Chapter 2

Installation Pre-Requisites

Before the book goes through the case studies, it would be helpful to set up the required environment needed to develop and deploy the projects. The following sections will ensure that you have as much of the setup completed to have a smoother time in later chapters.

Picking a Base OS

The book highly recommends that you use a native installation of Linux of your preferred flavour. Virtualizations like using a VM or WSL can lead to performance as well as compatibility issues that are hard to resolve especially when working on a large-scale project that has large resource requirement.

To demonstrate the installation procedure, we will use a fresh installation of Linux Mint. To get started we will want to update our package sources. We can achieve this using the following command:

```
sudo apt-get update
```

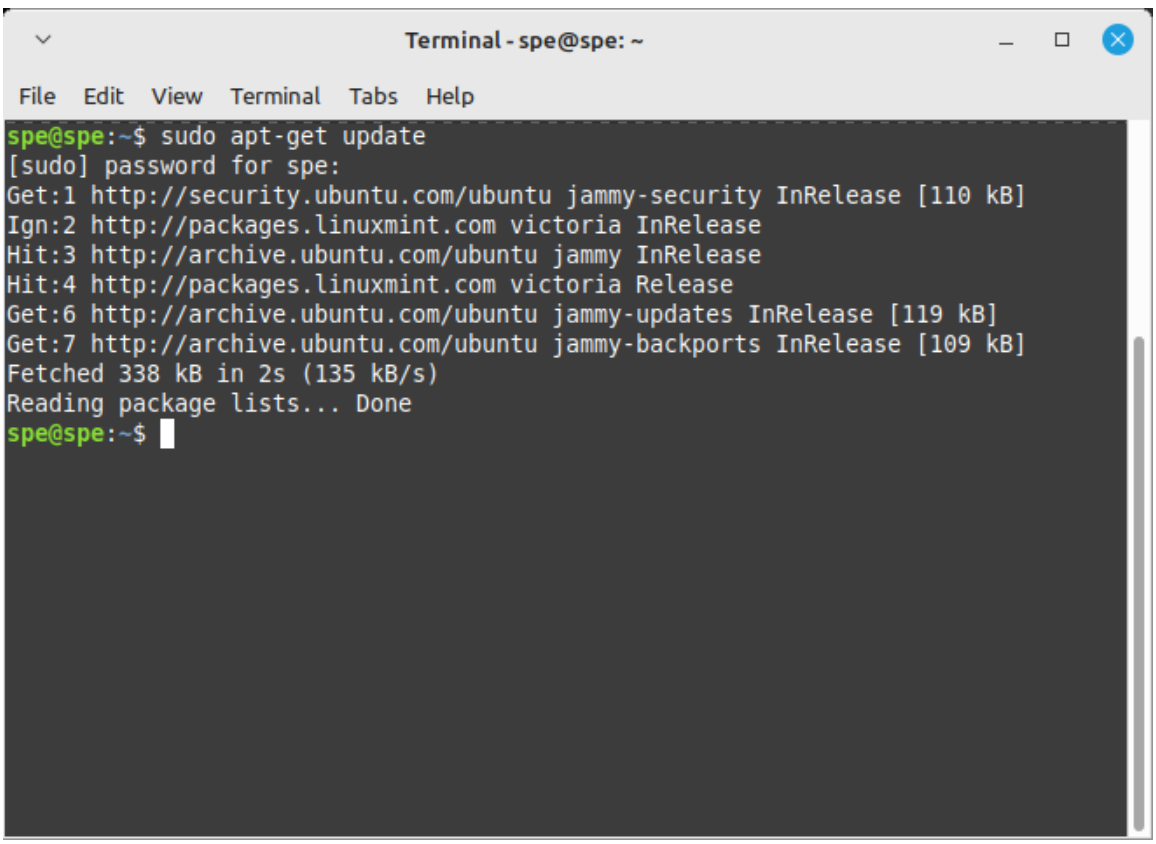


Fig 2.1 Updating library and package version details

The output of running the command should be that we have fetched the latest versions of the packages and dependencies. To upgrade said packages and dependencies we run the following command:

```
sudo apt-get upgrade
```

This will then upgrade all the packages and dependencies to the latest version. This should minimise the number of errors caused by version incompatibility. This process however can take a while and will ask for user input for every package that gets updated.

