**­­­PROJECT-1**

**DEPLOYING WORDPRESS WEB APPLICATION THROUGH JENKINS USING EXECUTING SHELL AND DOCKER**

# INTRODUCTION

We'll deploy **WordPress** via **docker-composer** onto the AWS EC2 instance (**t2.micro**) and access it with a domain name defined in **Route53**.

WordPress is the easiest way to manage and create content. Its flexibility is loved by authors: with a couple of plugins, you can do everything from hosting a cute kitten’s photo gallery to hosting an e-commerce site.

Let’s face it: seen from the IT guy point of view, WordPress is a technical nightmare. When someone has to deal with it the horror begins: scalability is challenging, installation isn’t scripted and a LAMP stack is not always easy to maintain.

In this article, we’ll give you some technical hints and examples to ease your relationship with WordPress in a cloud environment based on AWS.

We'll try to use as many AWS-managed services as we can to be able to offload boring and dangerous tasks

# PRE-REQUISITES:

* Amazon Web Services
* Account GitBash Tool
* Jenkins Tool
* Github account

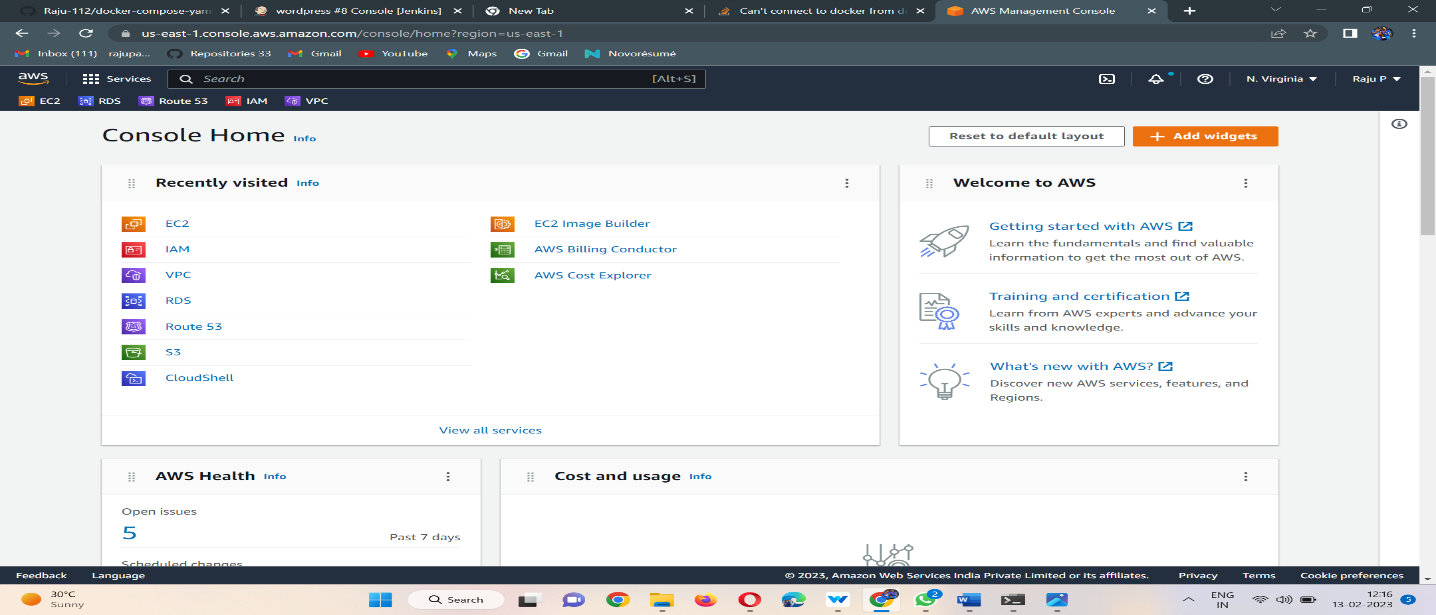
**What is Jenkins?**

Jenkins is a self-contained, open-source automation server that can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

