Creating job in Jenkins

Introduction :

This Documentation provides step by step guide for Creating Jobs in Jenkins. Creating a job on Jenkins is a fundamental step in setting up automated builds, tests, and deployments for your software projects.

Prerequisites for Creating Job:

- ✓ Jenkins Installation (For Installing refer Git-link)
- ✓ Access Credentials
- ✓ Plugins
- ✓ Build Environment
- ✓ Access to a web browser

What Is a Build Job?

A Jenkins build job contains the configuration for automating a specific task or step in the application building process. These tasks include gathering dependencies, compiling, archiving, or transforming code, and testing and deploying code in different environments.

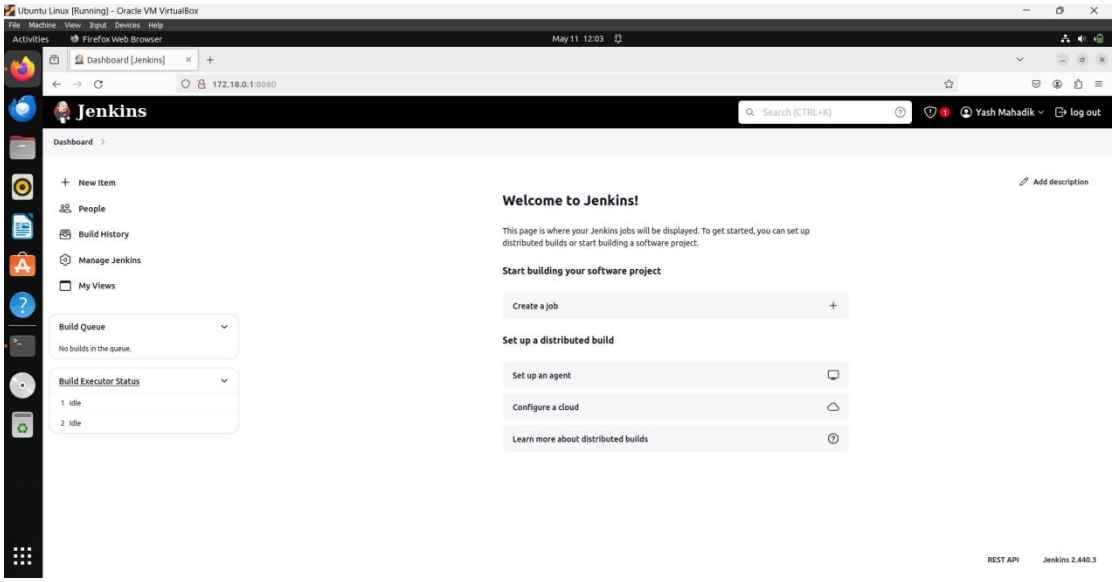
Jenkins supports several types of build jobs, such as freestyle projects, pipelines, multi-configuration projects, folders, multi branch pipelines, and organization folders.

What is a Jenkins Freestyle Project?

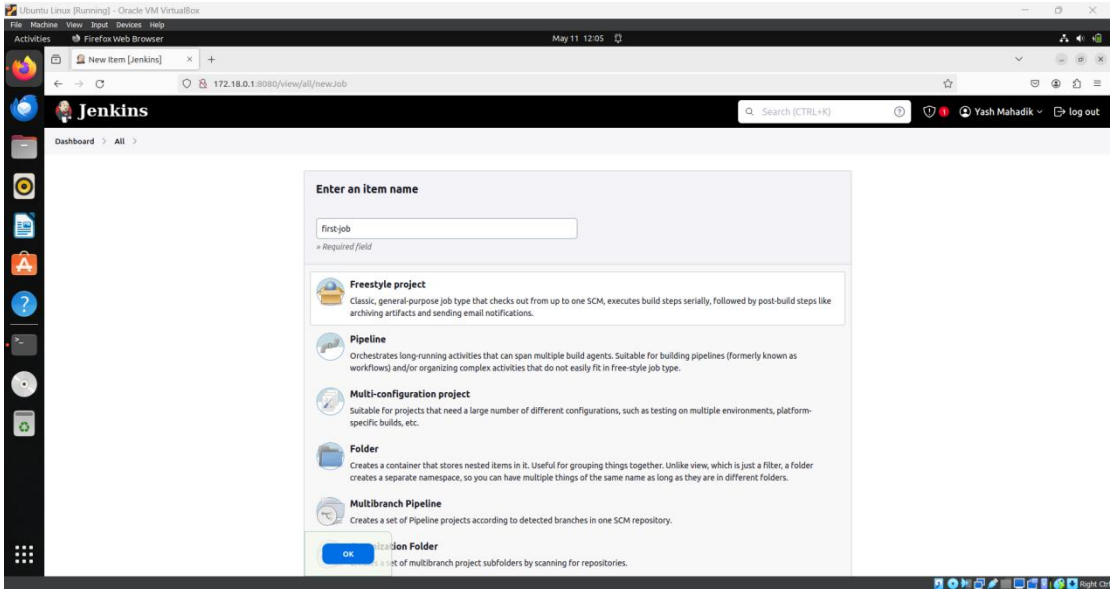
Jenkins freestyle projects allow users to automate simple jobs, such as running tests, creating and packaging applications, producing reports, or executing commands. Freestyle projects are repeatable and contain both build steps and post-build actions.

A step-by-step guide to Set up a Build Job in Jenkins

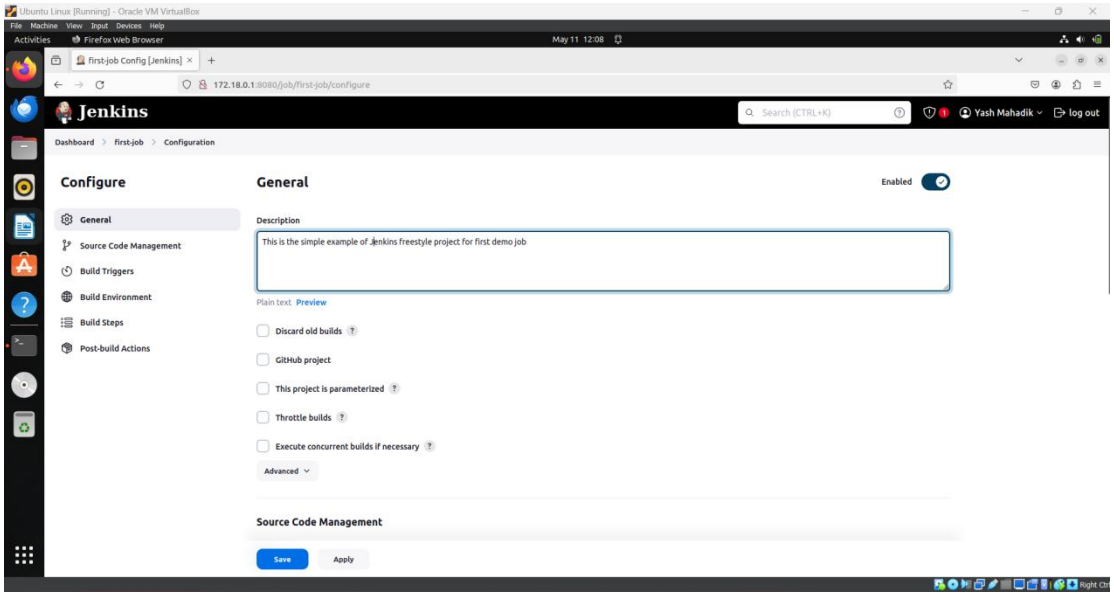
1) Click the New Item link on the left-hand side of the Jenkins dashboard.



