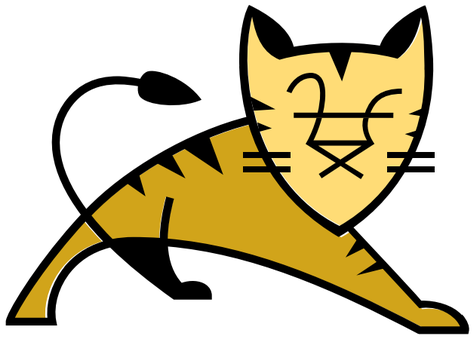
Installing Deploy Plugin to Implement Continuous Deployment: Steps

Here are the steps to install and configure the deploy plugin for implementing continuous deployment:



Setting up Apache Tomcat

Apache Tomcat serves as the target environment for deploying the Java web applications built with Jenkins.



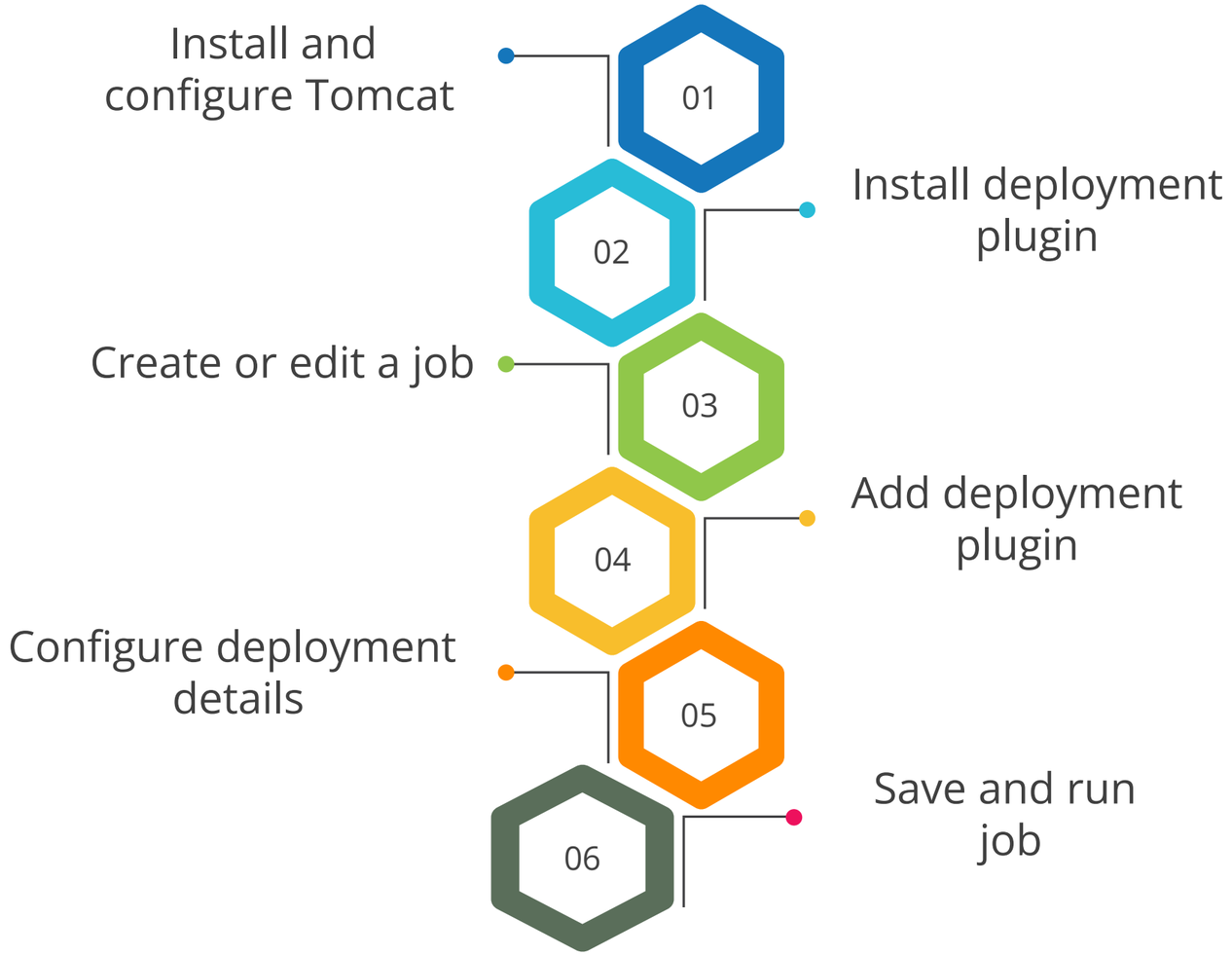
Working of Apache Tomcat:

- Listens to client requests
- Loads the respective servlet class using servlet mapping
- Executes the servlet class
- Unloads the servlet class

Jenkins automates the process of building the application and pushing it to the preconfigured Tomcat server, streamlining the CI/CD pipeline.

Setting up Apache Tomcat: Steps

Here are the steps to set up Apache Tomcat:



Setting up Apache Tomcat: Steps

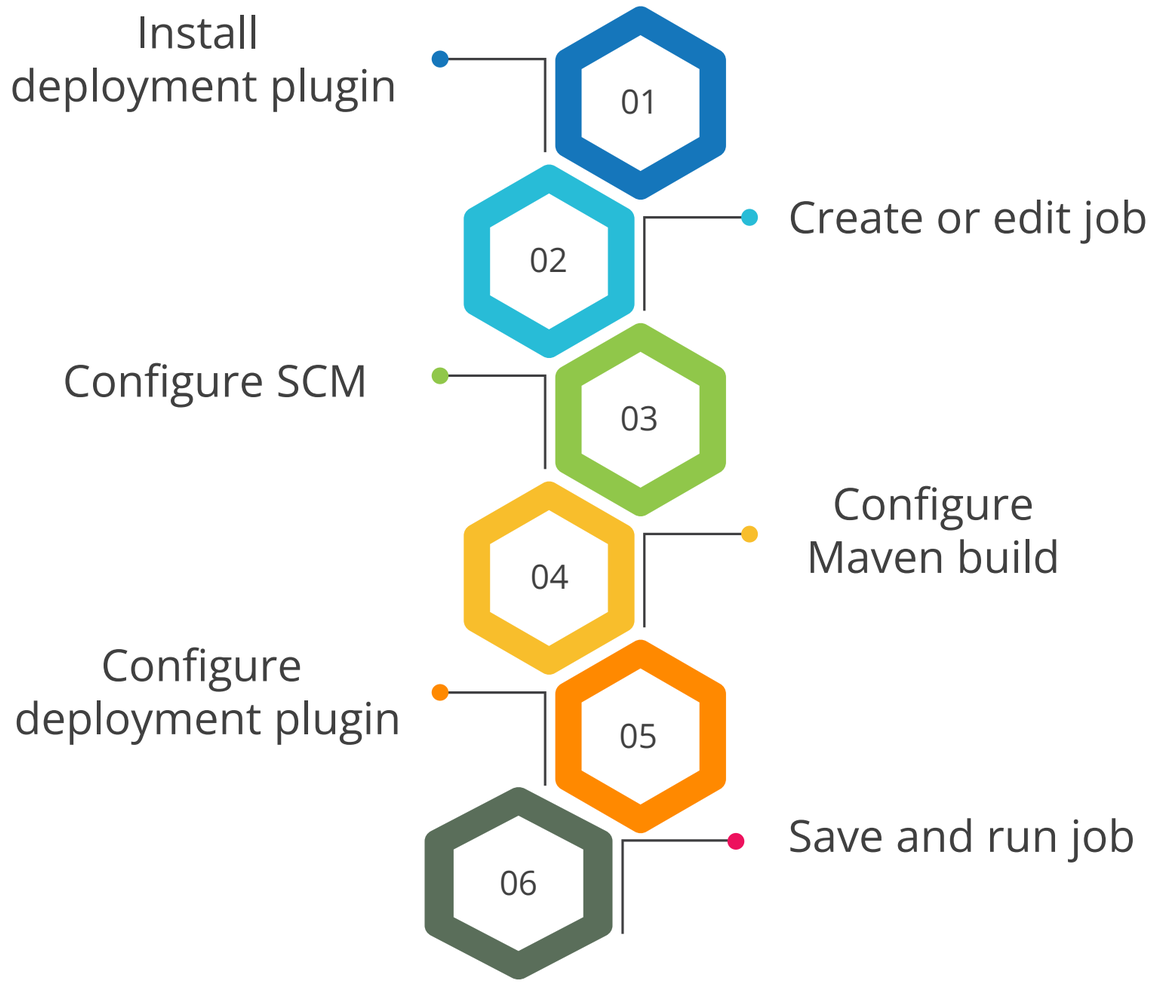
Install and configure Tomcat	Install Tomcat on the server and configure it (users, roles, context paths)
Install deployment plugin	Install a plugin for deploying to Tomcat (for example: Deploy to Tomcat)
Create or edit a job	Create a new Jenkins job or edit an existing one for your application