# What is WordPress?

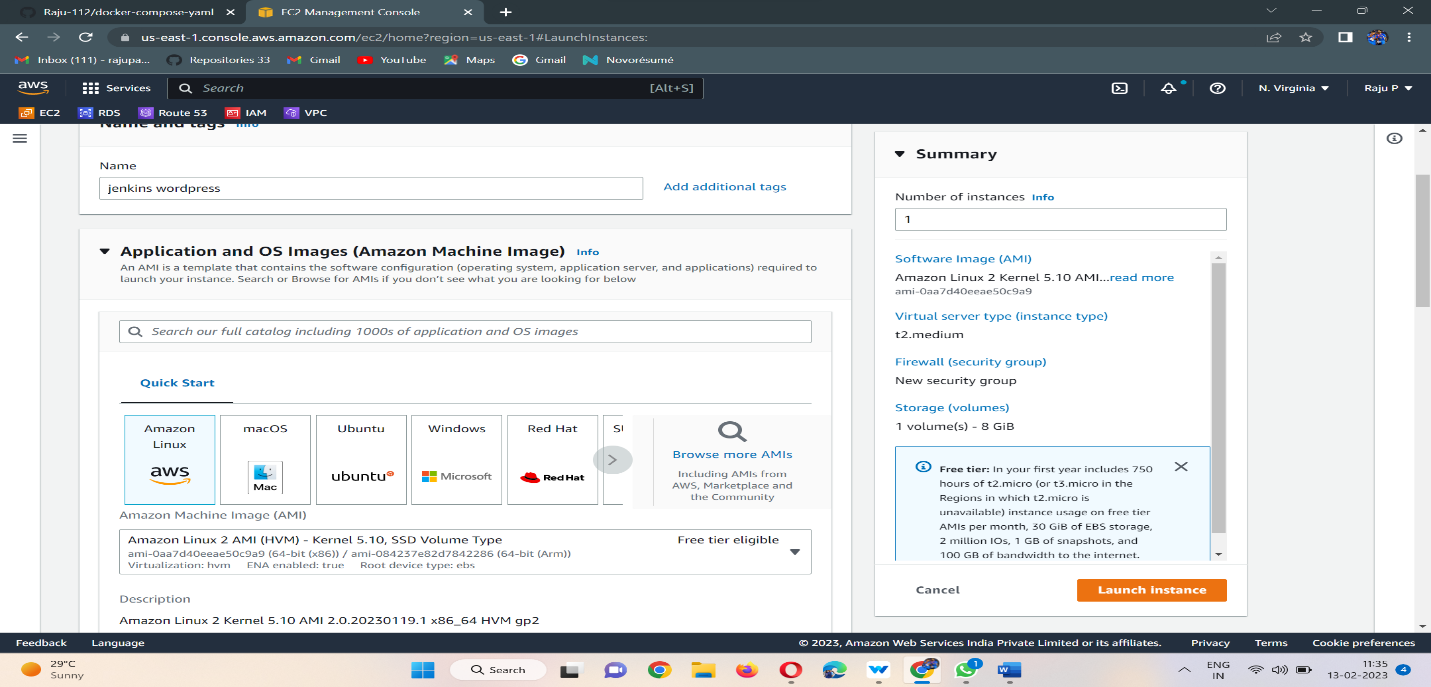
WordPress is a free and open-source content management system written in PHP and paired with a MySQL or MariaDB database with supported HTTPS. Features include a plugin architecture and a template system, referred to within WordPress as Themes.

­­­­**Module – 1: Now Creating and launching an Amazon Linux EC2 instance**



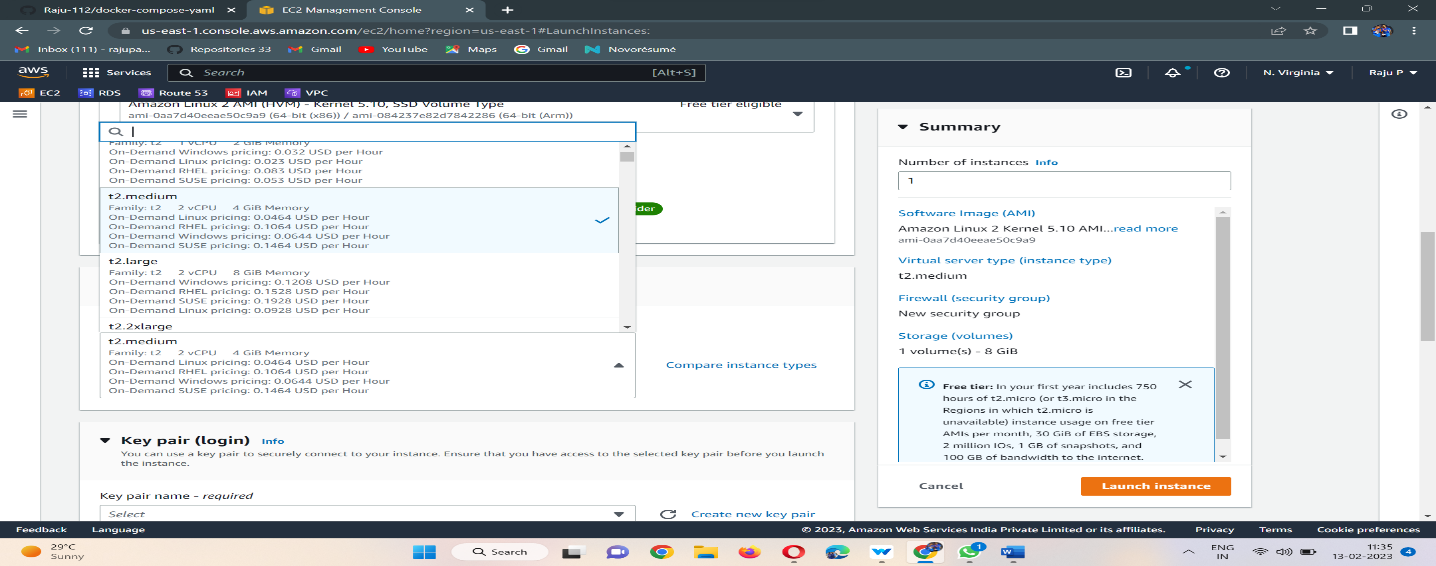
# 2. Choosing an AMI – Amazon Linux 2 AMI