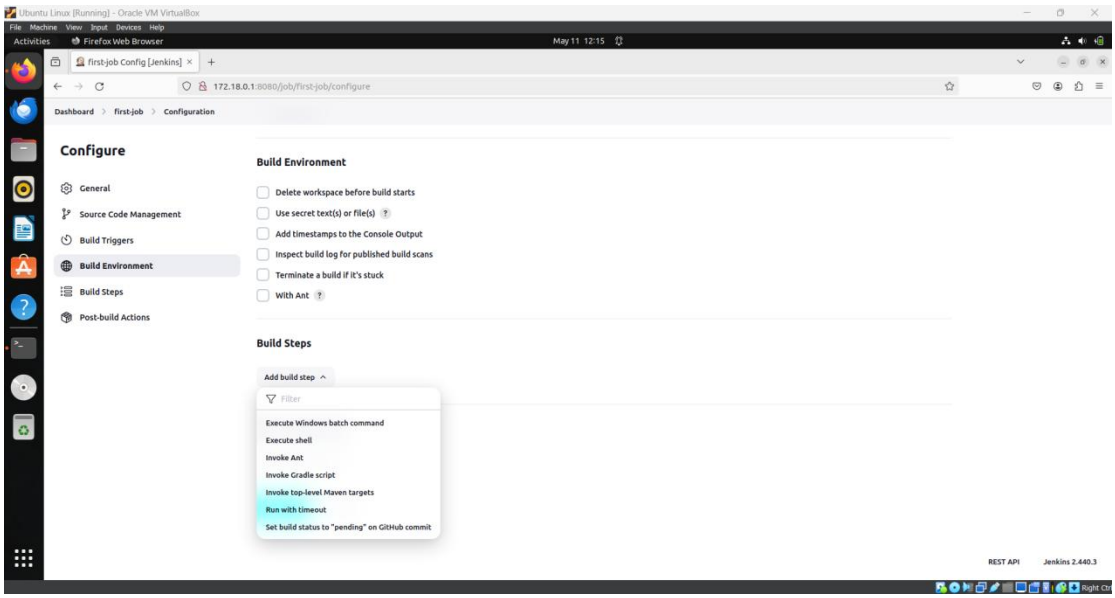
2) Enter the new project's name in the Enter an item name field here we have given name as first-job and select the Freestyle project type. Click OK to continue.



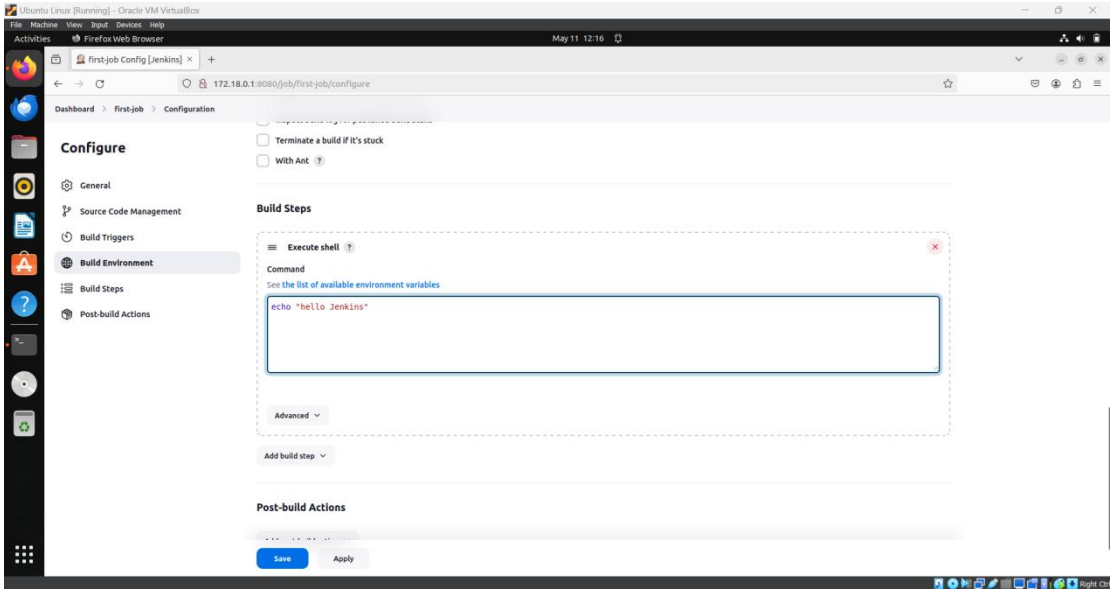
3) Under the General tab, add a project description in the Description field which is optional. Here we have given simple example of Jenkins freestyle project for first demo job.