Installing JDK

To install the Java Runtime Environment, use the following command:

```
sudo apt install default-jre
```

If you do not have java installed on your system, this will download and install the default JRE that is compatible. In case JRE already is installed (as part of sudo apt-get upgrade) then you will see that no packages we installed by running the above command. To verify that we have successfully installed JDK, run the following command:

```
java --version
```

This should show you the current version of JRE that has been installed. An example output is as follows:

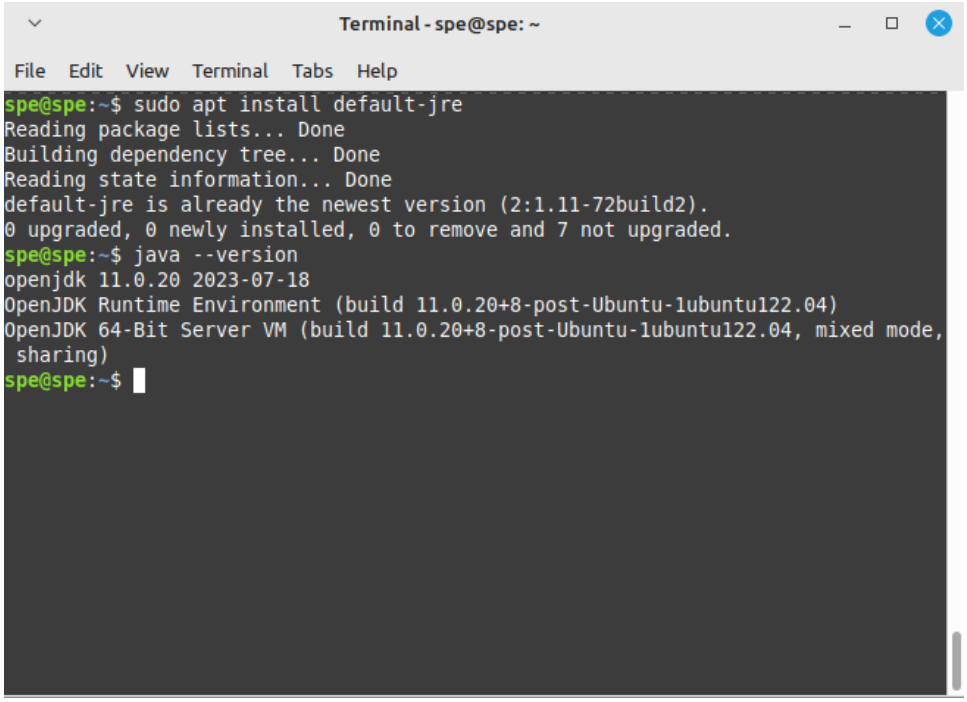


Fig 2.2 Installing and verifying JDK version

Now to install Java Development Kit, we will use the following command:

```
sudo apt install default-jdk
```

```
spe@spe:~$ javac
Command 'javac' not found, but can be installed with:
sudo apt install openjdk-11-jdk-headless  # version 11.0.20.1+1-0ubuntu1~22.04,
or
sudo apt install default-jdk              # version 2:1.11-72build2
sudo apt install ecj                      # version 3.16.0-1
sudo apt install openjdk-17-jdk-headless  # version 17.0.8.1+1~us1-0ubuntu1~22.0
4
sudo apt install openjdk-18-jdk-headless  # version 18.0.2+9-2~22.04
sudo apt install openjdk-19-jdk-headless  # version 19.0.2+7-0ubuntu3~22.04
sudo apt install openjdk-8-jdk-headless   # version 8u382-ga-1~22.04.1
spe@spe:~$ sudo apt install default-jdk
```

We can then verify the installation of the JDK using the following command:

```
javac --version
```

```
Terminal - spe@spe: ~
File  Edit  View  Terminal  Tabs  Help
spe@spe:~$ javac --version
javac 11.0.20.1
spe@spe:~$
```

Installing an IDE

The next step is to set up a development environment and for that we are going to install an IDE. You can choose whatever IDE that you feel comfortable in. The book is going to use IntelliJ as an example.

