Introduction to Jenkins

Module 7: Jenkins

Page 1

Author: Praveen Akkineni Subject: Underline Date: 06/10/2024 3:02:42 PM

Jenkins

Module 7: Contents

- What is Jenkins?
- CI/CD in Jenkins.
- Jenkins installation.
- Freestyle jobs.
- Plugins.
- Pipeline jobs.

Objectives

Introduction to Jenkins

Understand Jenkins and the CI/CD pipeline

- Understand common phraseology used with Jenkins and CI/CD servers.

Understand how to setup Jenkins

- Install and configure a Jenkins server on a virtual machine.

Explore the different builds in Jenkins

- Build an application through a Jenkins freestyle job, integrating with SCM.

What is Jenkins?

What is Jenkins: Module 7

Introduction to Jenkins

What is Jenkins?

Jenkins is an open-source automation server, widely recognized for its capability to provide an easy-to-use continuous integration (CI) and continuous delivery (CD) pipeline.

It enables developers and DevOps teams to automate the building, testing, and deployment phases of their development process, ensuring more rapid and reliable software releases.

Key features of Jenkins

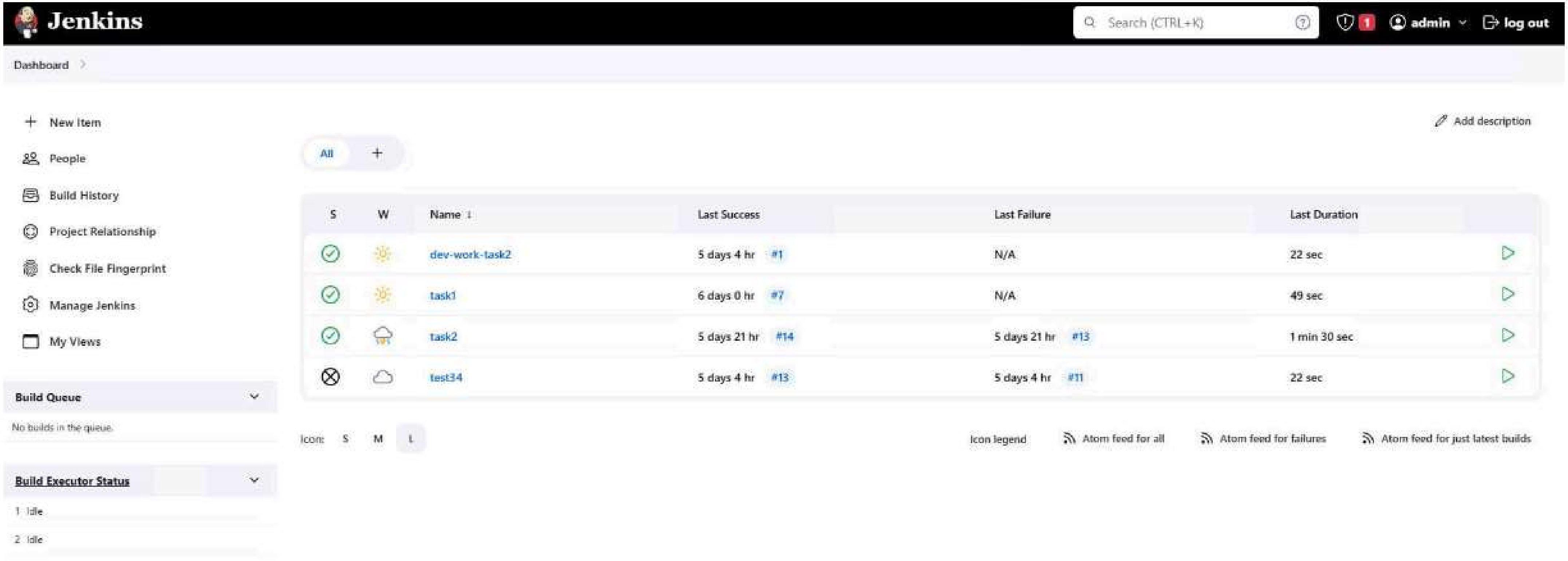
Jenkins has many key features designed to optimise and accelerate development cycles. Some pivotal aspects include:

- **Extensive Plugin Ecosystem:** A rich library of plugins allowing Jenkins to integrate with numerous tools and platforms.
- **Easy Setup and Configuration:** With its intuitive GUI, Jenkins provides an effortless setup and hassle-free configuration, making it accessible to various skill levels.
- **Automation of Build and Test Phases:** Automatically compile, test, and build code, enabling frequent and consistent software releases.
- **Pipeline-as-Code:** Describe the entire CI/CD pipeline through code via a Jenkinsfile, ensuring version control and simplicity in pipeline sharing.