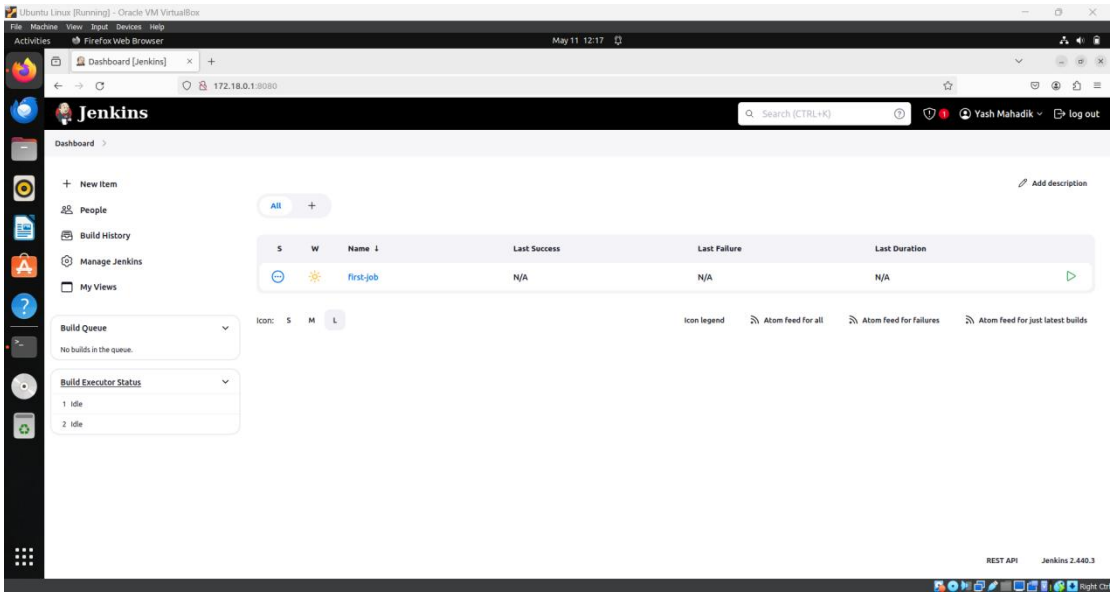
4) Scroll down to the Build section and open the Add build step drop-down menu and select Execute Shell (you can select any Build as per your OS)



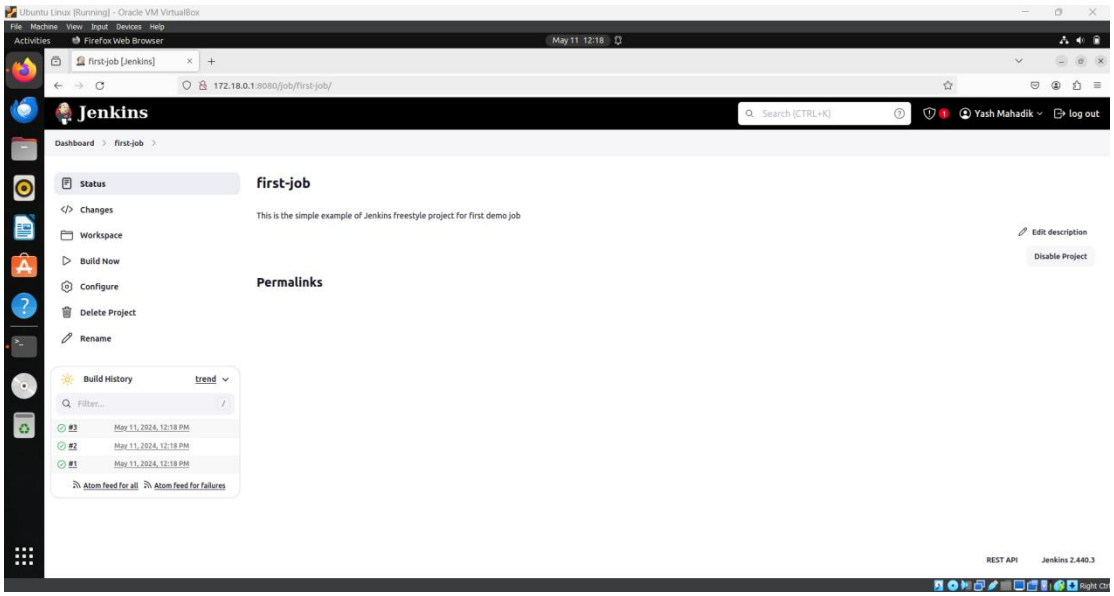
5) Enter the commands you want to execute in the Command field. For this, we are using a simple set of commands that display **hello Jenkins** using **echo "hello Jenkins"**



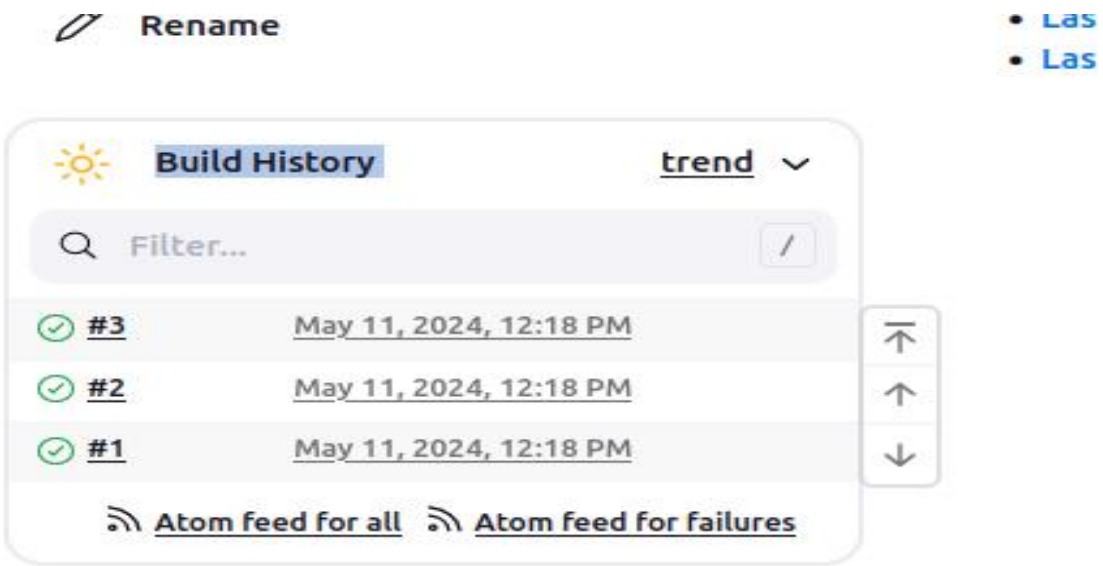
6) Click the Save button to save changes to the project.