* An AMI is a virtual image used to create a virtual machine within an EC2 instance.
* You can also create multiple instances using a single AMI when you need instances with the same configuration.
* You can also create multiple instances using different AMI when you need instances with a different configuration



**3. Choosing the instance type t2-medium**

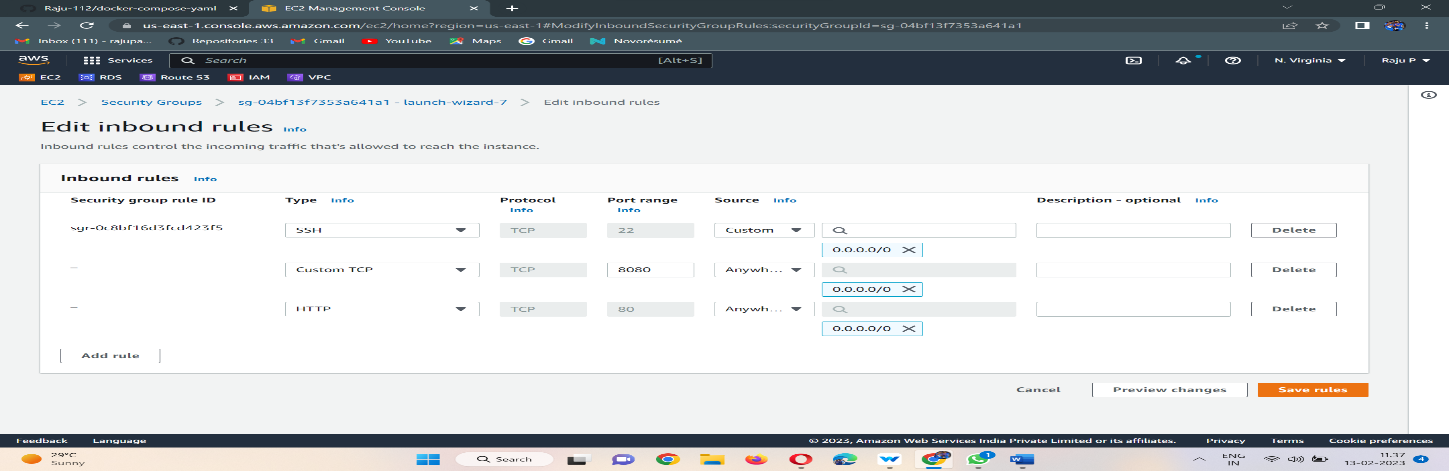
Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.



# 4 . Configure the Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below

Here, I am assigning the port numbers for the inbound rule SSH-22, HTTP-80, and 8080 All Traffic for the outbound rule.