To install IntelliJ, first visit their website and download the Toolbox App in the form of a tarball (.tar.gz) and then extract and run it. This will open a GUI installer from where you can install the IDE. The detailed installation instructions can be found on their download website.

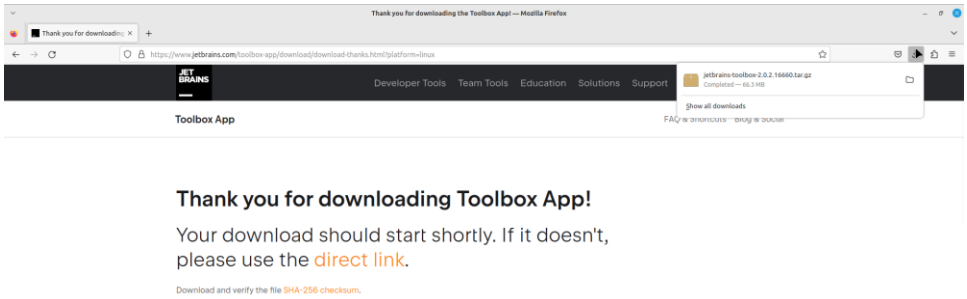


Fig 2.3 Downloading the IntelliJ Toolbox App

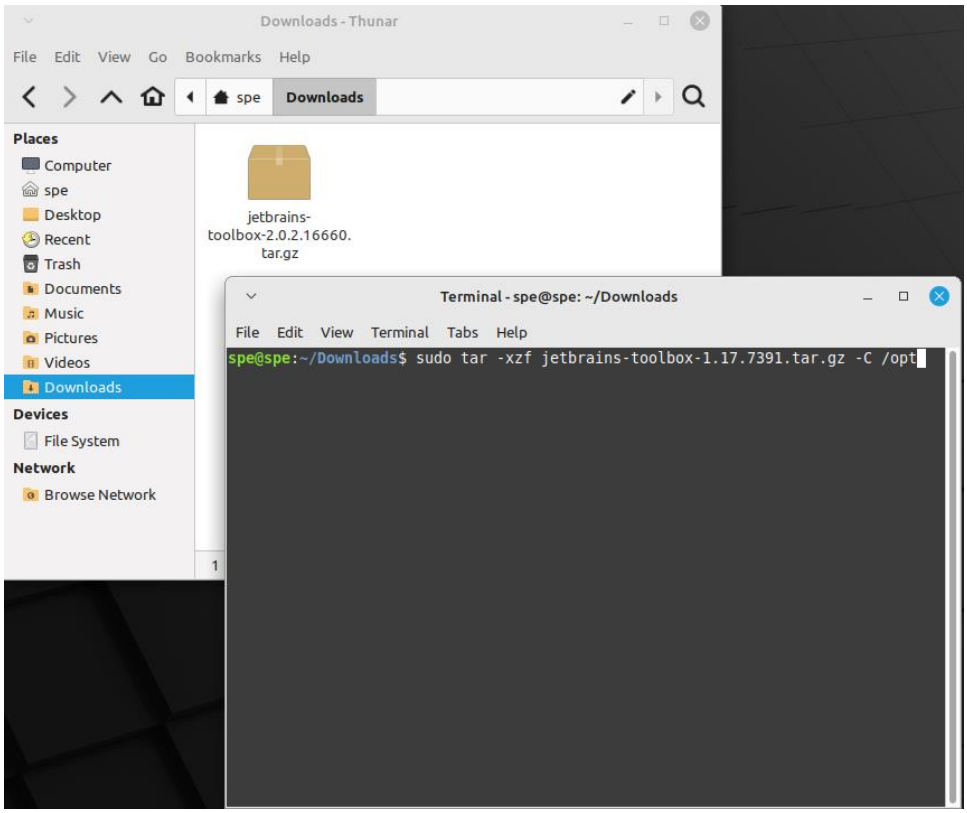


Fig 2.4 Running the IntelliJ Toolbox App

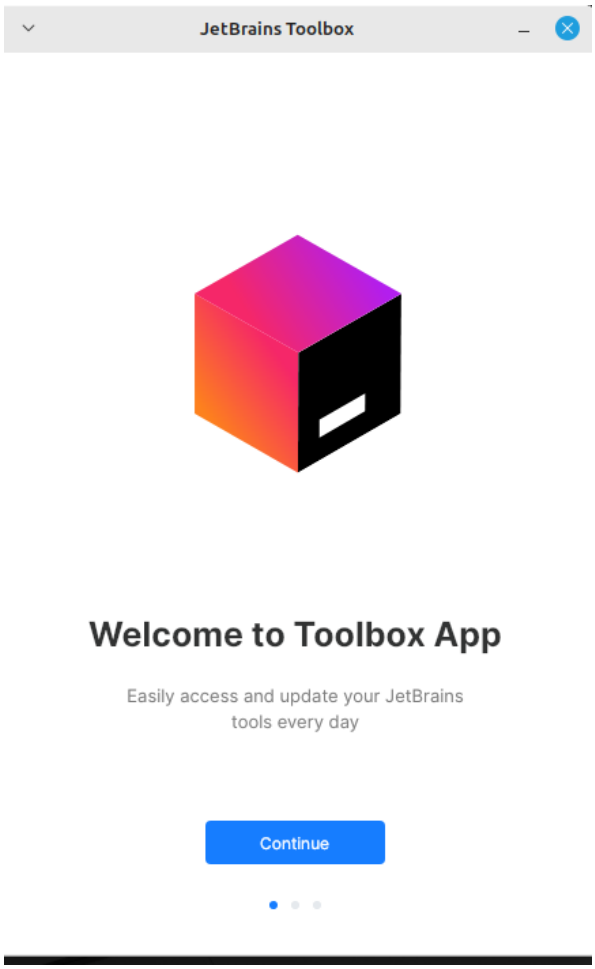


Fig 2.5 Starting the IntelliJ Toolbox App

