Module – 11

Installing Jenkins:

- Update Ubuntu

sudo apt update

sudo apt upgrade

- Install Java:

sudo apt install openjdk-11-jdk

- Install Jenkins:

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins

- Start Jenkins:

sudo systemctl start jenkins

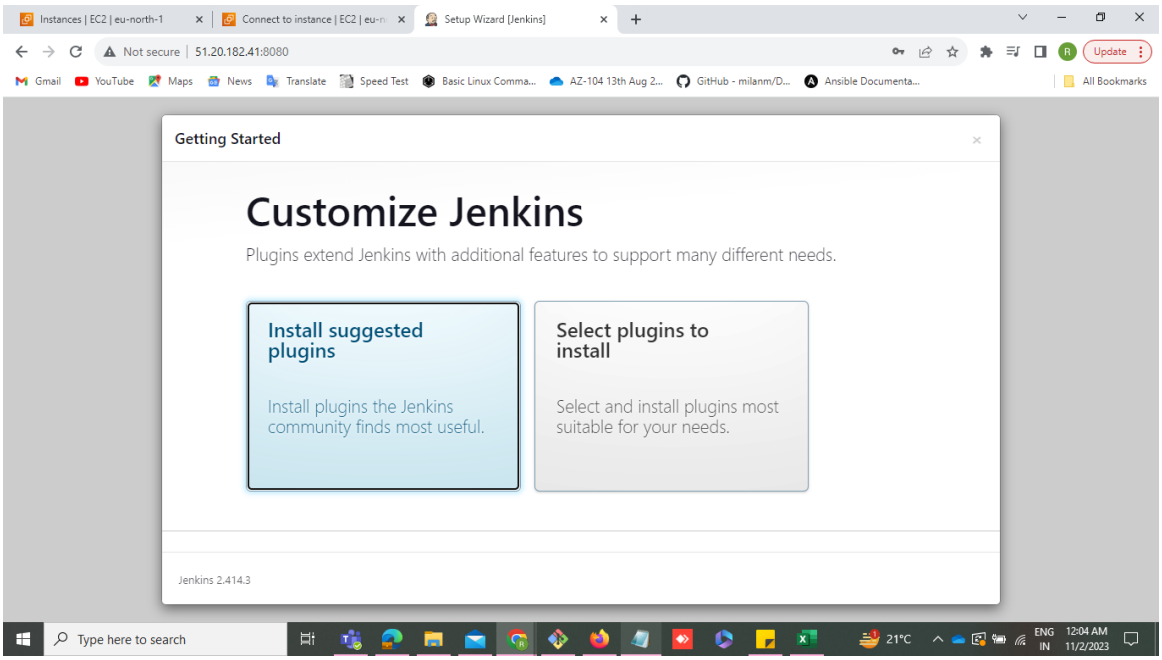
sudo systemctl enable jenkins

- Open the firewall:

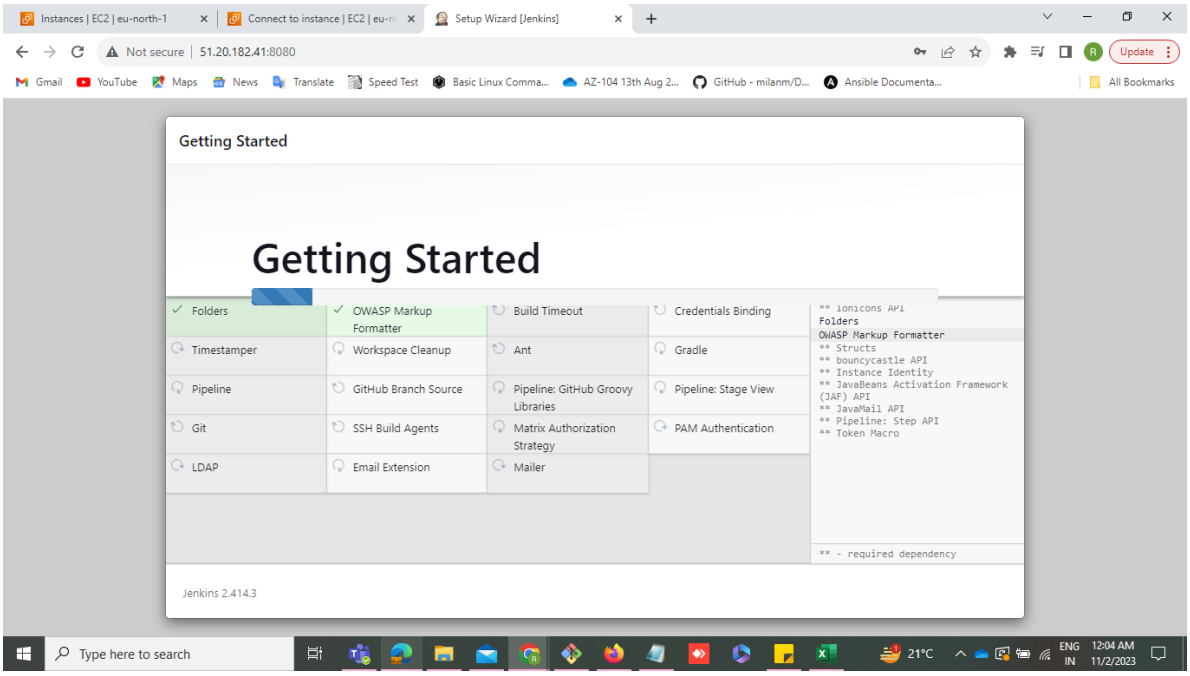
sudo ufw allow 8080

- Get the unlock jenkins key:

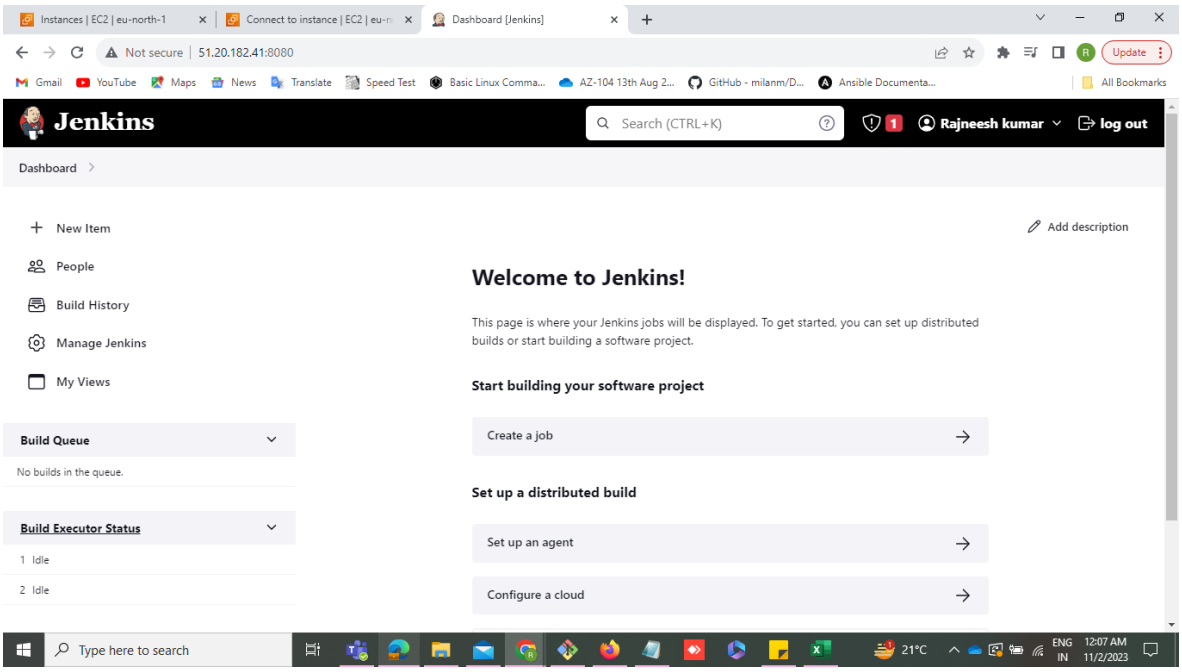
sudo cat /var/lib/jenkins/secrets/initialAdminPassword



- Installing plugins:



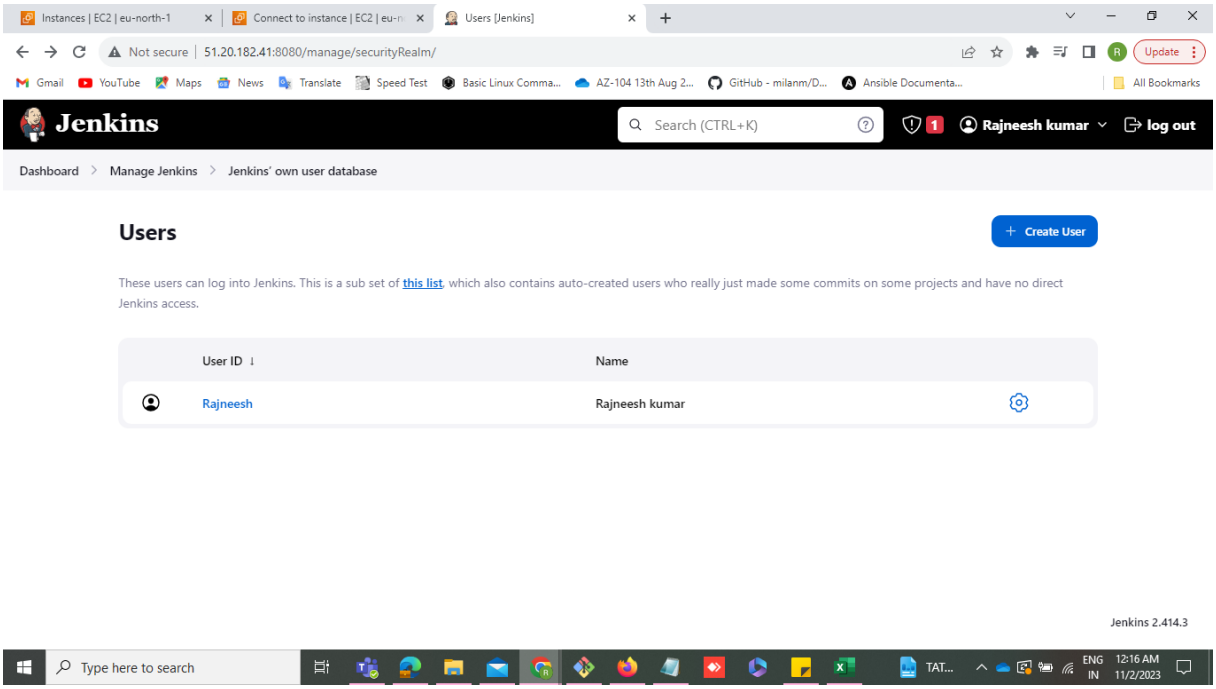
- Installed and working fine:



- Reset the password of the jenkins users:

After logging in as an administrator, follow these steps to reset the password for a user:
a. Click on "Manage Jenkins" on the Jenkins home page.
b. Click on "Manage Users."
c. Find the user for whom you want to reset the password and click on their username.
d. In the user's profile page, click on the "Configure" button.
e. In the "User Details" section, you'll see an option to change the user's password. Enter the new password.
f. Click "Save" to apply the changes.

From here we can manage users:

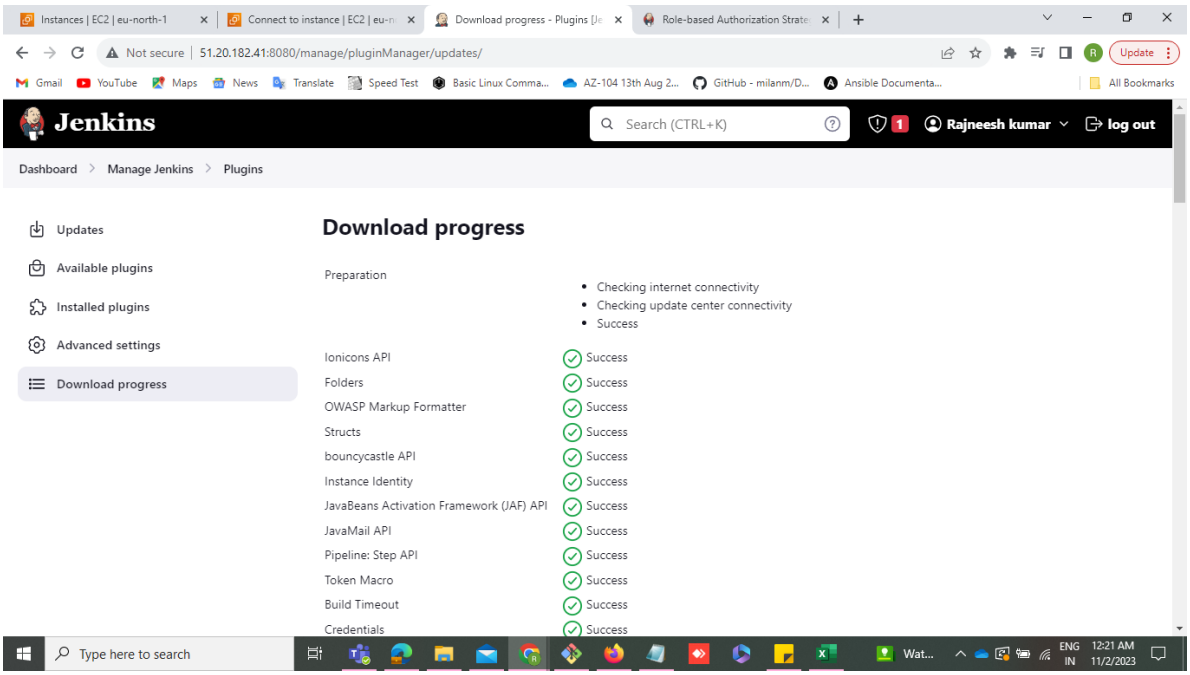


- Configure Jenkins roles and define permissions:

Jenkins provides a Role-Based Access Control (RBAC) plugin that allows you to configure roles and define permissions for different users and groups. Here are the general steps to configure Jenkins roles and define permissions using the RBAC plugin:

Step 1: Install the Role-Based Access Control Plugin

- Open a web browser and navigate to your Jenkins server.
- Log in to Jenkins with administrative credentials.
- Click on "Manage Jenkins" in the left-hand menu.
- Click on "Manage Plugins."
- In the "Available" tab, search for "Role-Based Access Control" or "RBAC."
- Select the "Role-Based Access Control" plugin and click the "Install without restart" button.



Step 2: Create Roles

- After installing the RBAC plugin, go to the Jenkins home page.
- Click on "Manage Jenkins" in the left-hand menu.
- Click on "Manage and Assign Roles."
- Click on the "Manage Roles" tab.
- Click on the "Add Role" button to create a new role.
- Give the role a name and a description. For example, you could create roles like "Admin," "Developer," "Tester," etc.
- Click "Save" to create the role.

Step 3: Define Permissions for Roles

- In the "Manage and Assign Roles" page, click on the "Assign Roles" tab.
- Select a role from the list of roles you've created.
- Under "Assignable Scopes," choose the appropriate permissions for the role. You can define permissions for different parts of Jenkins, such as "Overall," "Job," "Run," "View," and more.
- Configure permissions according to your requirements. For example, you can grant "Job/Build" permission for the "Developer" role to allow them to build jobs.
- Click "Save" to save the role's permissions.

Step 4: Assign Roles to Users and Groups

- In the "Manage and Assign Roles" page, go to the "Assign Roles" tab.
- Select a role from the list of roles.
- In the "Assignees" section, you can assign the role to specific users or groups. You can use Jenkins users or user groups from your authentication system (e.g., LDAP groups).
- Click "Add" to add users or groups to the role.
- Configure the roles and assignments according to your organization's requirements.

Step 5: Test Permissions

- Log in to Jenkins with a user account that has been assigned a specific role.
- Verify that the user can perform the actions defined by the role's permissions.

This is a general overview of configuring roles and permissions in Jenkins using the RBAC plugin. The specific permissions and roles you define will depend on your organization's needs and Jenkins usage. It's important to carefully plan and document your role and permission assignments to ensure the security and proper functioning of your Jenkins environment.