Setting up Apache Tomcat: Steps

Add deployment plugin	Include the installed deployment plugin in the build steps of the job configuration
Configure deployment details	Specify server details (hostname, port, context path), credentials, and deployment options (artifact location, environment variables) within the plugin
Save and run job	Save the job configuration and run the Jenkins job to trigger deployment to the Tomcat server

Maven Jenkins Job for Deploying Apache Tomcat

Below are the steps for setting up a Jenkins job with Maven build steps and deployment configurations to deploy applications to an Apache Tomcat server:



Maven Jenkins Job for Deploying Apache Tomcat: Steps

Install deployment plugin

- Search and install a plugin for deploying Tomcat (for example, Deploy to Tomcat)
- Restart Jenkins for stability

Create or edit job

- Create a new job in Jenkins for the application or edit an existing one
- Assign a clear name reflecting the deployment purpose (for example, Deploy MyWebApp)

Configure SCM

- Select your SCM tool (for example, Git) and provide the application's Git repository URL
- Configure additional SCM options like credentials or specific branches

Maven Jenkins Job for Deploying Apache Tomcat: Steps

Configure maven build

- Choose Invoke Maven in the build section and specify Maven goals to execute (for example, clean package)
- Configure POM locations or Maven options if needed

Configure deployment plugin

- Add a post-build action and choose the installed deployment plugin
- Provide server details like hostname/IP, port (typically 8080), and context path in Tomcat

Save and run job

- Save the job configuration after defining all details
- Run the Jenkins job to initiate the deployment process