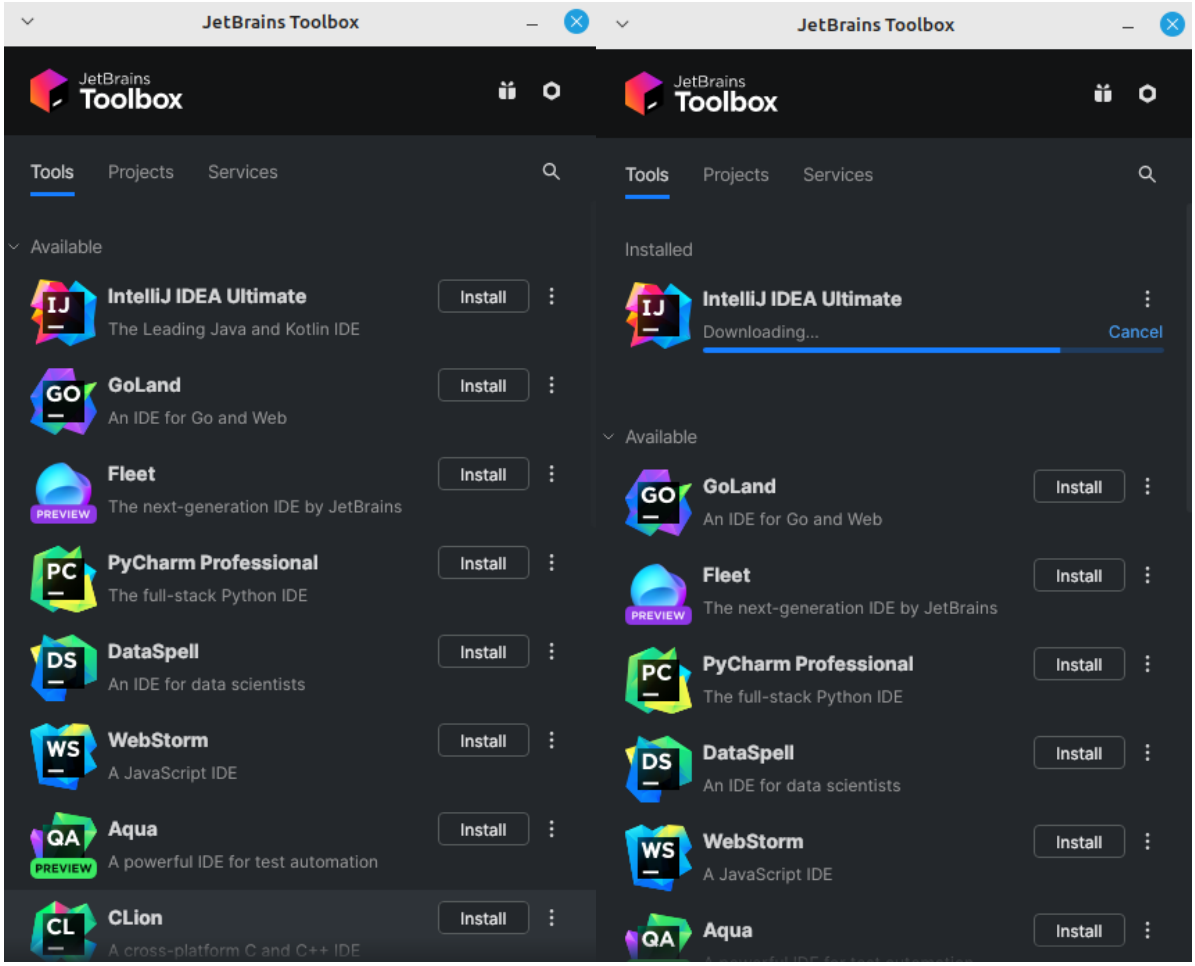


Fig 2.6 Installing IntelliJ IDEA

Fig 2.7 Installation successful

Installing Git

Git is the version control system that we will be using in this book. Git will be used by the IDE to sync our project to a remote repository which will be in GitHub.

