# **Jenkins lab 1**

# **Part 1 – Explore IDE and create a Jenkin job**

### **Objectives**

* In this part you will install Jenkins using Docker technology.
* Investigate the Jenkins IDE.
* Create a basic Job and explore its options

1. Create a folder
2. Open the folder in VS-Code
3. Create a file called docker-composed.yml
4. Copy the following to the yml file

**services:**

**jenkins:**

**container\_name:** jenkins-docker-container

    image: jenkins/jenkins:lts

    ports:

      - "8080:8080"

    networks:

      - ecommerce-network

**networks:**

  ecommerce-network:

    name: ecommerce-network

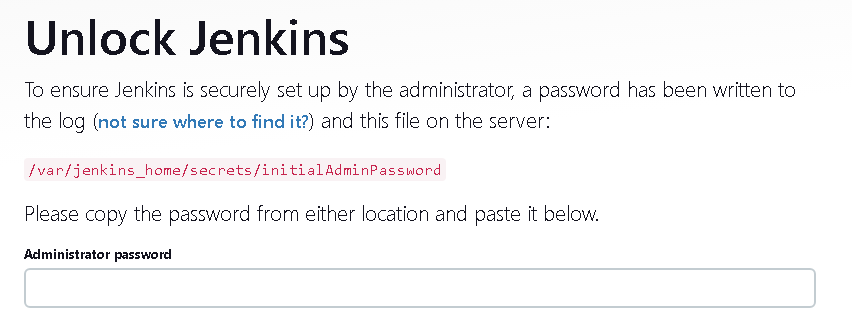
    external: true

Here is a picture showing the correct indentation

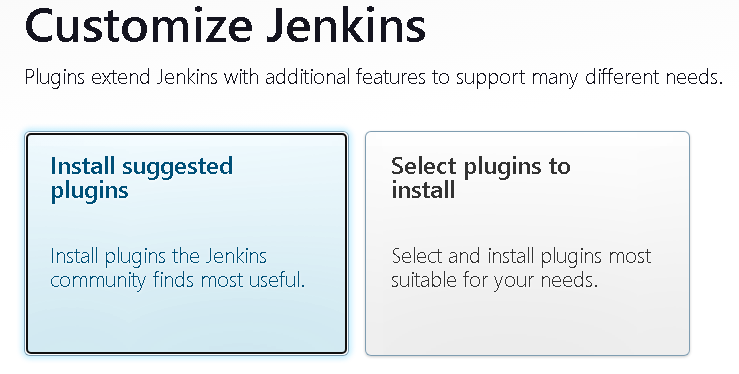
A screen shot of a computer

Description automatically generated

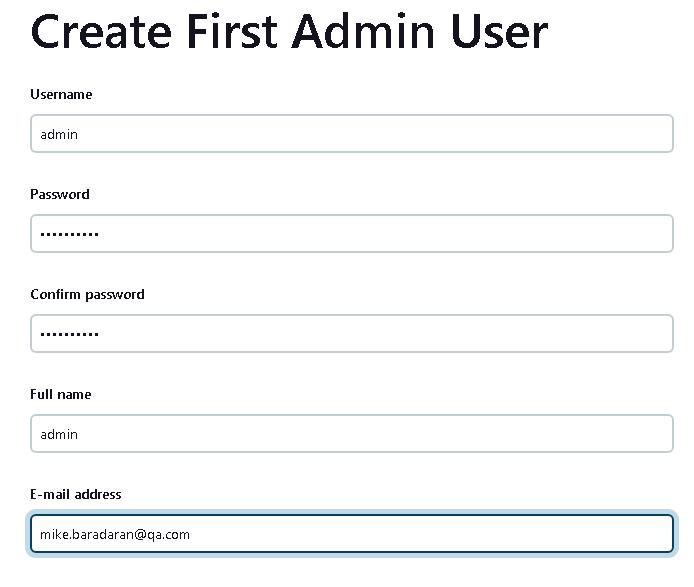
1. Test your file by typing the following command in a Terminal window: **docker-compose config**  
   This command will type the contents of the file if it is valid.
2. Run the command: **docker-compose up -d**And then wait for a minute. This command will install Jenkins in detached and will not echo the secret password which you need to start Jenkins.
3. Get the Jenkin’s docker ID (**docker ps**). In this example we assume the ID starts with **e7b**
4. Run the following command to get the Jenkins secret password:  
   **docker exec -it e7b cat** /var/jenkins\_home/secrets/initialAdminPassword  
   copy the secret password
5. In a browser type: **localhost:8080**Copy the secret Administrator passwordfrom step 10



