Jenkins Cheatsheet

A comprehensive cheat sheet covering essential Jenkins concepts, commands, and configurations for DevOps and Cloud environments.

Core Concepts & Setup

Jenkins Fundamentals

Continuous Integration (CI): Automating the building, testing, and integration of code changes.

Continuous Delivery (CD): Automating the release of validated code to a repository.

Continuous Deployment: Automating the release of code directly into production.

Pipeline: A user-defined model of a CD pipeline. Code that defines the entire build, test, and deployment process.

Node: A machine which is part of the Jenkins environment and is capable of executing Pipelines.

Agent: Defines where the Pipeline will execute. Can be a specific node, a Docker container, or any available agent.

Installation (Ubuntu)

Install Java (OpenJDK 8 or 11)	`sudo apt update`
	 `sudo apt install openjdk-8-jdk`

Add Jenkins repository key	`wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -`
Add Jenkins repository to apt	`sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`

Install Jenkins	`sudo apt update`
	 `sudo apt install jenkins`

Start Jenkins service	`sudo systemctl start jenkins`
Check Jenkins status	`sudo systemctl status jenkins`

Initial Setup

1. Access Jenkins web interface (default port 8080).
2. Retrieve initial admin password from `/var/lib/jenkins/secrets/initialAdminPassword`.
3. Install suggested plugins or select plugins to install.
4. Create admin user.

Pipeline as Code

Declarative Pipeline Syntax

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                // Steps to build the application
            }
        }
        stage('Test') {
            steps {
                // Steps to test the application
            }
        }
        stage('Deploy') {
            steps {
                // Steps to deploy the application
            }
        }
    }
}
```

Pipeline Directives

`agent`	Specifies where the entire Pipeline or a specific stage will execute. Options: `any`, `none`, `label '...'`, `docker {...}`.
`stages`	Contains a sequence of one or more stage directives.
`steps`	Contains a sequence of one or more steps to be executed in a stage.
`environment`	Defines environment variables to be used within the Pipeline.
`options`	Configures Pipeline options, such as `skipDefaultCheckout`, `timeout`, `retry`.
`parameters`	Defines parameters that can be passed to the Pipeline when it's triggered.

Common Steps

`sh 'command'` - Executes a shell command.

`bat 'command'` - Executes a Windows batch command.

`git 'url'` - Checks out code from a Git repository.

`mvn 'goal'` - Executes a Maven goal.

`docker build ...` - Builds a Docker image.

Plugins & Integrations

Popular Plugins

Git Plugin	Integrates with Git repositories for source code management.
Maven Integration Plugin	Provides seamless integration with Maven projects.
Docker Plugin	Enables building and managing Docker containers.
Cobertura Plugin	Generates code coverage reports.
Slack Notification Plugin	Sends notifications to Slack channels.
Email Extension Plugin	Provides enhanced email notification capabilities.
Kubernetes Plugin	Allows Jenkins to dynamically provision and manage build agents in a Kubernetes cluster.

Integration with Cloud Platforms

Jenkins can be integrated with various cloud platforms such as AWS, Azure, and Google Cloud using plugins or CLI tools.

Example (AWS): Use the AWS CLI plugin to interact with AWS services like S3, EC2, and ECS within your Jenkins pipelines.

Credentials Management

Use Jenkins' built-in credentials management to securely store and manage secrets, passwords, and API keys.

Access credentials in your pipelines using the `withCredentials` step.

```
withCredentials([usernamePassword(credentialsId: 'my-credentials', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {
  sh "echo Username: $USERNAME, Password: $PASSWORD"
}
```

Advanced Configuration

Jenkins CLI

The Jenkins CLI allows you to interact with Jenkins from the command line.

Usage:

```
java -jar jenkins-cli.jar -s http://your-jenkins-url:8080 command [options]
```

Example:

```
java -jar jenkins-cli.jar -s http://localhost:8080 safe-restart
```

Security Considerations

- **Enable authentication:** Ensure that Jenkins is protected by user authentication.
- **Use role-based access control (RBAC):** Grant users only the necessary permissions.
- **Secure credentials:** Properly manage and protect credentials.
- **Regularly update Jenkins and plugins:** Keep Jenkins and installed plugins up to date to patch security vulnerabilities.
- **Implement network security:** Restrict network access to Jenkins.

Distributed Builds

Configure Jenkins to distribute builds across multiple nodes (agents) to improve build performance and scalability.

- **Add nodes:** Connect additional machines to your Jenkins master.
- **Configure agents:** Specify labels and resources for each agent.
- **Use labels in your pipelines:** Direct builds to specific agents using the `agent` directive.