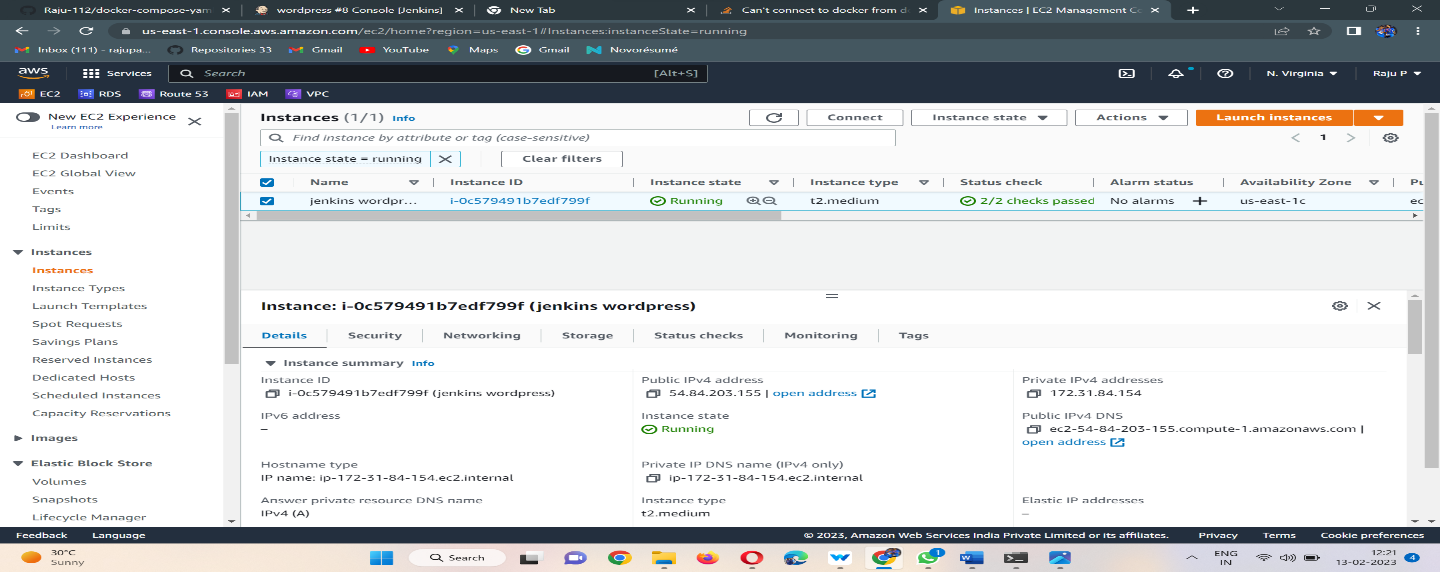


# 4. Select an Existing keypair or create a new keypair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.

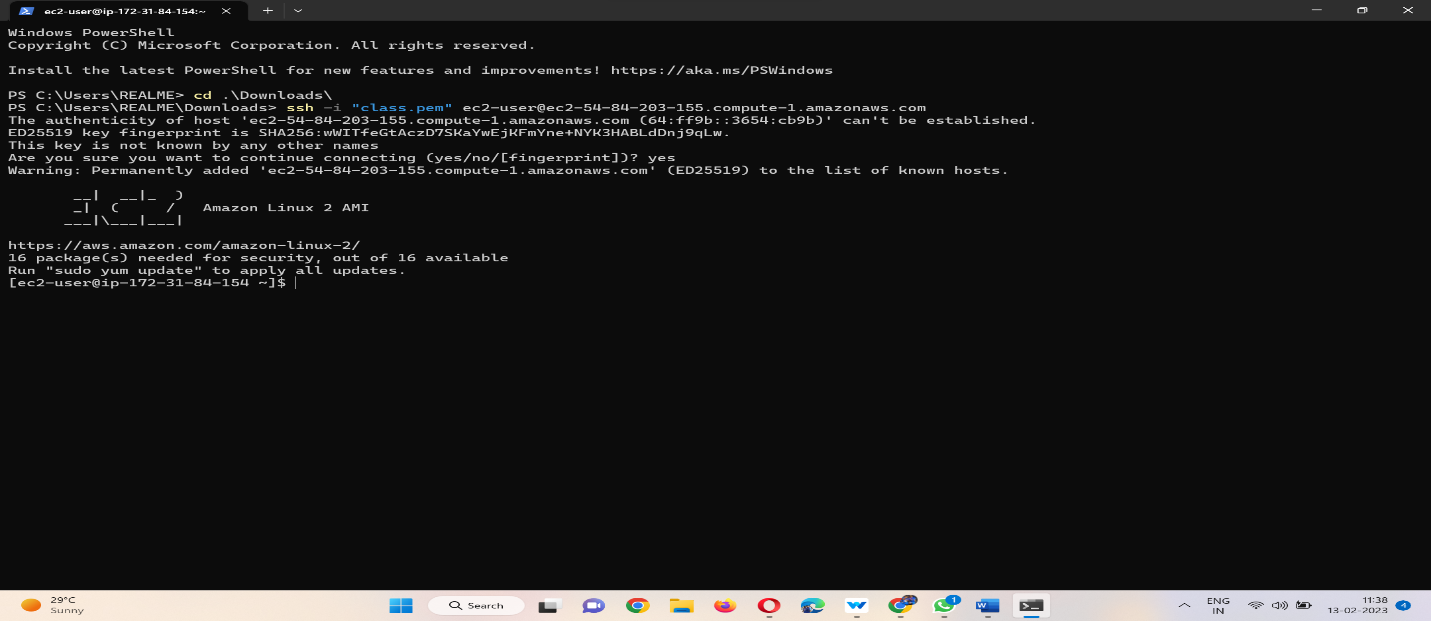


* Finally, Instance launched a WordPress-Server Instance.
* Now to connect an EC2 instance we can connect with the console.