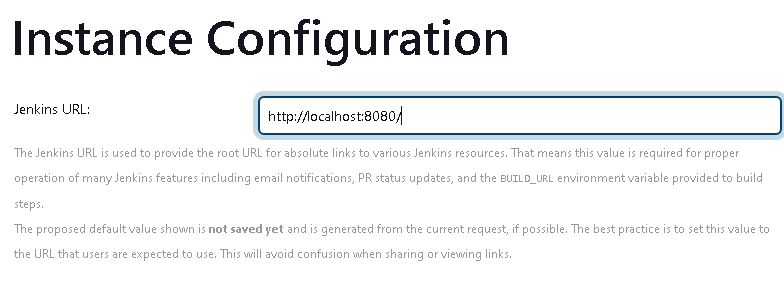
1. Install the suggested plugins

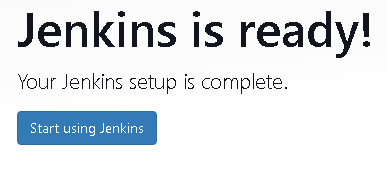


1. Create the admin user (any name and password will do. The email is not checked)



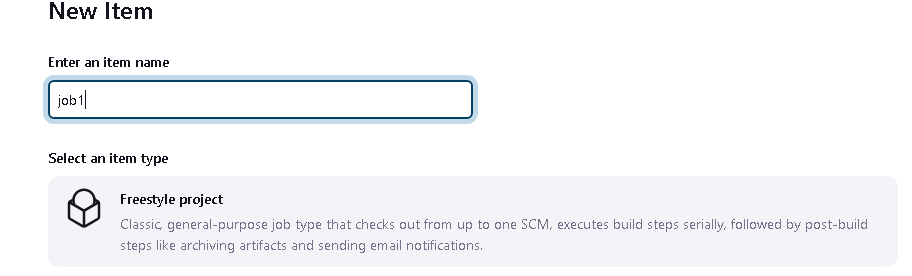
1. Follow the rest of the dialogs



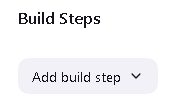


1. Create a new job by clicking [**+ New Item**]  
   A screenshot of a computer

   Description automatically generated
2. Choose a name for this job



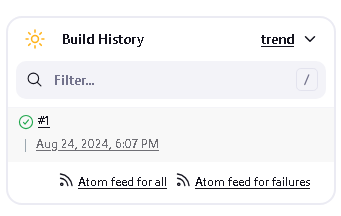
1. Scroll down and find the Buil Step section and then select these options:

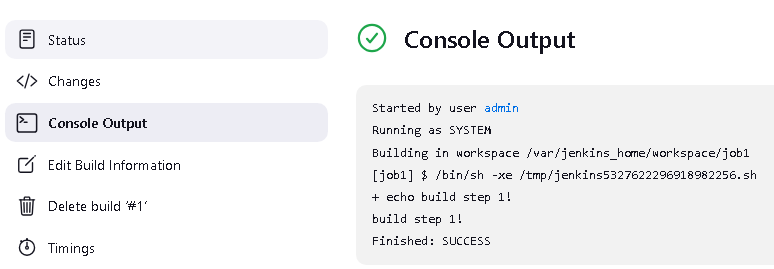


We will not deploy an application, only a simple script to get to know this tool.

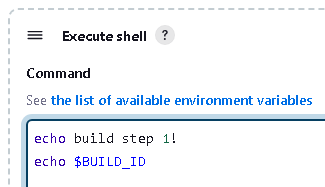
1. Click **Save** and then 



1. Click on #1 (build number) and then click 

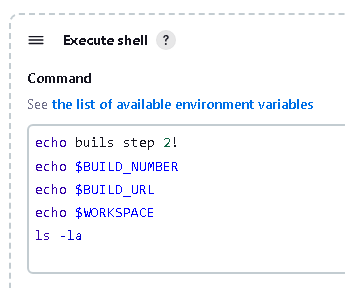


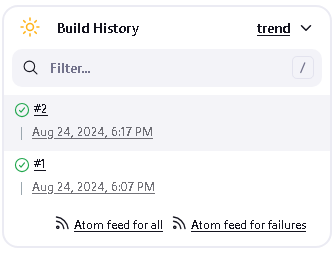
1. Jenkins provides many environmental variables. We will have a look at a few. View these by clicking on the  link.
2. Click on **job1** title and then select the configure option