- Configure Jenkin master & slave

Configuring Jenkins with a master-slave (also known as controller-agent) architecture allows you to distribute the load and run jobs on multiple agents to improve scalability and efficiency. Here are the general steps to configure a Jenkins master and slave setup:

Step 1: Set Up Jenkins Master

- Install Jenkins on a dedicated server or virtual machine. You can refer to the previous responses for instructions on how to install Jenkins.
- Access the Jenkins web interface on the master by navigating to `http://your-master-ip:8080` in a web browser (replace "your-master-ip" with your master's IP address).
- Log in to Jenkins as an administrator.

Step 2: Install and Configure Java on the Master

Ensure Java is installed on the master server as Jenkins relies on it.

Step 3: Create a Jenkins Slave

- On the Jenkins master, go to the Jenkins dashboard.
- Click on "Manage Jenkins" in the left-hand menu.
- Select "Manage Nodes and Clouds."
- Click on the "New Node" or "New Agent" option.
- Provide a name for the slave node, choose "Permanent Agent," and click "OK."
- In the node configuration, you'll need to set up some options:
- **# of Executors:** Define the number of concurrent builds the slave can handle.
- **Remote root directory:** This is where Jenkins stores data on the slave. Ensure it exists on the slave server.
- **Labels:** Optionally assign labels to the node to identify its capabilities.
- **Usage:** Choose "Utilize this node as much as possible."
- **Launch method:** Depending on your infrastructure, you can use SSH, JNLP (Java Web Start), or other methods to connect to the slave. SSH is a common choice for security.
- Save your configuration.

Step 4: Set Up the Slave Server

On the server that will be the Jenkins slave:

- Install Java (as Jenkins agents require Java to run).
- Install Jenkins agent software. The specific steps depend on your operating system. For example, for Linux, you might download and run the agent JAR file.
- Start the agent by running a command like:
- bashCopy code

  `http://your-master-ip:8080/computer/your-node-name/slave-agent.jnlp`

- Replace "your-master-ip" with the IP address of the Jenkins master.
- Replace "your-node-name" with the name you assigned to the node in Jenkins.
- Replace "your-secret-key" with the secret key provided by Jenkins when configuring the node.