1. Go to the instance page and click on connect then go to SSH Client/EC2 instance connect.

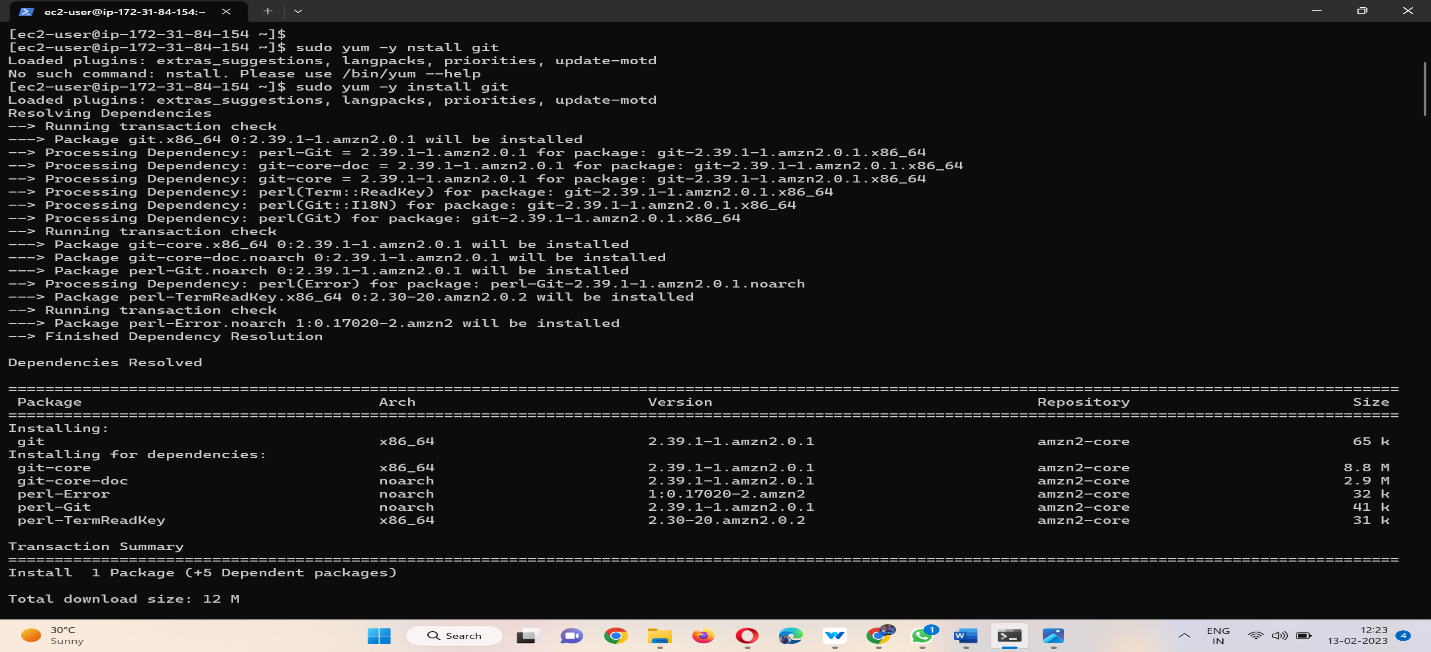
* By using the SSH command we are accessing our instance in Command Line Interface Terminal.



# Module – 2: Installing GIT, Docker, Java and Jenkins, and related repos

1. Installing GIT in our instance by using the below command.

* Sudo yum -y install git



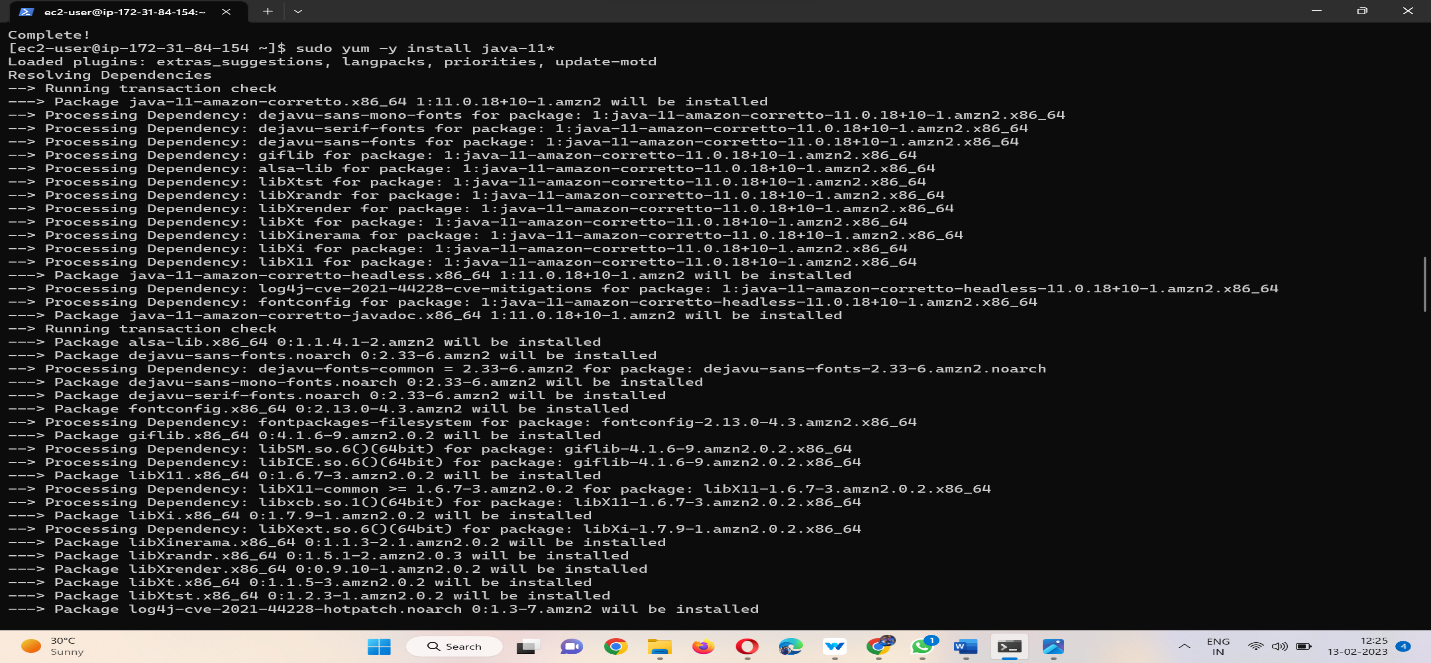
1. Installing Docker in our instance by using below command.