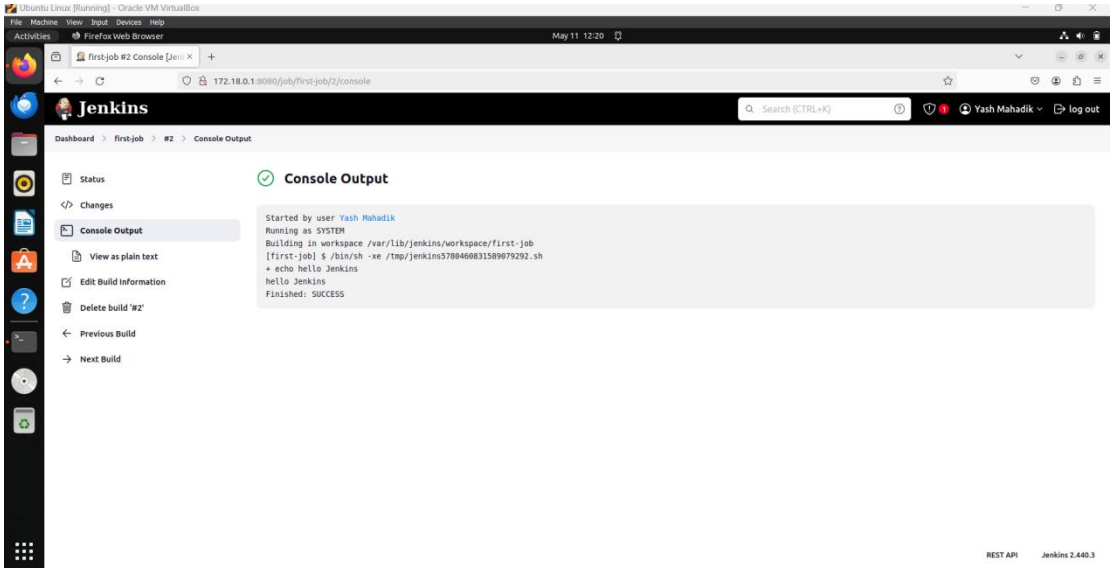
7) Click the **Build Now** link on the left-hand side of the new project page



8) Click the link to the latest project build in the Build History section. You can see you job has been Build successfully.

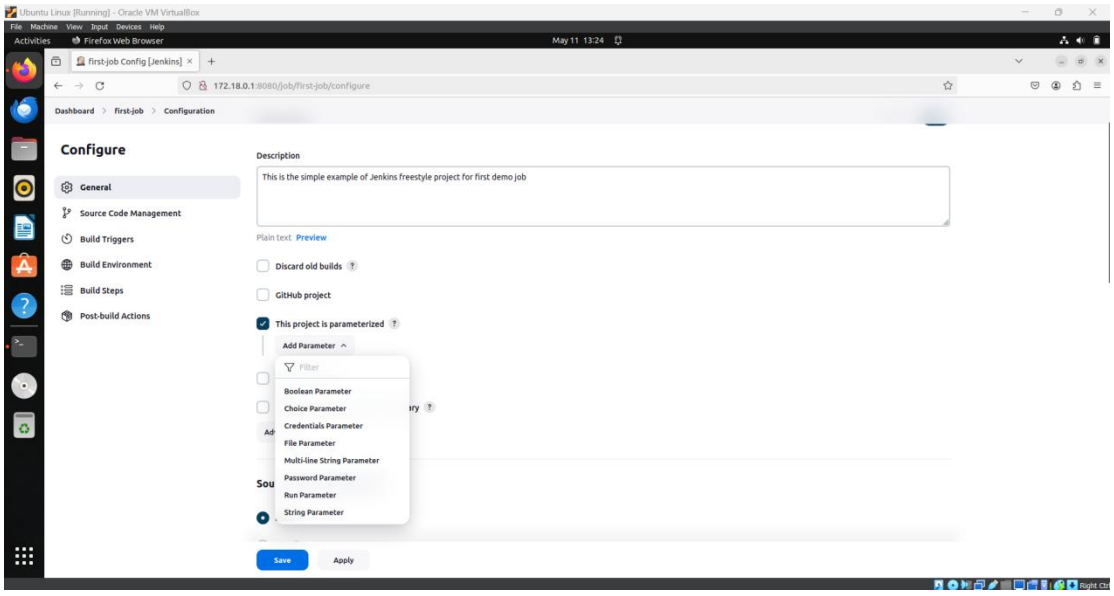


9) Click the **Console Output** link on the left-hand side to display the output for the commands you entered. You can see **hello Jenkins**. So here we Create out first job and built it successfully.

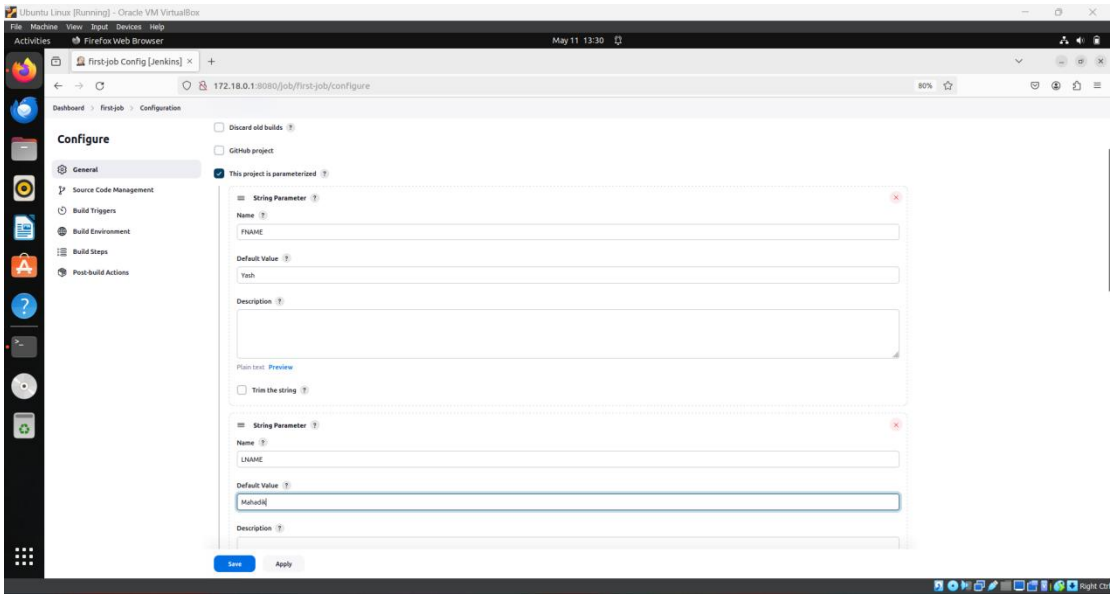


We have Created our first job and so let see how we can parameterized the job.

