DevOps Shack

Jenkins Part-1 | Theory

Jenkins is an open-source automation server that is widely used for building, testing, and deploying software projects. It is one of the most popular and powerful tools in the field of Continuous Integration and Continuous Deployment (CI/CD) and offers a wide range of features to facilitate automation and improve the software development and delivery process. Here are the key features of Jenkins in detail:

1. **Automation of Repetitive Tasks:**
    - Jenkins automates repetitive tasks such as building, testing, and deploying code, freeing up developers from manual, error-prone processes.
2. **Integration with Version Control Systems:**
    - Jenkins can integrate with various version control systems like Git, Subversion, Mercurial, and more, allowing it to trigger builds automatically upon code changes.
3. **Extensive Plugin Ecosystem:**
    - Jenkins has a rich ecosystem of plugins (over 1,500 at the time of my last update), which extend its functionality to integrate with various tools and technologies, making it highly customizable.
4. **Distributed Build Support:**
    - Jenkins supports distributed builds, allowing you to set up a master-slave architecture where build tasks can be distributed across multiple machines to improve performance and scalability.

5. **Pipeline as Code:**
    - Jenkins offers a feature called "Pipeline" that allows you to define your build and deployment pipelines as code (Jenkinsfile). This enables version control, sharing, and reuse of your build processes.
6. **Wide Range of Build Environments:**
    - Jenkins supports a variety of build environments, including different operating systems and build tools, making it versatile for a wide range of projects.
7. **Continuous Integration and Continuous Deployment (CI/CD):**
    - Jenkins is a powerful tool for implementing CI/CD pipelines, ensuring that code changes are automatically built, tested, and deployed to various environments.
8. **Customizable Dashboards:**
    - Jenkins provides a web-based user interface with customizable dashboards and widgets, allowing teams to monitor build and deployment statuses, trends, and test results.
9. **Notification and Alerts:**
    - Jenkins can send notifications and alerts through email, chat services (Slack, Microsoft Teams), or other communication channels to keep the team informed about build and deployment outcomes.
10. **Security and Access Control:**
    - Jenkins offers robust security features with role-based access control (RBAC) to control who can perform specific actions and access certain resources within the system.
11. **Integration with Testing Frameworks:**
    - Jenkins integrates with various testing frameworks (JUnit, TestNG, Selenium, etc.) to automate testing as part of the CI/CD process.
12. **Artifact Management:**
    - Jenkins can integrate with artifact repositories (e.g., Nexus, Artifactory) to manage and store build artifacts and dependencies.

13. **Plugins for Cloud Services:**
    - Jenkins has plugins for cloud services such as AWS, Azure, and Google Cloud, enabling the provisioning of resources and deployment to cloud environments.
14. **Community and Support:**
    - Jenkins has a large and active community of users and developers, providing support, documentation, and a wealth of resources.
15. **Freemium and Open Source:**
    - Jenkins is open-source and free to use, making it accessible to organizations of all sizes. Commercial support options are also available.

Manage Jenkins

Configure System

Configuring the system in Jenkins involves setting up various global settings that affect the behavior of Jenkins as a whole. These settings impact how Jenkins interacts with your environment, agents (slaves), and jobs. Here's a guide on how to configure the system in Jenkins:

1. **Access Jenkins Configuration:**
    - Log in to the Jenkins web interface.
    - Click on "Manage Jenkins" on the left sidebar.
2. **Configure Global Settings:**
    - Click on "Configure System" to access the global configuration settings.
3. **Configure JDK and Tools:**
    - You can configure the Java Development Kit (JDK) installations that Jenkins will use for building and testing your projects. Click on "JDK installations..." to configure JDKs.
    - You can also configure other tools like Git, Maven, Ant, etc., under the "Global Tool Configuration" section.