* Sudo yum -y install docker
* Sudo systemctl start docker



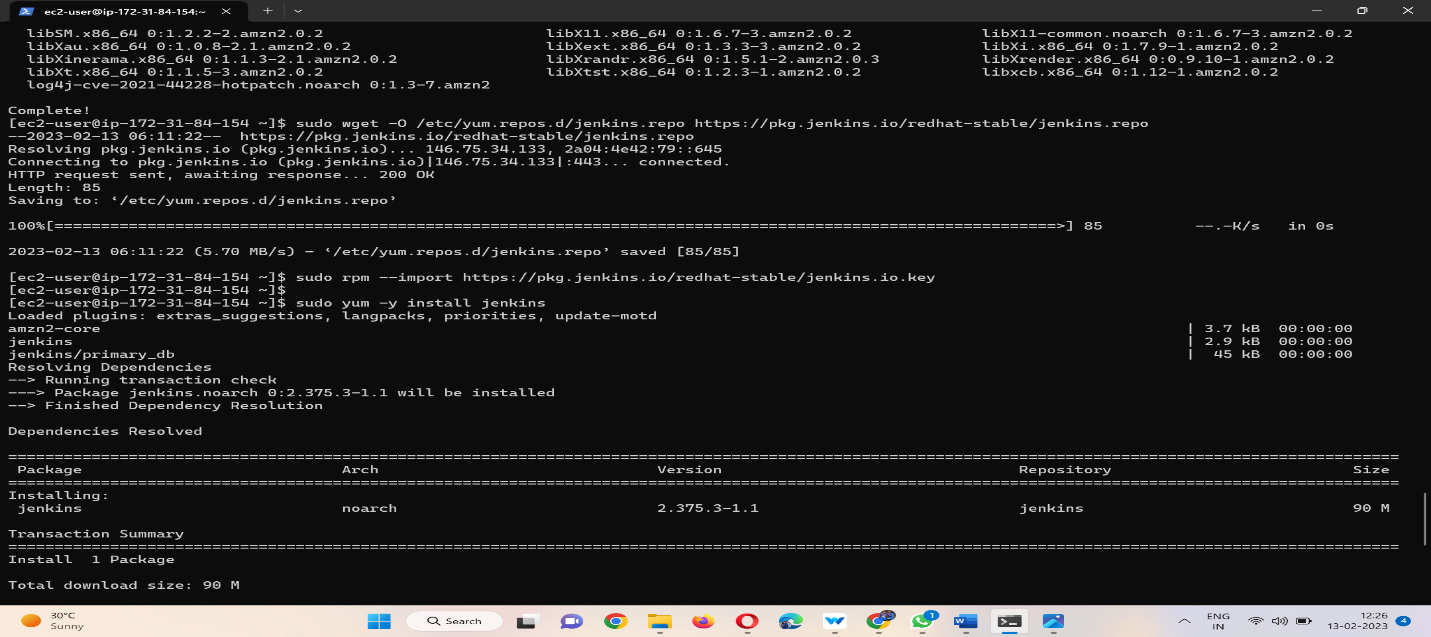
1. Installing Java in our instance by using the below command.