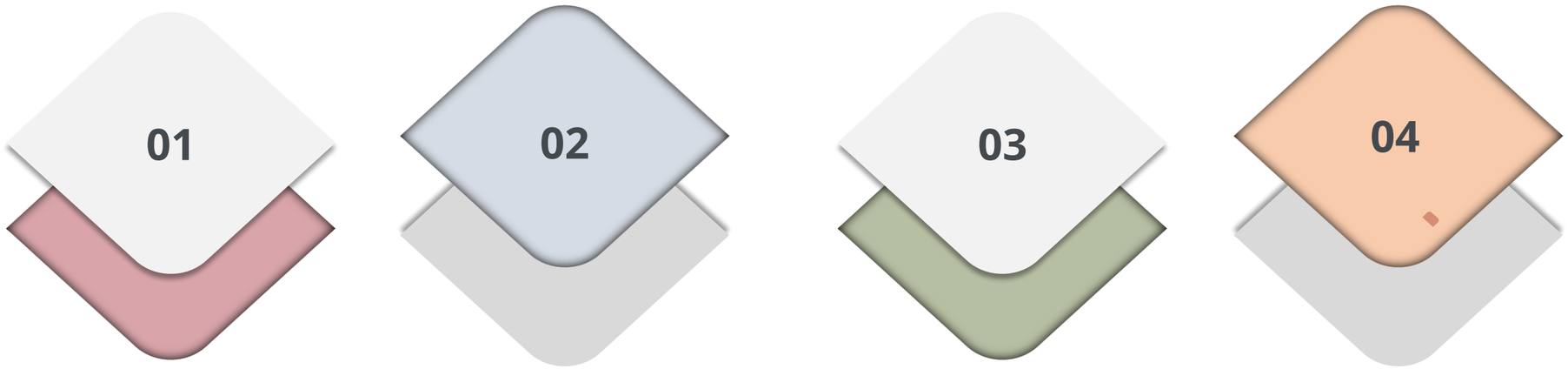
Maven Jenkins Job for Deploying Apache Tomcat: Steps

Below screenshot shows the successful installation of Apache Tomcat:

Maven Jenkins Job for Deploying Apache Tomcat: Benefits

Maven Jenkins job automates deployment of Java web applications to Apache Tomcat server.

This automation offers several advantages:



Reduced manual work and errors

Streamlined CI/CD pipeline

Centralized control and visibility

Improved efficiency and scalability

Maven Jenkins Job for Deploying Apache Tomcat: Benefits

Reduced manual work and errors	Eliminates the need for manual deployments, saving time and effort while minimizing the risk of mistakes
Streamlined CI/CD pipeline	Integrates seamlessly with CI/CD workflows, allowing for automated building, testing, and deployment of your application
Centralized control and visibility	Provides a central hub (Jenkins) to manage deployments for various applications and Tomcat servers
Improved efficiency and scalability	Scales easily to handle more deployments and applications as your project grows

Assisted Practice

Configuring Deploy plugin for performing automated CD **Duration: 20 Min.**

Problem statement:

You have been assigned a task to configure a CI/CD pipeline in Jenkins for deploying a Java application to Tomcat Apache.

Outcome:

By the end of this demo, you will be able to configure a CI/CD pipeline in Jenkins to automatically deploy a Java application to Apache Tomcat, streamlining your development workflow.

Note: Refer to the demo document for detailed steps:

Assisted Practice: Guidelines

Steps to be followed:

1. Install Tomcat Apache 9 on Ubuntu VM
2. Log in to Jenkins CI tool and install Deploy to Container plugin
3. Configure deployment stage in Jenkins pipeline

ASSISTED PRACTICE

Quick Check

As part of your project requirements, you need to deploy a Java web application to an Apache Tomcat server. What is a recommended approach to achieve this using Jenkins?