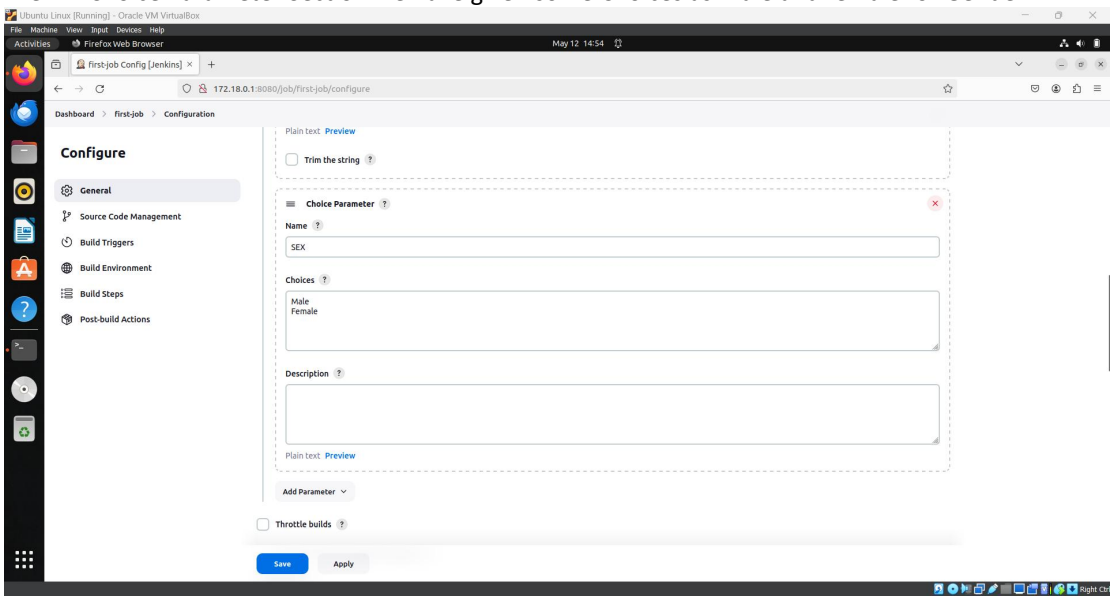
1) Select the Jenkins job for which you need to add parameter. For example “first-job” in bellow screenshot and the Click on configure , again Description section is optional. In general section you will see “t**his project is parameterized**” tick mark on it. Click on Add Parameters. There are multiple options for now we will use “**String Parameter**” from the options and “**Choice Parameter”**.



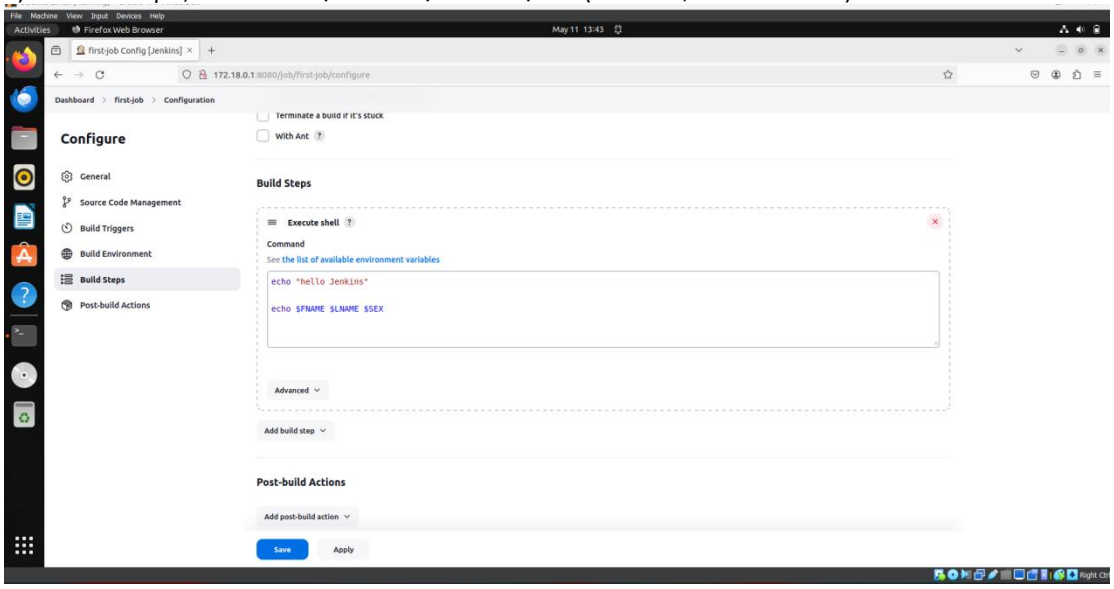
2) Add the variable name as per your requirement for below example I am using “**FNAME**” as variable name and another variable name as “**LNAME**” with some default value which can we changed later.



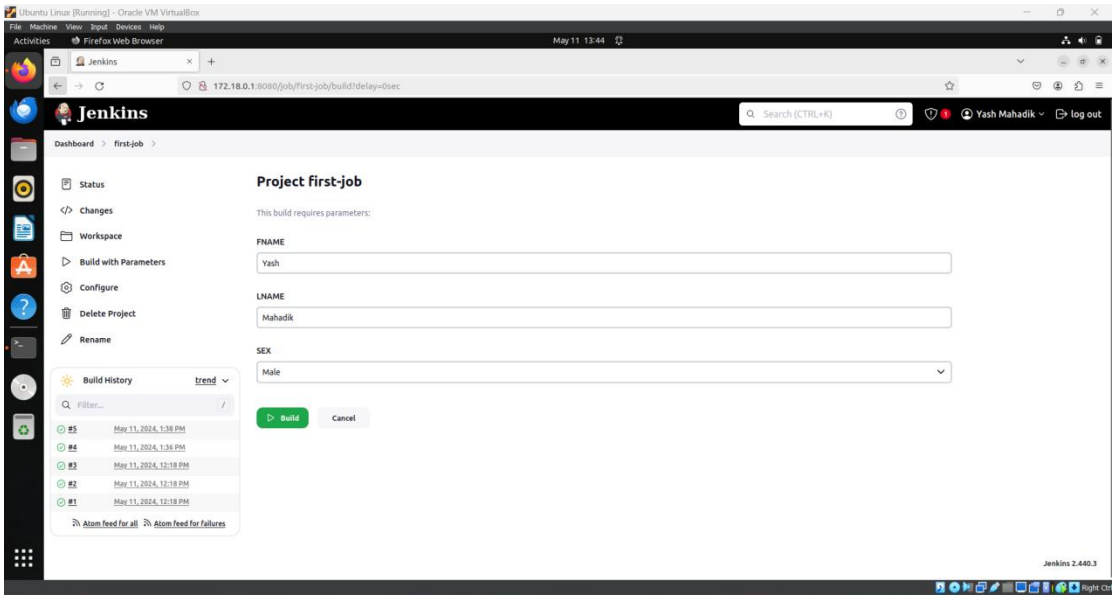
Even in Choice Parameter Section we have given some Choices as Male and female for Gender