* sudo yum -y install java-11\*



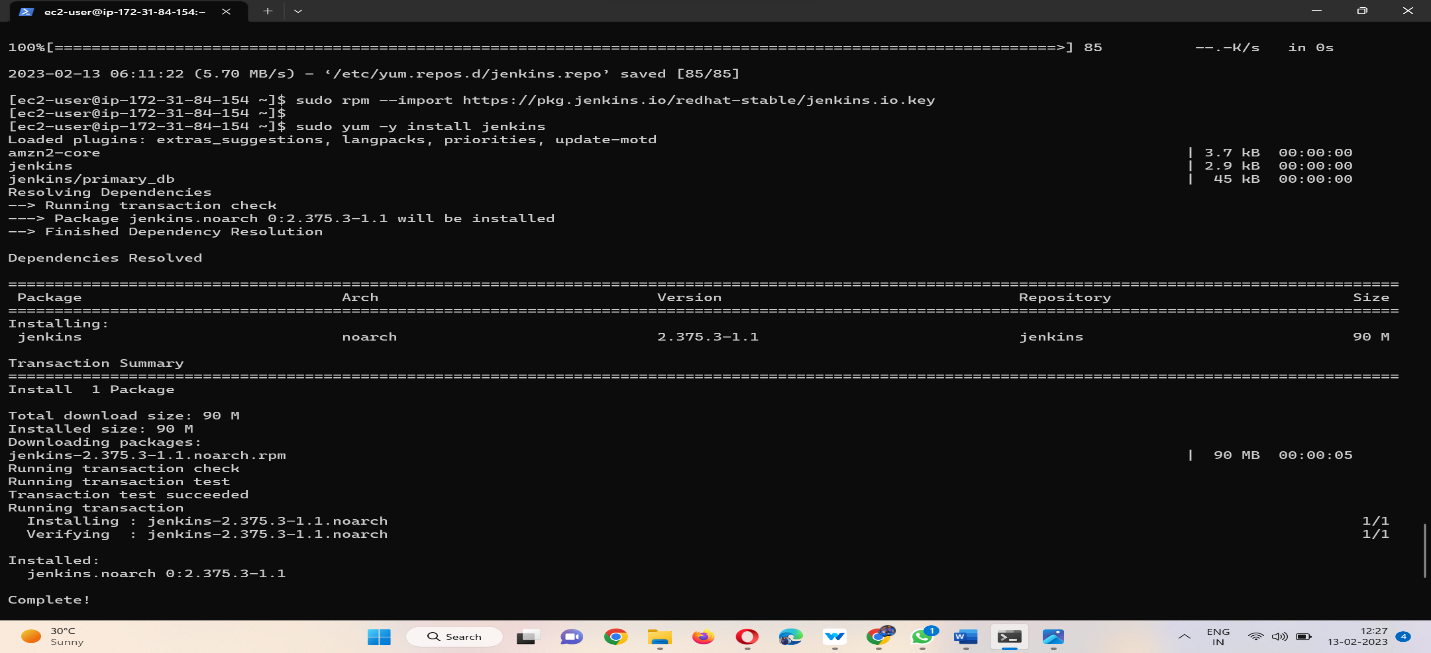
1. **Before we can install Jenkins through the package manager, we’ll need to add the Jenkins repository, and keys By using this command.**

* **sudo wget -O /etc/yum.repos.d/jenkins.repo** <https://pkg.jenkins.io/redhat-stable/jenkins.repo>
* **sudo rpm --import** <https://pkg.jenkins.io/redhat-stable/jenkins.io.key>