Page 6

Author: Praveen Akkineni   Subject: Underline   Date: 06/10/2024 3:42:10 PM

QA Jenkins GUI



Jenkins

Search (CTRL+K)

1 admin log out

Dashboard >

- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓	☀	dev-work-task2	5 days 4 hr #1	N/A	22 sec	▷
✓	☀	task1	6 days 0 hr #7	N/A	49 sec	▷
✓	🌧	task2	5 days 21 hr #14	5 days 21 hr #13	1 min 30 sec	▷
⊗	☁	test34	5 days 4 hr #13	5 days 4 hr #11	22 sec	▷

Icon: S M L

Icon legend

Atom feed for all

Atom feed for failures

Atom feed for just latest builds

CI/CD in Jenkins

CI/CD in Jenkins: Module 7

Introduction to Jenkins

CI/CD in Jenkins

- **Automated Builds:** Triggering automatic builds upon code commits.
- **Continuous Testing:** Automatically initiating tests to validate code, ensuring reliability.
- **Seamless Deployment:** Automating the deployment process for consistent and reliable software delivery.
- **Pipeline-as-Code:** Utilising Jenkinsfile to define pipelines, ensuring reproducibility and version control.
- **Scalability:** Enabling builds and tests to be distributed across several servers.
- **Collaborative Development:** Promoting a collaborative environment by ensuring code integrations are continuously validated, keeping codebase healthy and agile.

Jenkins installation

Jenkins installation: Module 7

Introduction to Jenkins

Jenkins installation

Jenkins can be installed through native system packages, container tools such as Docker, or simply ran standalone by a machine with the Java Runtime Environment (JRE) installed.

Jenkins can be installed on Linux, Mac OS, and Windows.

The following installation guide is for Ubuntu/Debian. It is recommended to use the installation script from the official Jenkins website:

www.jenkins.io/doc/book/installing

Ensure port 8080 is open on the security group for your VM.

Jenkins installation

Install Java:

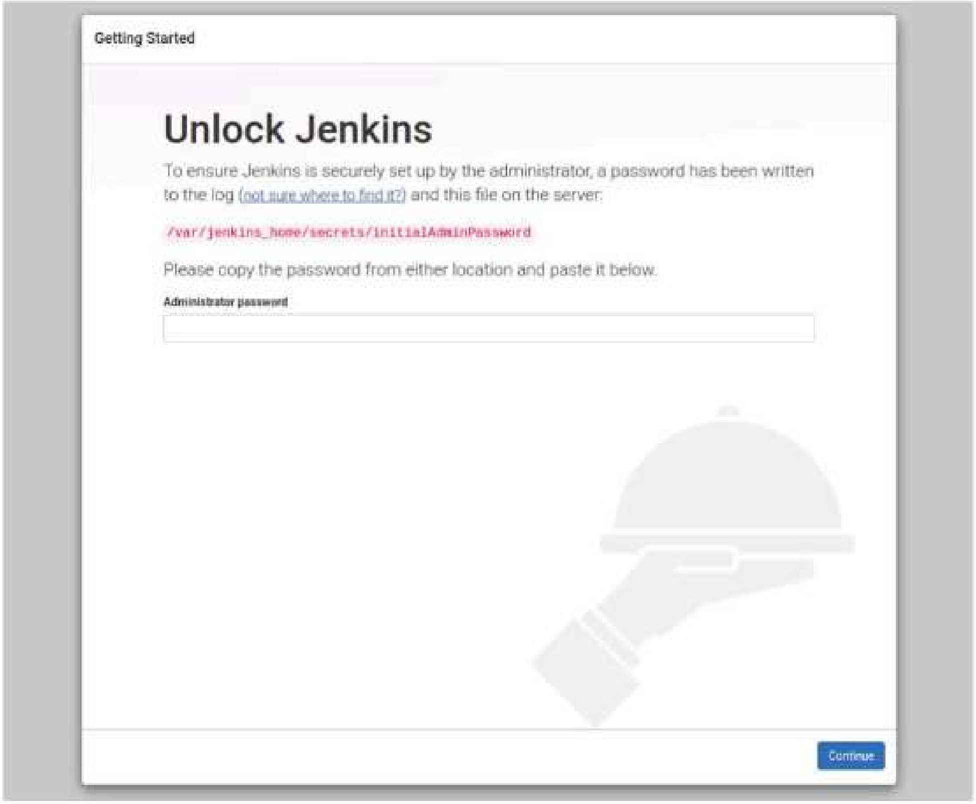
```
sudo apt update
sudo apt install fontconfig openjdk-17-jre
```