A. Configure Jenkins to directly deploy to Tomcat

B. Install the Deploy plugin in Jenkins

C. Use Jenkins pipelines for deployment

D. Set up Tomcat without Jenkins integration

Code Scanning Tools

Code Scanning Tools

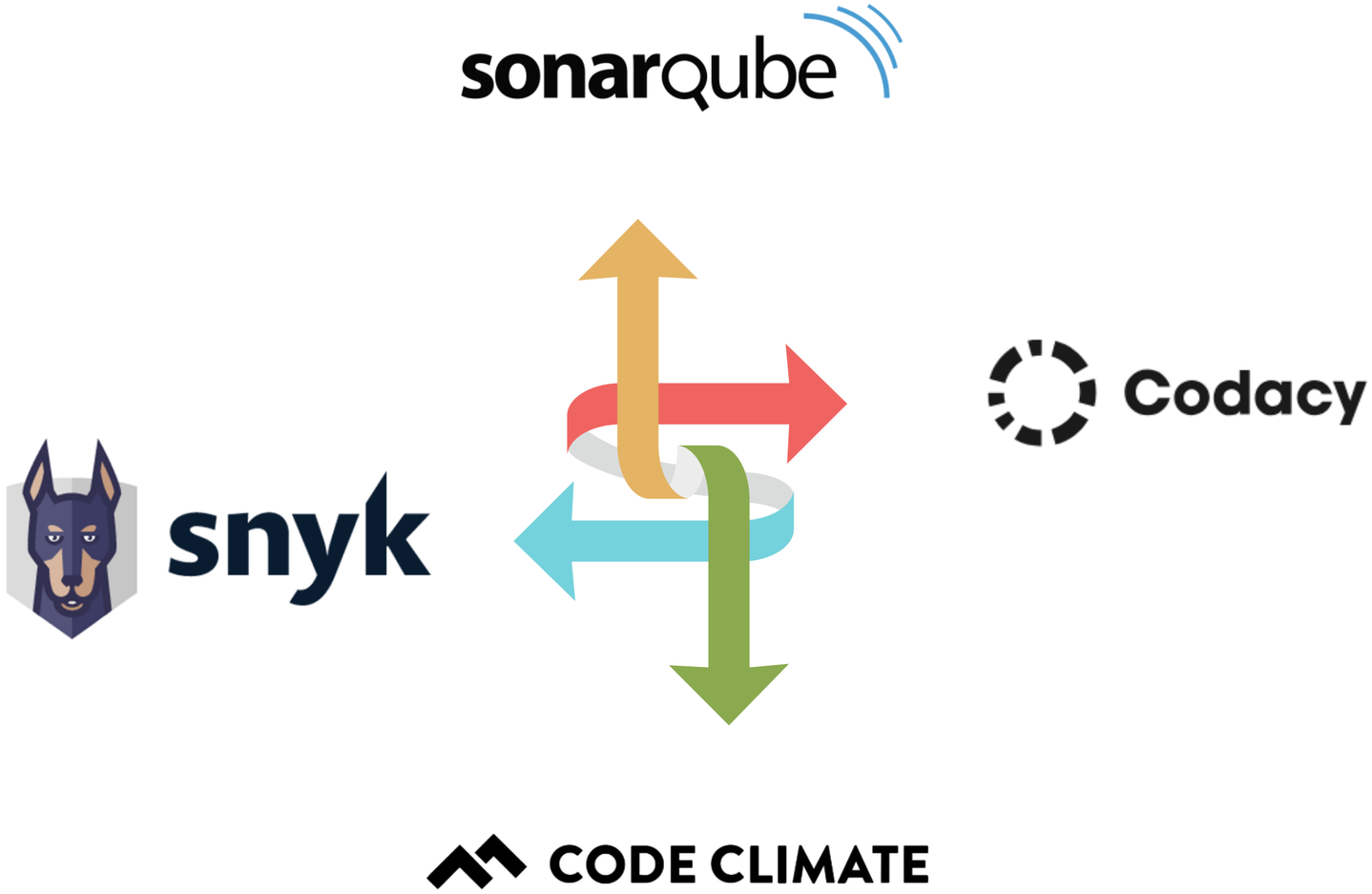
Code scanning tools analyze code and share detailed reports with the developers. They are also known as **Static Application Security Test** (SAST) tools.

Code scanning tools

- Can be easily integrated with IDE
- Can be integrated with continuous integration process
- Can check code vulnerabilities and coding standards

Code Scanning Tools

Below are some of the code scanning tools:



Code Scanning Tools

Integrates code scanning in the CI/CD pipeline, analyzing code during builds to find bugs, vulnerabilities, and inefficiencies for better code quality

Streamlines code analysis and reviews, integrating seamlessly into CI workflows for real-time issue detection and improved development efficiency

Code Scanning Tools

Focuses on security; it scans code during Jenkins builds, pinpointing vulnerabilities within your project's open-source dependencies and your own codebase