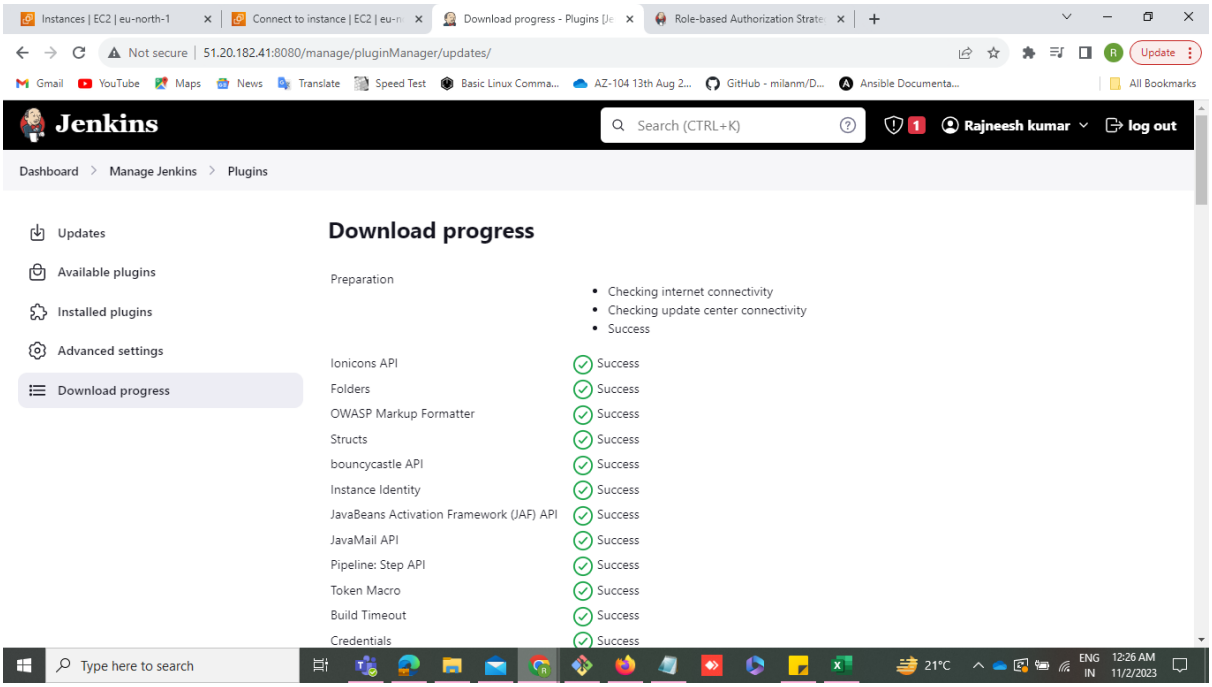
Step 5: Verify the Connection

On the Jenkins master, you should see the new node listed as online in the "Manage Nodes and Clouds" section. You can now create jobs in Jenkins and specify that they should run on the slave node by assigning the appropriate label.

By following these steps, you have successfully configured a Jenkins master and slave setup. Jenkins will distribute the workload to the slave(s) based on the job configurations, allowing you to scale your Jenkins infrastructure to handle larger workloads.

- Create a Jenkin pipeline with Github and deploy a web server & Wordpress website in ubuntu EC2 instance

Installing Github intigration plugin:



Creating a Jenkins pipeline that automates the deployment of a web server and a WordPress website to an Ubuntu EC2 instance, integrated with GitHub, involves several steps. Below, I outline the high-level process to achieve this:

Step 1: Set Up Your EC2 Instance

- Launch an EC2 instance using Ubuntu AMI, and ensure it has a security group with the necessary rules to allow SSH, HTTP, and HTTPS traffic.
- Allocate an Elastic IP address to the instance if you want to associate a static IP.
- Connect to your EC2 instance via SSH and perform the initial server setup, including installing required software like a web server (e.g., Nginx or Apache) and PHP.

Step 2: Install Jenkins on the EC2 Instance

Follow the instructions provided earlier to install Jenkins on your EC2 instance running Ubuntu.

Step 3: Configure Jenkins

- Open Jenkins in your web browser by navigating to `http://your-ec2-instance-ip:8080`.
- Install necessary plugins, including the "GitHub Integration" plugin.
- Set up credentials for connecting to your GitHub repository.
- Create a Jenkins job or pipeline to build and deploy your WordPress website. This pipeline should include the following stages:

Step 4: Create a Jenkins Pipeline

Here's a simplified Jenkins pipeline script to deploy a WordPress website on your Ubuntu EC2 instance:

groovyCopy code

```
pipeline {
  agent any
  stages {
    stage('Checkout from GitHub') {
      steps {
        git branch: 'main', credentialsId: 'your-github-credentials', url: 'https://github.com/yourusername/your-repo.git'
      }
    }

    stage('Deploy to EC2') {
      steps {
        sshagent(credentials: ['your-ssh-key-id']) {
          sh '''
            ssh -o StrictHostKeyChecking=no -i /var/lib/jenkins/.ssh/id_rsa ubuntu@your-ec2-instance-ip '
            # Your deployment commands here (e.g., install and configure web server, deploy WordPress)
            '
          '''
        }
      }
    }
  }
}
```

- **`your-github-credentials`**: The Jenkins credentials ID for your GitHub account.

- `https://github.com/yourusername/your-repo.git`: The URL to your GitHub repository.
- `your-ssh-key-id`: The Jenkins credentials ID for the SSH key associated with your EC2 instance.
- `your-ec2-instance-ip`: The public IP or DNS name of your EC2 instance.

Step 5: Configure GitHub Webhook

To trigger the Jenkins pipeline automatically on code changes, set up a GitHub webhook that points to your Jenkins server. Use the "GitHub Integration" plugin in Jenkins to enable webhook-based triggering of your pipeline.

Step 6: Test and Run the Pipeline

Make code changes in your GitHub repository and check if the Jenkins pipeline is triggered automatically. Jenkins will deploy your WordPress website to your EC2 instance whenever you push changes to your GitHub repository.

This is a simplified pipeline. Depending on your specific deployment requirements and the web server software you're using, you may need to customize the pipeline stages with additional deployment steps (e.g., setting up a database, configuring the web server, or installing WordPress).