1. Install Jenkins by using this command.

* sudo yum -y install jenkins
* sudo systemctl start jenkins



# Module – 3: Setup the Jenkins with related repos. And create jobs.

# Unlock the Jenkins by using this command

# sudo cat /var/lib/jenkins/secrets/initialAdminPassword

# 

# Set up the configuration with user details and save the details.

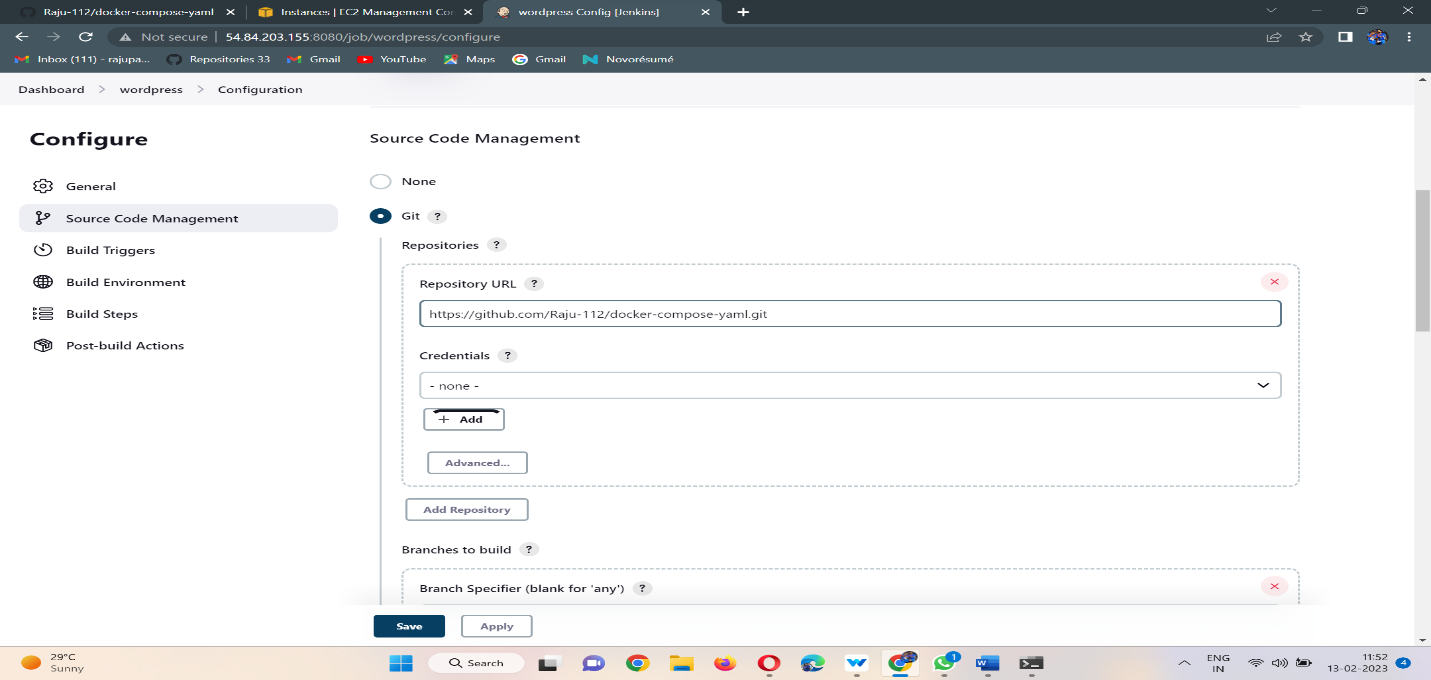
# 

# And then lastly, I had a look to see that this was running correctly.

# This is the Jenkins welcome page.

# 

# Create the first job by cloning the Git link, and save the job



# After saving the first job build the job.

# Here is the console output.

# Give the permissions to Jenkins and docker by using these commands

# sudo vi /etc/sudoers

# ## Allow all users without passwords.

# jenkins ALL=(ALL) NOPASSWD: ALL

# docker ALL=(ALL) NOPASSWD: ALL

# 

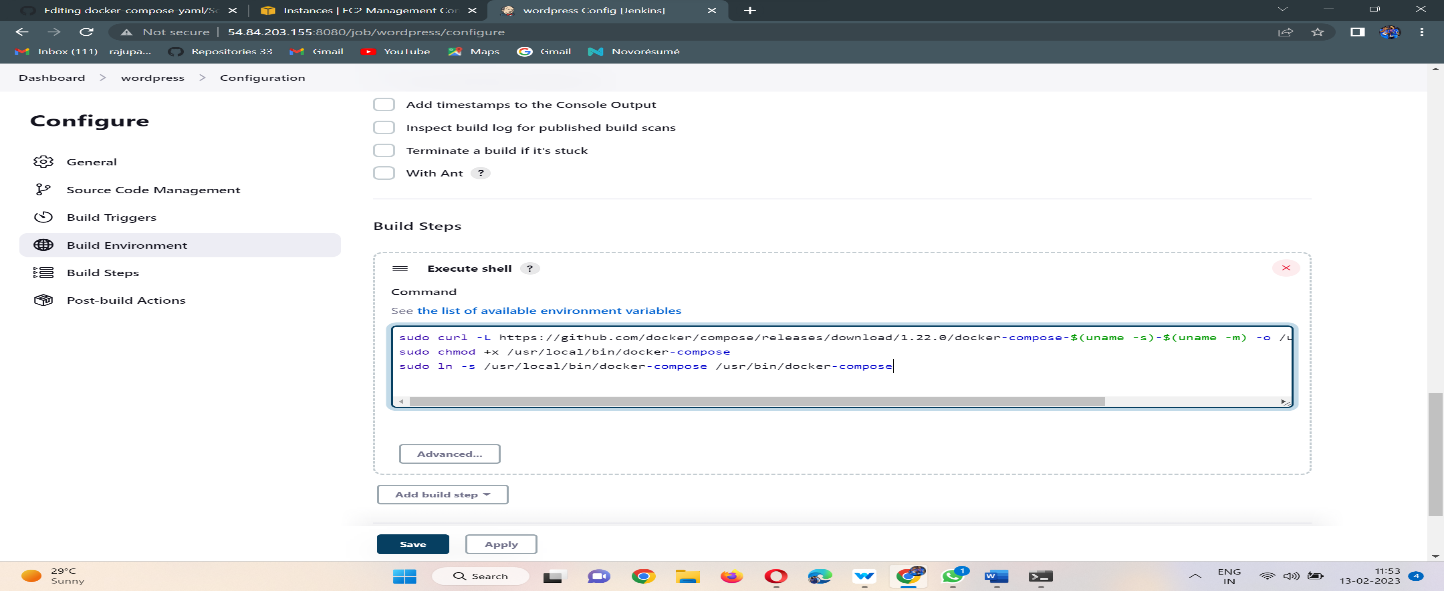
# Create the second WordPress build job by using execute a shell.

# Compose the docker by using these commands.

1. Apply executable permissions to the binary and create a symbolic link.

# sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

* sudo chmod +x /usr/local/bin/docker-compose
* sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose



# Save the job build the job

# Here is the console output.

# Finally pull the images by using this command.

# sudo docker-compose up -d

# 

# Here is the console output.

# 

# 

# Finally deploy the WordPress application.

# And then lastly, I had a look to see that this was running correctly.

# 

1. This is the WordPress welcome page

# 