Enables development teams to deliver higher-quality code and offers static analysis for languages such as PHP, Java, Python, and Ruby, fostering improved code efficiency

Code Scan Metrics

Below metrics provide insights into various aspects of the codebase:

Code quality	Measures code complexity, duplication, maintainability, and compliance with coding standards
Security	Identifies vulnerabilities, security hotspots, and compliance with security standards
Performance	Tracks execution time, memory usage, and resource leaks for performance optimization

Code Scan Metrics

Coverage	Measures the effectiveness of tests in covering different code sections and execution paths
Dependencies	Assesses the health of external libraries your project relies on, identifying conflicts and outdated versions
Issue tracking	Tracks the number of bugs, code review comments, and time to resolve issues for better code quality control

Quick Check

As a developer working with Jenkins on a software development project, which tool would be most appropriate for performing code quality and security analysis?

A. SpotBugs

B. Jenkinsfile checker

C. GitHub actions

D. SonarQube

Slack Collaborative Tool

Introduction to Collaboration Tools

Collaboration tools centralize communication, file sharing, and task management, boosting teamwork and efficiency.

These tools streamline workflows, reduce errors, and increase productivity by facilitating seamless information sharing and task organization.

Slack

It is a collaboration tool that boosts software development by streamlining communication, knowledge sharing, and integrations with developer tools.

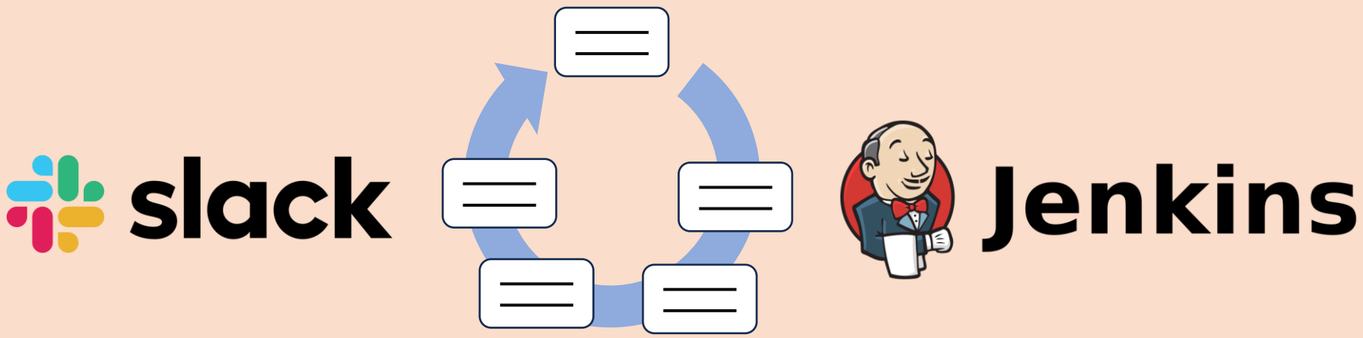
- Slack can be used by Jenkins to send notifications from all the jobs to the team
- Slack channels can be used to send specific notifications to specific teams
- Slack provides Jenkins hook as an app that can be accessed on the Slack workspace

It is a popular and well-crafted platform offering instant messaging, file transfers, and powerful message search.

Integration between Jenkins and Slack

It facilitates seamless communication with real-time notifications and enhances visibility in the build statuses and deployments.