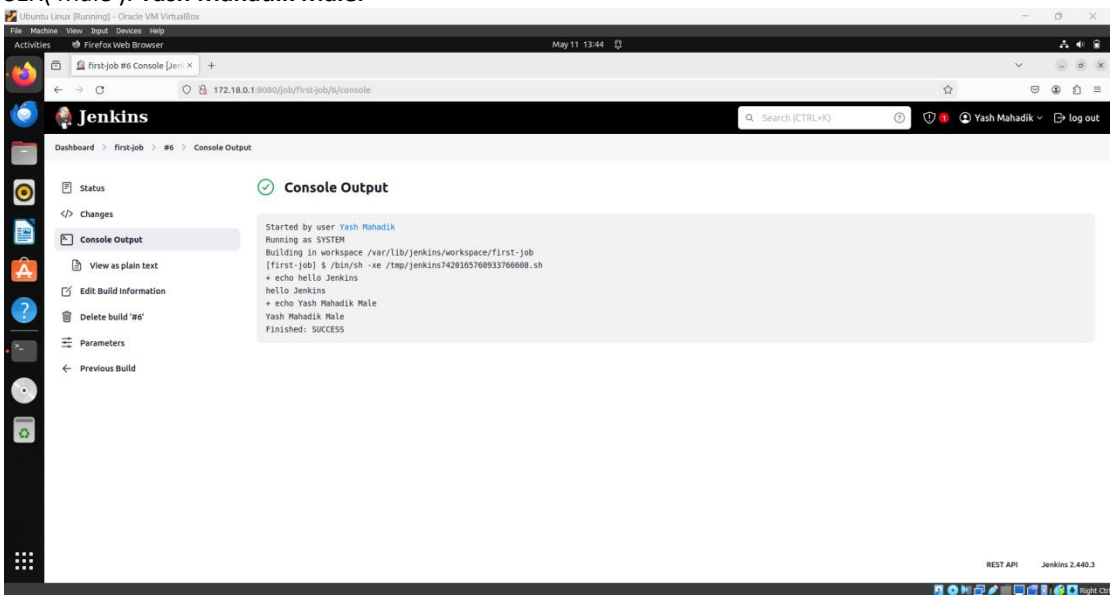
3) In Build Steps , Enter **echo $FNAME $LNAME $SEX** (we use $ in Linux shell). Click on Save.



Before Building the job we can change the values for our Variables which is a good way. Click on Build with Parameters as we have include some parameters to the job.

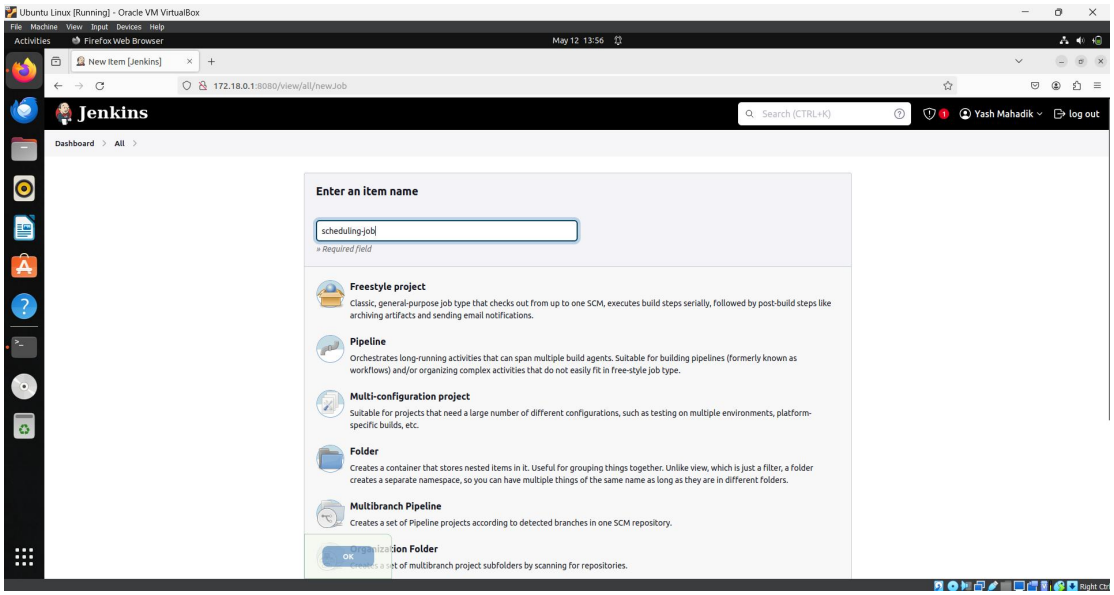


4) In Console Output you can see the Parameter values FNAME(Yash), LNAME(Mahadik), SEX(Male). **Yash Mahadik Male.**