Install Jenkins:

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo
tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

Unlocking Jenkins

- To begin the setup process for Jenkins, you will need to go to port 8080 on your machine (in a browser enter your public IP with port 8080 <IP>:8080).
- Jenkins setup requires an initial admin password to unlock the setup and installation. This is a simple verification process Jenkins includes to ensure an administrator is installing the tool.



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

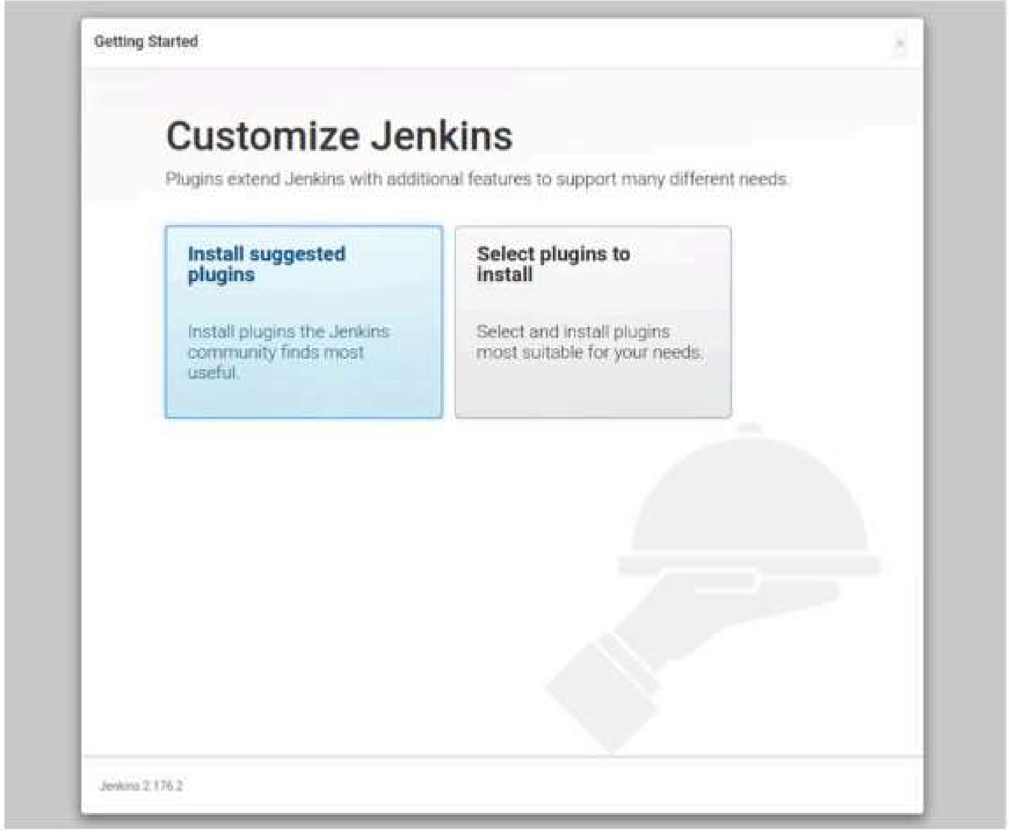
Please copy the password from either location and paste it below.

Administrator password

```
cat /var/jenkins_home/secrets/initialAdminPassword
```

Customise Jenkins

- Jenkins is highly configurable due to the number of plugins that you can install.
- This is fantastic, but if you are new to Jenkins, you may not know what plugins you want!
- Fortunately, the setup gives you the option to install suggested plugins.
  - Select this option.