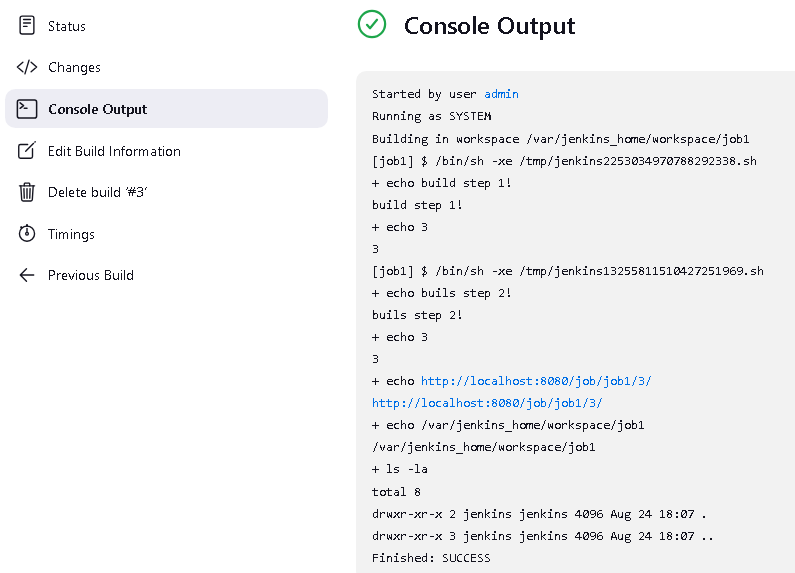
1. Click A close-up of a sign

   Description automatically generated
2. View the Console log and note the BUILD\_ID value.
3. Modify Job1 by adding a few more lines

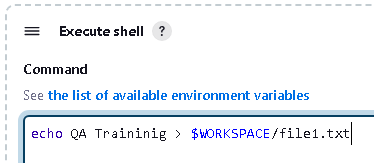
(Execute shell) 



1. View the console log

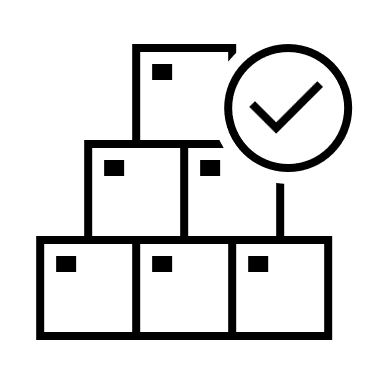


1. Add another step to the build to execute a shell command to reference Jenkin’s Workspace  
   which is a directory on the Jenkins server where the files related to a specific job are stored during the build process. The following command create a file called file.txt with content of “QA Training”.

1. Lokk for the Job 1’s Workspace menu and display the content of file.txt



Congratulations, you have successfully created a job using Jenkins and explored it’s various IDE options.  
 Please continue to **Part 2** below.

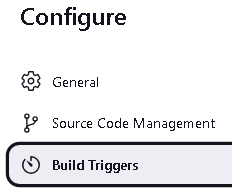
# **Part 2- Triggers**

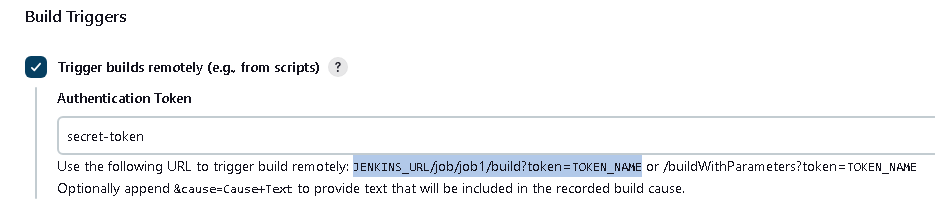
## **Objectives**

In this part you will create a Jenkin **trigger**, and explore other forms of triggers in other labs.