4. **Manage Node and Cloud:**
    - If you're using the master-slave architecture, configure Jenkins nodes (slaves) under the "Manage Nodes and Clouds" section. You can add, configure, and remove slave nodes here.
5. **Configure Security:**
    - Under the "Access Control" section, you can configure security settings like matrix-based security, project-based security, and more. This controls user access to Jenkins and specific jobs.
6. **Configure Global Properties:**
    - The "Global properties" section allows you to set environment variables that apply globally to all jobs and builds.
7. **Configure Jenkins URL:**
    - Under "Jenkins Location," you can set the Jenkins URL. This is important for correct links and notifications.
8. **Configure Email Notifications:**
    - If you want to configure email notifications for build results and alerts, you can do so under the "E-mail Notification" section.
9. **Configure Maven Settings:**
    - If you use Maven, you can configure Maven-related settings, including specifying a settings.xml file.
10. **Configure SCM Trigger Polling:**
    - Under "Polling SCM," you can set the polling interval for jobs that use SCM (Source Code Management) triggers.
11. **Configure Workspace Cleanup:**
    - If you want Jenkins to clean workspaces after builds, you can configure workspace cleanup settings.
12. **Configure Content Security Policy:**
    - Under "Script Security," you can manage and configure the content security policy for running scripts.

13. **Configure Cloud and Docker Agents (if applicable):**
    - If you're using cloud agents or Docker agents, you can configure the relevant settings under the corresponding sections.
14. **Save Changes:**
    - After making the desired configuration changes, scroll down and click "Save" to apply the changes.

Remember to carefully review the documentation and tooltips for each setting to understand their implications before making changes to your Jenkins system configuration. Configuration changes can affect how your builds, jobs, and the Jenkins environment as a whole operate.

Plugins

The choice of Jenkins plugins can vary depending on your specific needs and the technologies you use. However, there are several popular and widely used Jenkins plugins that are considered essential for many CI/CD pipelines. Here are 15 important Jenkins plugins:

1. **Pipeline Plugin:** This plugin enables you to define and manage your build and deployment pipelines as code using Jenkinsfile.
2. **Git Plugin:** Allows Jenkins to integrate with Git repositories, making it easy to trigger builds based on code changes.
3. **GitHub Integration Plugin:** Provides deeper integration with GitHub, enabling you to trigger builds and update GitHub statuses.
4. **Docker Plugin:** Integrates Jenkins with Docker, allowing you to build, push, and run Docker containers as part of your CI/CD process.
5. **ArtifactDeployer Plugin:** Facilitates the archiving and deployment of build artifacts to various repositories.
6. **Maven Plugin:** Streamlines the integration of Apache Maven with Jenkins, making it easy to build and deploy Java projects.
7. **JIRA Integration Plugin:** Enables integration with Atlassian JIRA for tracking and managing issues related to builds and deployments.
8. **Email Extension Plugin:** Allows you to send customizable email notifications upon build success, failure, or other events.

9. **Copy Artifact Plugin:** Lets you copy build artifacts from one job to another, useful for passing artifacts between pipeline stages.

10. **Workspace Cleanup Plugin:** Automatically cleans up workspaces to free up disk space after builds.

11. **Build Timeout Plugin:** Adds the ability to set build timeouts, ensuring that builds don't hang indefinitely.

12. **Credentials Plugin:** Provides a secure way to manage and use credentials, such as API keys, passwords, and SSH keys, within your Jenkins jobs.

13. **NodeJS Plugin:** Simplifies the installation and management of Node.js versions on Jenkins agents.

14. **Blue Ocean Plugin:** Offers a modern, user-friendly UI for creating and visualizing Jenkins pipelines, making it easier to understand and troubleshoot builds.

15. **Slack Notification Plugin:** Integrates Jenkins with Slack, allowing you to send build notifications and updates to Slack channels.

These plugins are just a starting point, and the choice of plugins should align with your specific project requirements and technology stack. Jenkins has a vast plugin ecosystem, and you may need additional plugins based on your use case. Always ensure that you keep your plugins up to date to benefit from the latest features and security fixes.