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

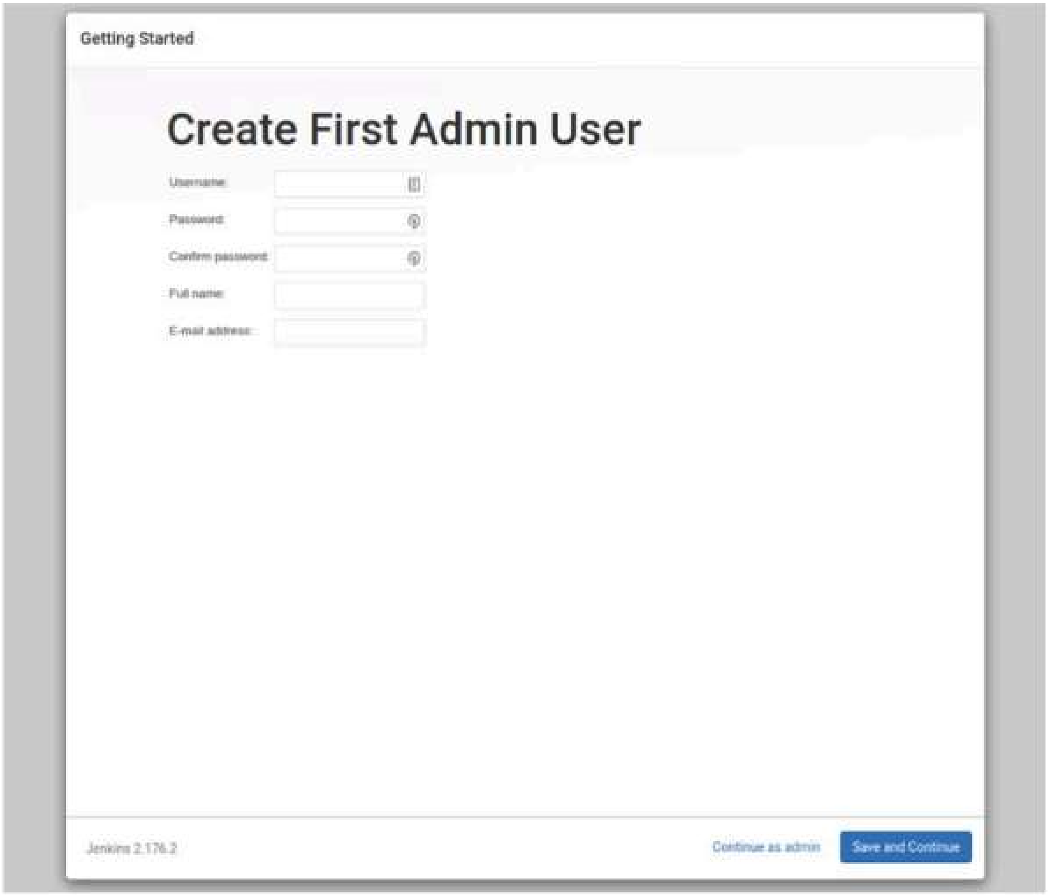
Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.176.2

Create the First Admin User

When prompted to create the first admin user, it's ok to just put 'admin' for each field.



Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.176.2

Dashboard links

- **New Item:** This is for creating new jobs in Jenkins. Jobs are essentially scripts that can be triggered.
- **People:** The users that are registered to this instance of Jenkins.
- **Build History:** A graph displaying the jobs that have been executed over time with this instance of Jenkins.
- **Manage Jenkins:** This is where to go for setting up plugins and other administrative settings for Jenkins.
- **My Views:** You can customise which jobs are listed on the dashboard and how they are presented.

Freestyle jobs

Freestyle jobs: Module 7

Jenkins fundamentals

Freestyle jobs

Jenkins Freestyle jobs provide a simple and straightforward way to configure and run builds, making them an ideal starting point for those new to the Jenkins ecosystem. They provide:

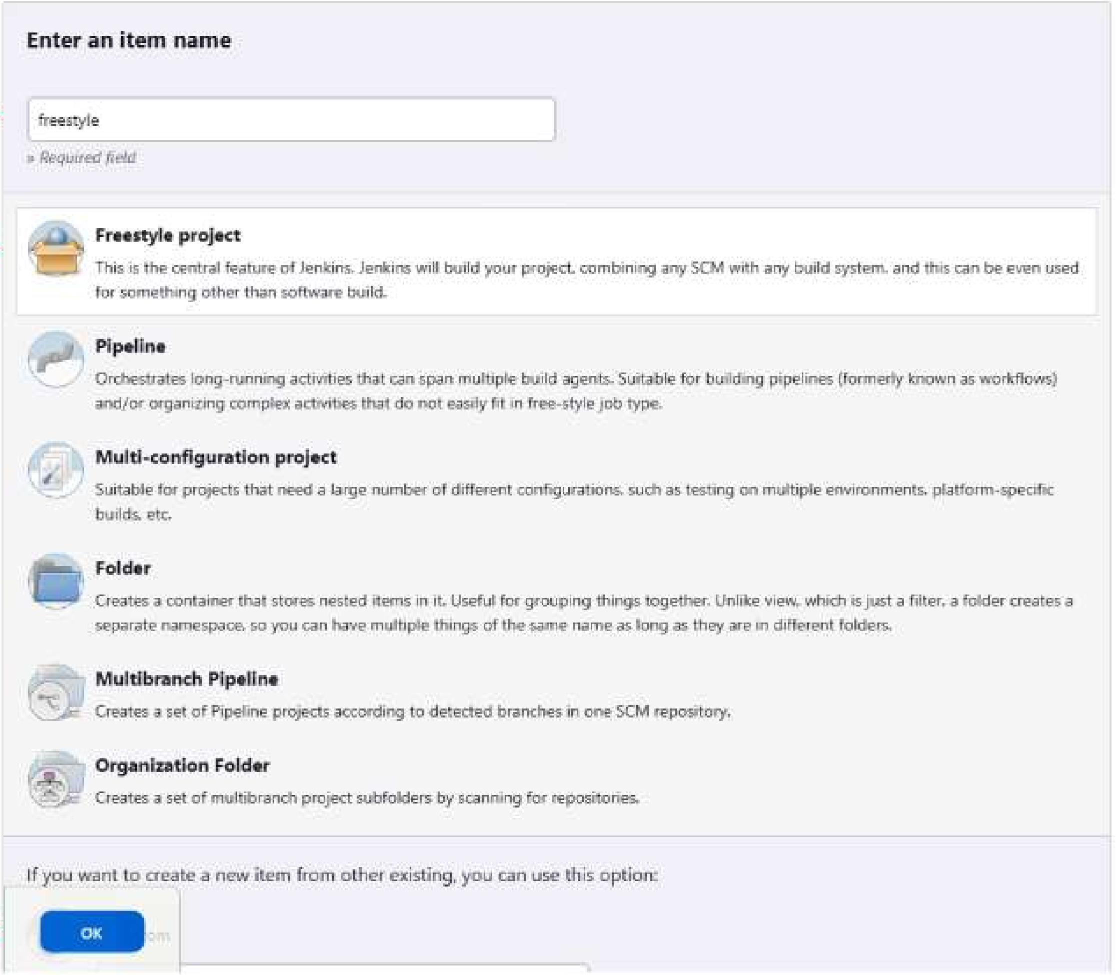
- ease of configuration.
- versatility.
- plug-in compatibility.
- SCM integration.
- post-build actions.

Freestyle jobs

Project name

Type of project

Click OK to build



Enter an item name

freestyle

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Freestyle job configuration

Once you have created a new job, you will then be redirected to a configuration page for that job.

Source Code Management:

- This section is used for configuring a source code repository to download.
- The job will download the repository you provide into the job's workspace.

Build Triggers:

- The easiest way to run a Jenkins job is by pressing the build button for the job.
- However, jobs can be triggered in many ways.

Freestyle job configuration

- **Build Environment:** The build environment allows us to configure various options for a job's environment.
- **Delete Workspace Before Build Starts:** The folder where the job runs on the host machine's file system will be deleted before building again. You will likely want this option checked!
- **Secret Texts & Files:** You may securely use secret texts and files that you have configured in the Credentials section here in the job. These secrets will also be hidden in the Jenkins logs as well.

Freestyle job configuration

- **Build:** The part of the job that is 'executed'. Functional steps of the job are configured here. This is likely where you will spend most of your time on a Jenkins job. The most common build step here is Execute shell. Other options are available, depending on what plugins you have installed.
- **Post-build Actions:** Events to trigger once a build has been completed. You may want to configure your job to react depending on how the build went. For instance, you could be notified by email if the job failed. You could also publish a report if the job completed successfully.

Freestyle job exercise

Using a Jenkins freestyle project, create a build which does the following:

- prints out all files in location (including hidden ones).
- creates a file called ‘coolScript.sh’.
- add an echo command to the coolScript.sh.
- get Jenkins to run the coolScript.sh.
- archive the coolScript.sh using a post build action.
- find your artefact in the jobs directory within the VM’s Jenkins files.