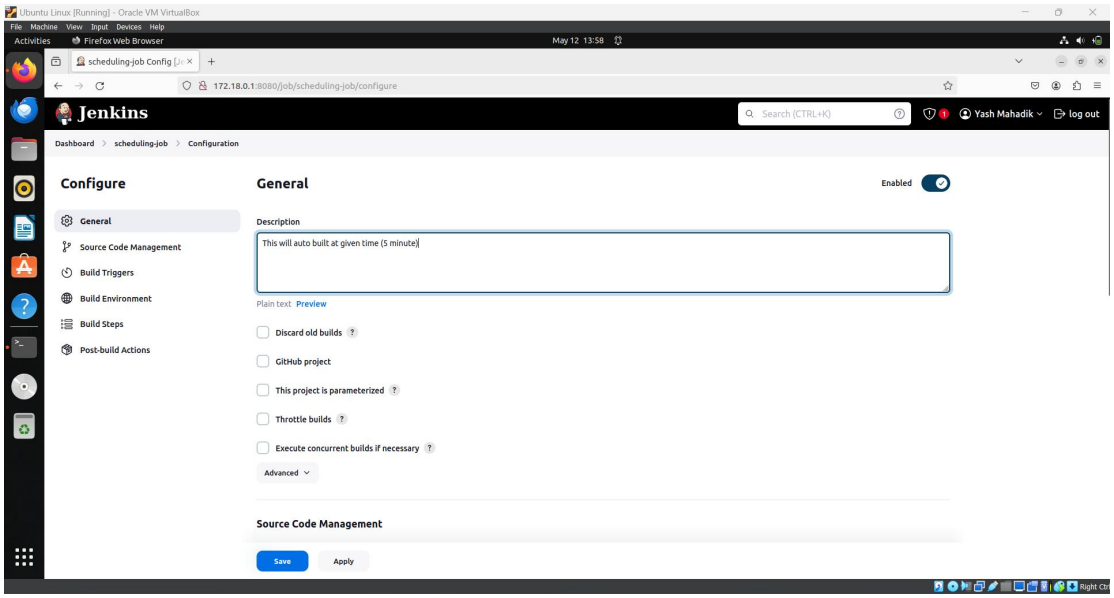


Creating a Simple Scheduling Job

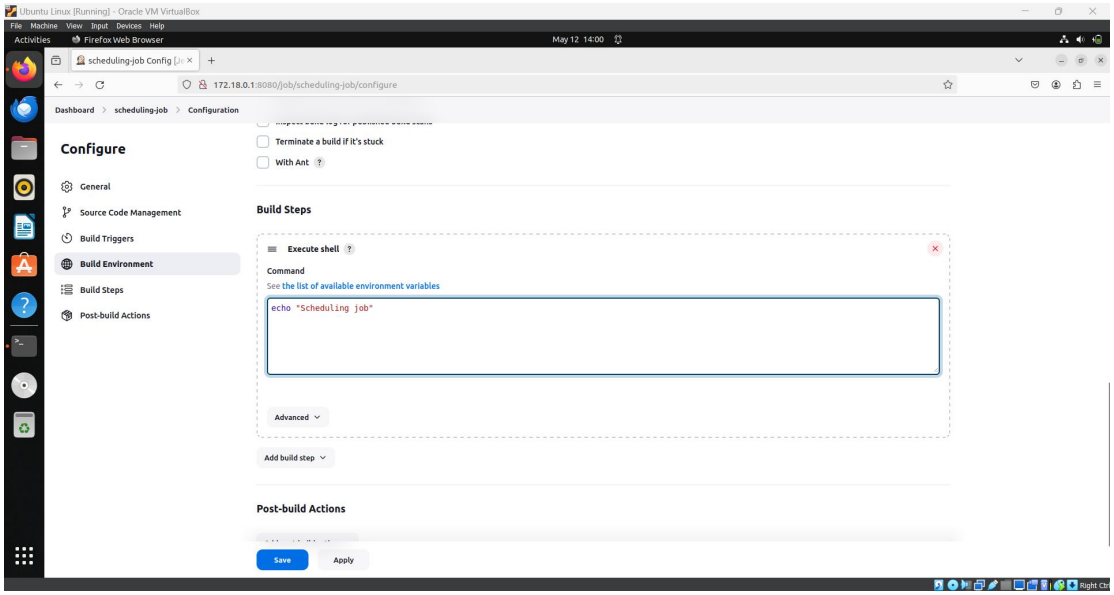
1) Start by creating a new item. Give your new item a name as Scheduling-job. Select Freestyle Project and Click OK, and Jenkins will create your empty job.



2) Description section is Optional.



3) In Build shell , select Execute shell and run **echo "Scheduling Job"** as an example.



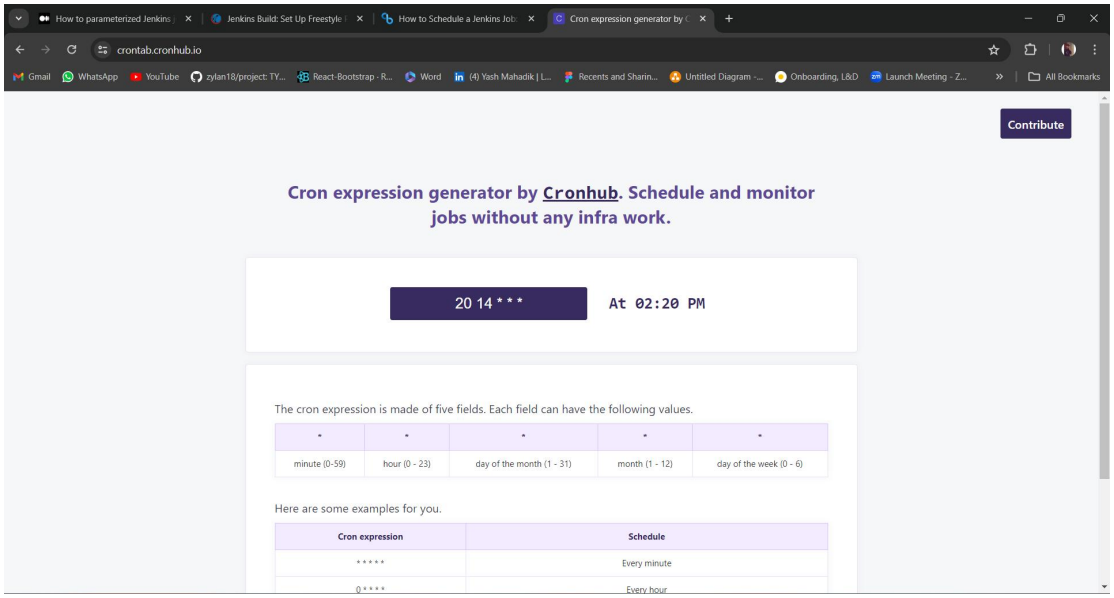
4) Let's look at the Jenkins job scheduling configuration. It looks a lot like Linux's cron syntax, but you don't have to be familiar with command line Linux to figure it out.
A scheduling entry consists of five white space-separated fields. You can schedule a job for more than one time by adding more than one entry