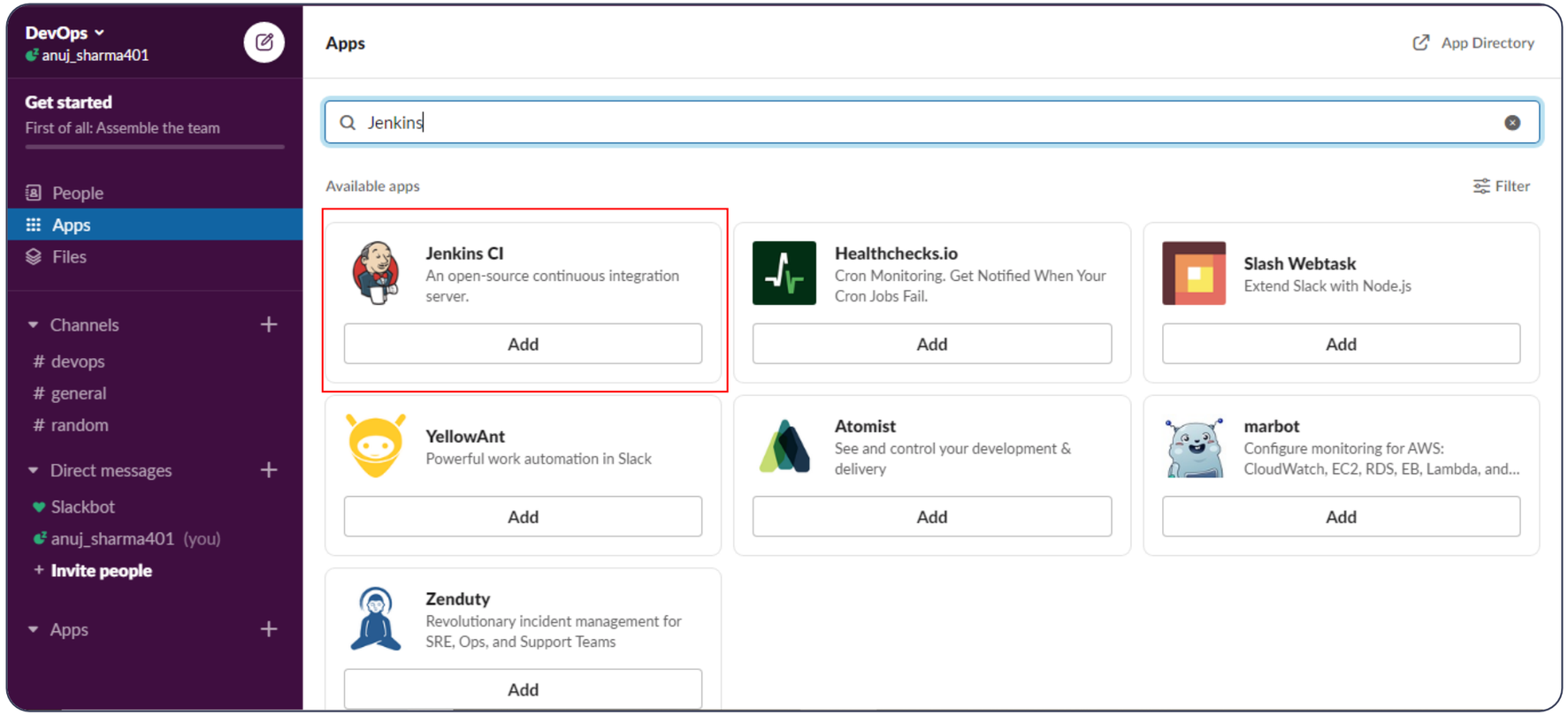
Integrating Slack with Jenkins



Integration

Integration between Jenkins and Slack

To set up Jenkins Build notifications, click the **Apps** tab on the left panel of the Slack workspace. Then, select **Jenkins CI** app.