**Trigger** is an event or action that initiates the execution of a job or pipeline (we will explore pipelines later).

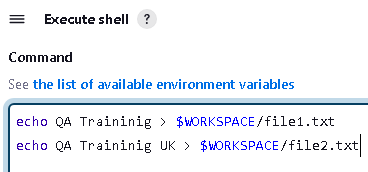
## **Create a Remote build trigger**

1. Configure Job1 and find the Buil Triggers section  
   
2. Create a token for invoking and API end point for triggering a build (any secret token word will do)

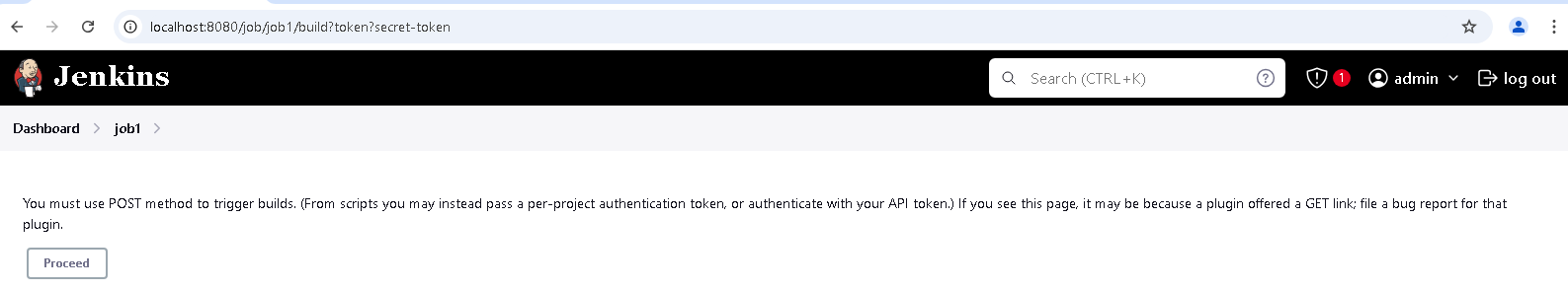




1. Make a change to the last build and then save.

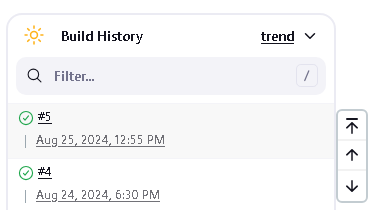


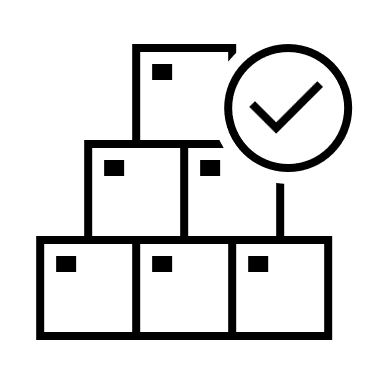
1. Type the following text in a browser and click the Proceed button to trigger the build.   
   Please note, this will result in a blank screen and will only shows errors if there are any.



http://localhost:8080/job/job1/build?token?secret-token

1. As mentioned, the screen will go blank but a new build is triggered (please wait a moment until it does)



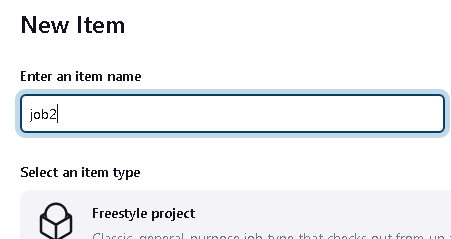
Congratulations, you have successfully created a Jenkins trigger. This is an important concept and you will explore other forms of Jenkin’s Triggers in other labs.  
 Please continue to **Part 3** below.

# **Part 3 - Chaining project builds**

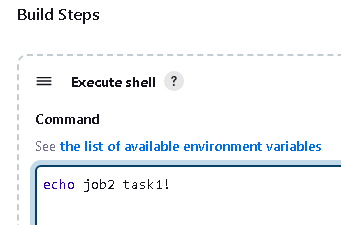
# **Objectives**

In this part you will chain 2 or more builds together.  
Build chaining in Jenkins automatically triggers one or more jobs after another job completes. This enables multi-step automation pipelines, where one job’s output becomes the next job’s input.

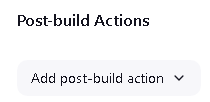
1. Create a new Job called **job2**

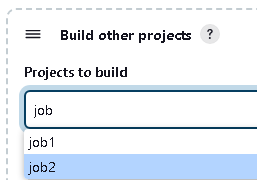


1. Assign a build step like

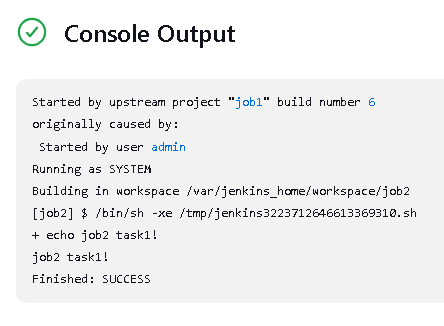


1. Configure Job1’s post-build action to trigger building Job2

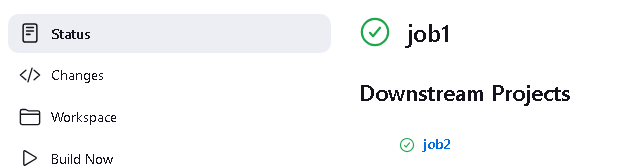


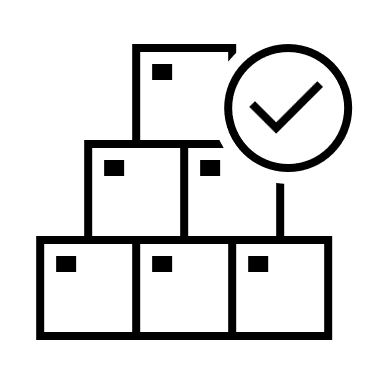
 

1. Build Job1  and note that both jobs get built  
   View the console output of Job1 and Job2 for proof



1. View Job1’s status for the Downstream project (Job2)



Congratulations, you have successfully created a Jenkins build chain.  
 Please continue to Lab2 to explore building and deploying a Maven app stored on GIT.