Minute	Hour	Day of Month	Month	Day of Week

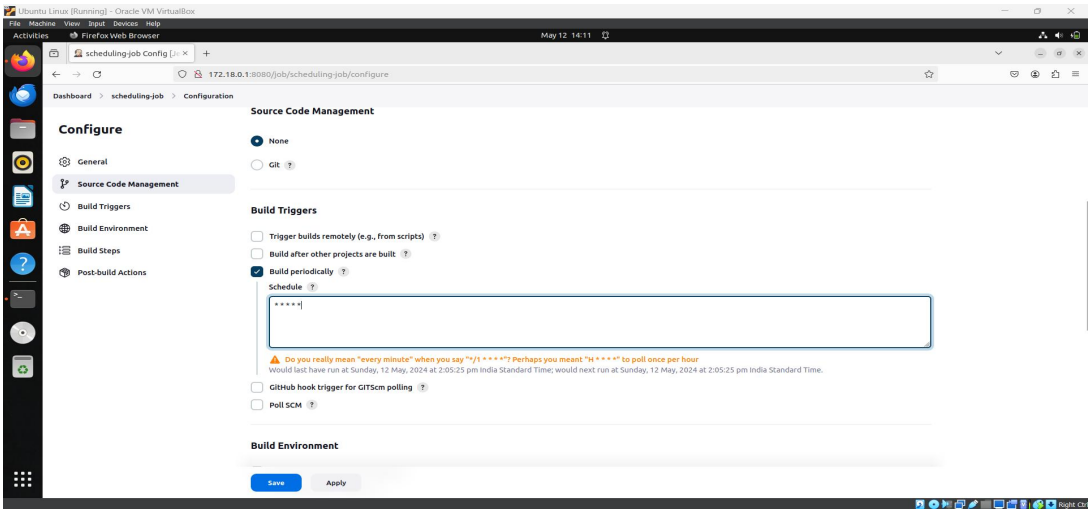
Each field can contain an exact value or use a set of special expressions:

- ✓ The familiar asterisk * indicates all valid values. So, a job that runs every day has a * in the third field.
- ✓ A dash separates ranges of values. For example, a job that runs every hour from 9:00 a.m. to 5:00 p.m. would have 9-17 in the second field.
- ✓ Intervals are specified with a slash /. A job that runs every 15 minutes has H/15 in the first field. Note that the H in the first field has a special meaning. If you wanted a job to run every 15 minutes, you could configure it as 0/15, which would make it run at the start of every hour. However, if you configure too many jobs this way, you can overload your Jenkins controller. Ultimately, the H tells Jenkins to pick a minute based on a hash of the job name.
- ✓ Finally, you can specify multiple values with a comma. So, a job that runs Monday, Wednesday, and Friday would have 1,3,5 in the fifth field.

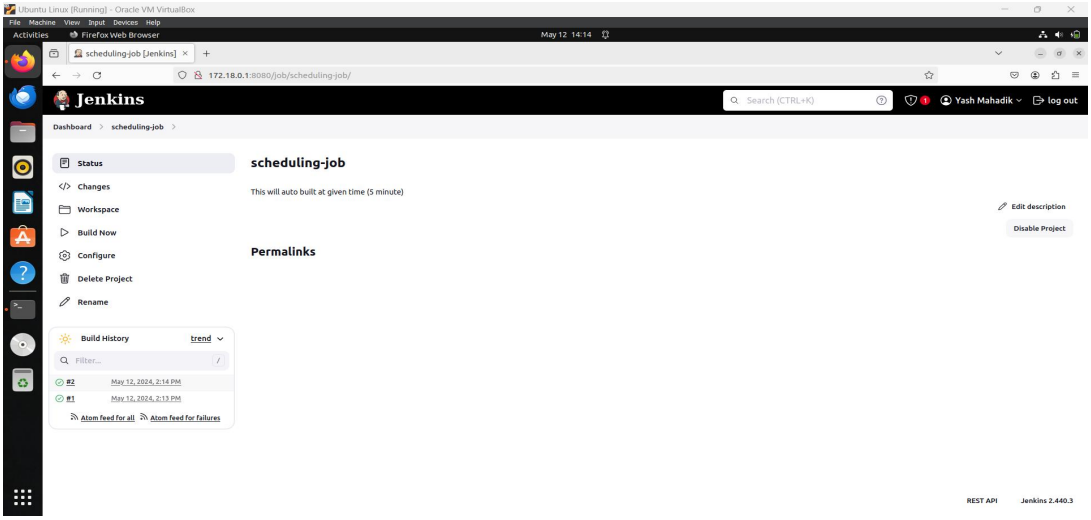
You can refer https://crontab.cronhub.io/ for scheduling job.

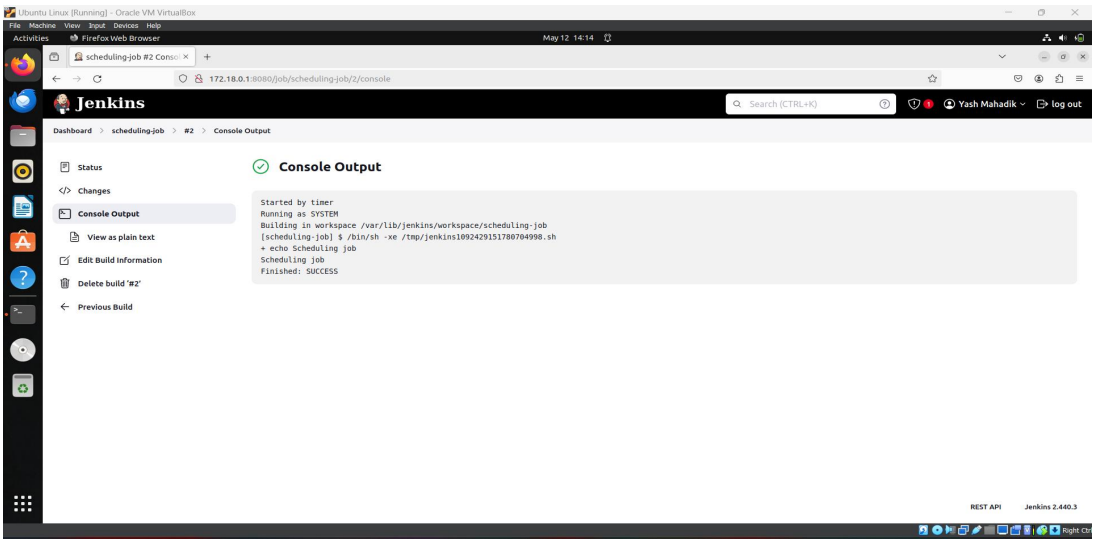


5)Head back to the job configuration and click the **Build Triggers** tab and select **Build Periodically** and schedule the job (* * * * *) indicates build the job every 1 minute of time interval. And click on Save.



6)In Build History Section You will see at every 1 minute a job has been build successfully





Conclusion : By following the outlined process and ensuring the necessary prerequisites are in place, you can leverage Jenkins to streamline your development pipeline, increase efficiency, and maintain consistent build and deployment processes. n essence, creating jobs in Jenkins marks the beginning of a journey toward enhanced automation and continuous integration/continuous deployment (CI/CD) practices, ultimately contributing to the delivery of high-quality software with greater speed and reliability.