Integration between Jenkins and Slack: Benefits

Benefits of Jenkins and Slack integration:

Real-time notifications:

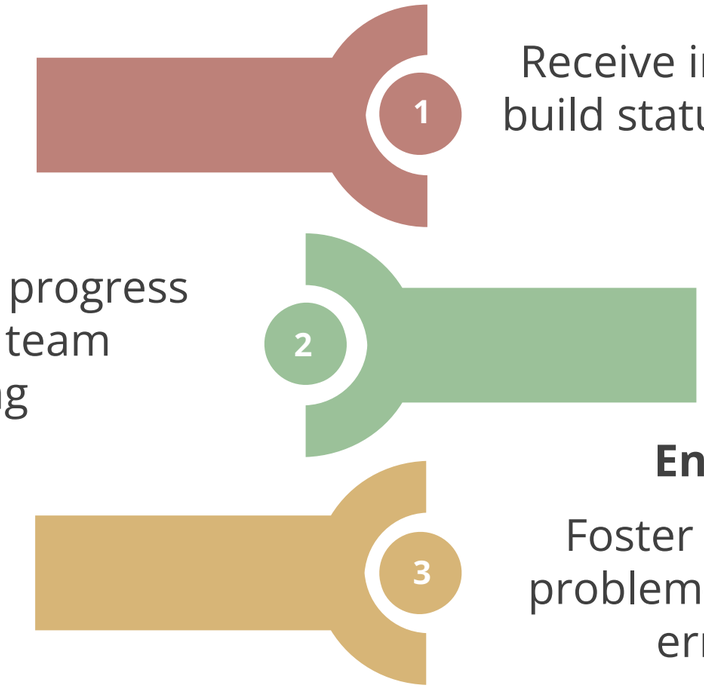
Receive immediate updates on Jenkins build statuses, deployments, and events in slack

Improved visibility:

Monitor build and deployment progress directly in slack, enhancing team awareness and reducing platform switching

Enhanced collaboration:

Foster better communication and problem-solving by discussing builds, errors, and deployments



Assisted Practice

Setting up Slack channel for configuring build notifications **Duration: 20 Min.**

Problem statement:

You have been assigned a task to set up a Slack channel for configuring build notifications for receiving updates about the build process.

Outcome:

By the end of this demo, you will be able to set up a Slack channel to receive build notifications, enabling streamlined updates about the build process in your workflow.

Note: Refer to the demo document for detailed steps:

Assisted Practice: Guidelines

Steps to be followed:

1. Log in to the Jenkins CI tool and create a freestyle job
2. Set up Slack channel
3. Configure Slack in Jenkins

Quick Check

As a project manager, how does the integration between Jenkins and Slack contribute to faster software development cycles?

A. By reducing team collaboration

B. By automating project planning

C. By enabling real-time notifications and feedback

D. By eliminating the need for version control systems

Key Takeaways

- Continuous deployment using Jenkins pipeline refers to automating the entire software delivery process.
- Jenkins provides several deployment plugins that can be used to implement continuous deployment.
- Code scanning tools analyze code and share detailed reports with the developers.
- Collaboration tools centralize communication, file sharing, and task management, boosting teamwork and efficiency.
- Slack is a collaboration tool that boosts software development by streamlining communication.

Integrating GitHub with Jenkins

Duration: 25 Min.

Project agenda: To create a Jenkinsfile pipeline script in GitHub and use it to set up a Jenkins pipeline job for cloning, compiling, testing, and packaging the codebase

Description: You are a DevOps engineer managing a web application on GitHub. To enhance the efficiency of the deployment process, you have taken the initiative to set up a Jenkins server. As part of this setup, a Jenkinsfile must be integrated it into your project's GitHub repository. This Jenkinsfile is responsible for orchestrating essential tasks such as code checkout, Maven-based building, and testing. Whenever you push updates to GitHub, Jenkins automatically triggers the pipeline, ensuring that your changes are seamlessly integrated and deployed.

LESSON-END PROJECT

Integrating GitHub with Jenkins

Duration: 25 Min.

Perform the following:

1. Create a Jenkinsfile pipeline script file in a GitHub repository
2. Create the Jenkins pipeline job
3. Execute the Jenkins job

Expected deliverables: A Jenkins pipeline job set up to perform tasks such as code checkout, Maven-based building, and testing whenever updates are pushed in GitHub

LESSON-END PROJECT

Thank You