Jenkins plugins

Jenkins plugins: Module 7
Jenkins fundamentals

Jenkins plugins

Jenkins has hundreds of plugins that can be used to increase the functionality of the automation server. Alongside this, developers can write their own plugins, which means the use of Jenkins is almost limitless.

Some of the reasons why we might want to use plugins include:

- **Source Code Management (SCM):** Integrating SCMs like Git into Jenkins.
- **Build steps:** Providing everything from a convenient UI to configure build tools, to sending emails post-build.
- **Authentication realms:** Integrating Jenkins with Single Sign-On systems, or external user directories (such as Azure Active Directory).

Plugins exercise

Objective: Create a Jenkins Maven job to build a simple Java project.

Task:

- Create a Maven project in a GitHub repository with a simple Java application, e.g. printing "Hello, World!".
- In Jenkins, create a new Maven Project.
- Configure the job to fetch the code from your repository and build it using the predefined Maven goals.

Pipeline jobs

Pipeline jobs: Module 7

Jenkins fundamentals

Jenkins pipeline jobs

Jenkins Pipelines provide a robust framework for managing continuous integration and continuous delivery (CI/CD) processes:

- **Pipeline-as-Code:** Define CI/CD workflows in a script, ensuring easy version control and sharing.
- **Consistent Workflow:** Introduce an automated, consistent path from code submission to deployment.
- **Parallel Execution:** Execute tasks concurrently to optimise resource usage and reduce build times.
- **Visualisation:** Use the built-in stage view for a clear, graphical representation of pipeline execution, aiding troubleshooting and analysis.

Jenkins pipeline jobs

Setting up a Jenkins Pipeline involves key steps, ensuring a solid foundation for CI/CD automation:

- **Pipeline Creation:** Choose 'New Item' from the Jenkins Dashboard and select 'Pipeline'.
- **Defining the Pipeline:** Define the pipeline directly in Jenkins or by linking to a Jenkinsfile in your source code repository.
- **Source Code Management (SCM):** Select your SCM. Input repository details and credentials to establish a connection.
- **Staging and Steps:** Organise tasks into stages and steps, ensuring a logical flow in the pipeline execution.

Jenkinsfile

```
pipeline {
agent any
  stages {
    stage('Build') {
      steps {
        //
      }
    }
    stage('Test') {
      steps {
        //
      }
    }
    stage('Deploy') {
      steps {
        //
      }
    }
  }
}
```

Pipeline: A user defined model of a CD pipeline. A pipeline's code defines the entire build process.

Agent: Specifies where the pipeline should be executed.

Stage: A stage block defines a conceptually distinct subset of tasks.

Steps: A task that Jenkins should run.

Webhooks

Webhooks are pivotal in actualising real-time communication between Jenkins and code repositories, enabling automated triggers for pipeline executions.

- **Immediate Notification:** Webhooks notify Jenkins instantly upon code changes.
- **Synchronisation:** Ensure a synchrony between code pushes to SCM and Jenkins build processes, maintaining up-to-date build and deployment cycles.
- **Configuration in Jenkins:** Utilise available plugins like GitHub Webhook to configure Jenkins to listen for incoming webhook payloads.
- **Repository Setup:** Configuring webhooks within the SCM, specify the Payload URL pointing to your Jenkins server, ensuring secure and reliable communication.

Pipeline job exercise

Objective: Create and configure a Jenkins Pipeline job that builds and tests a simple Java application, using a webhook for CI/CD automation.

Task:

- Enhance your existing Java application (from the previous exercise) by adding a unit test using a testing framework (e.g. JUnit).
- Create a Jenkinsfile in your project repository with the appropriate pipeline script to build and test your Maven project.
- Configure a webhook in your version control platform (e.g. GitHub) to trigger the Jenkins Pipeline job upon each push to the repository.
- Ensure the pipeline job builds successfully and runs the unit test whenever code is pushed to the repository.

Summary

Jenkins: Module 7

Understand Jenkins and CI/CD

- Understand common phraseology used with Jenkins and CI/CD servers and how to install.

Explore builds and plugins

- Build an application through a Jenkins freestyle job Integrating SCM, build tools, and plugins.

Thank you for listening

Any questions?