To install git, we will use the following command:

```
sudo apt install git
```

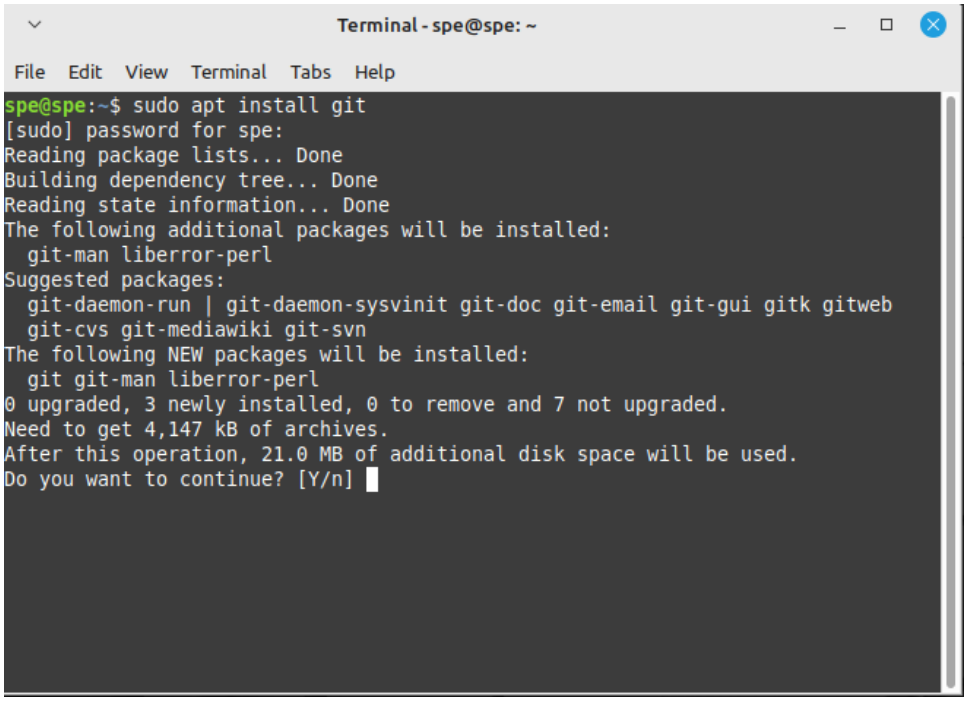


Fig 2.8 Installing git

Once you select Yes and the installation completes, you can verify it using the following command:

```
git –version
```

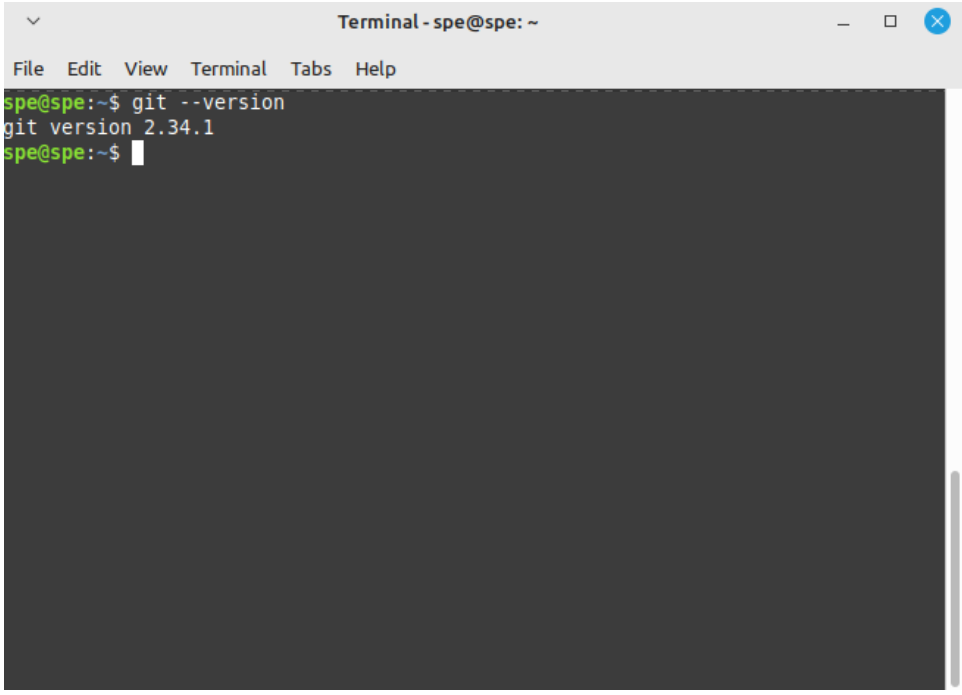


Fig 2.9 Verifying git installation version

Installing and Initialising Jenkins

Jenkins will be managing the CI/CD pipeline for us. To install Jenkins, we are going to be executing the following commands:

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

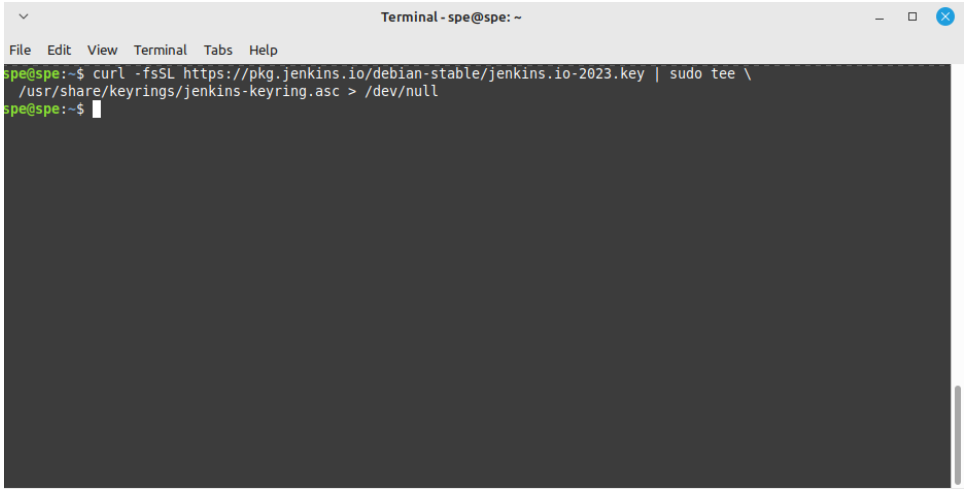


Fig 2.10 Fetching the certificates for Jenkins

```
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
```

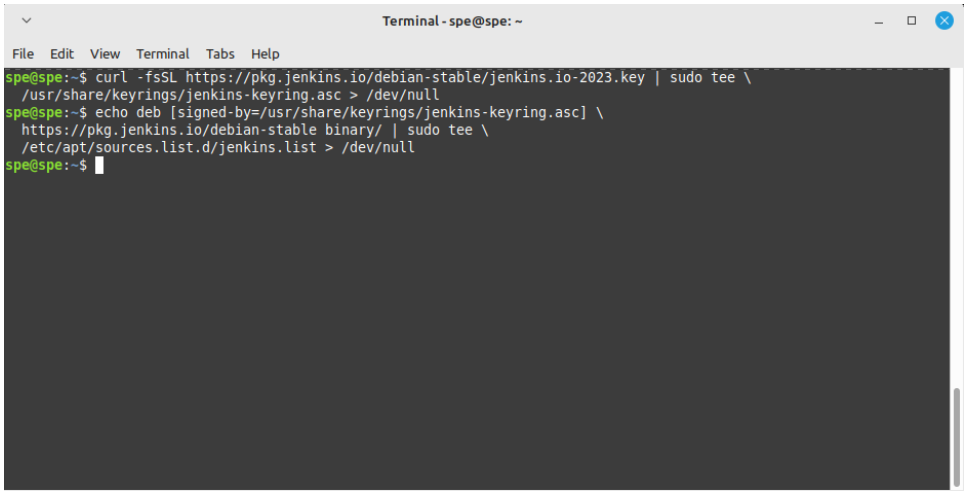


Fig 2.11 Fetching the installation dependencies for Jenkins

```
sudo apt install ca-certificates

sudo apt update

sudo apt install jenkins
```

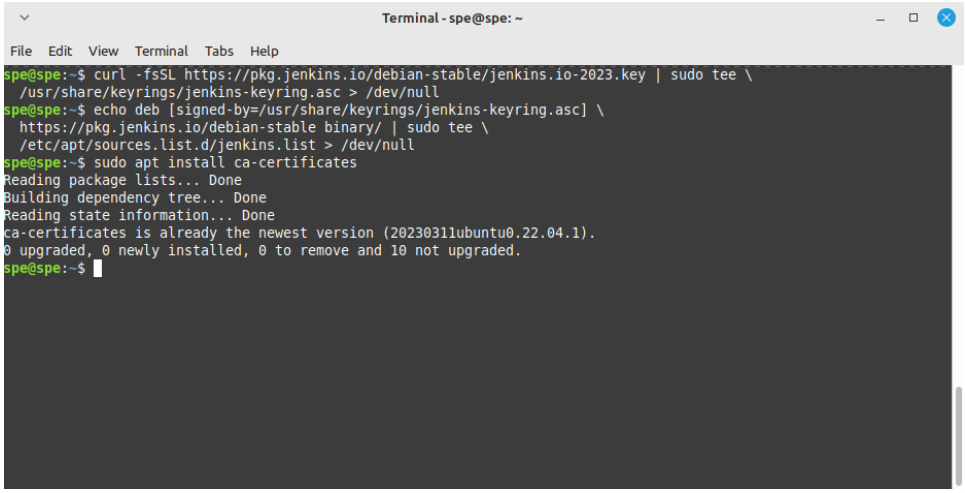


Fig 2.12 Refreshing the certificates

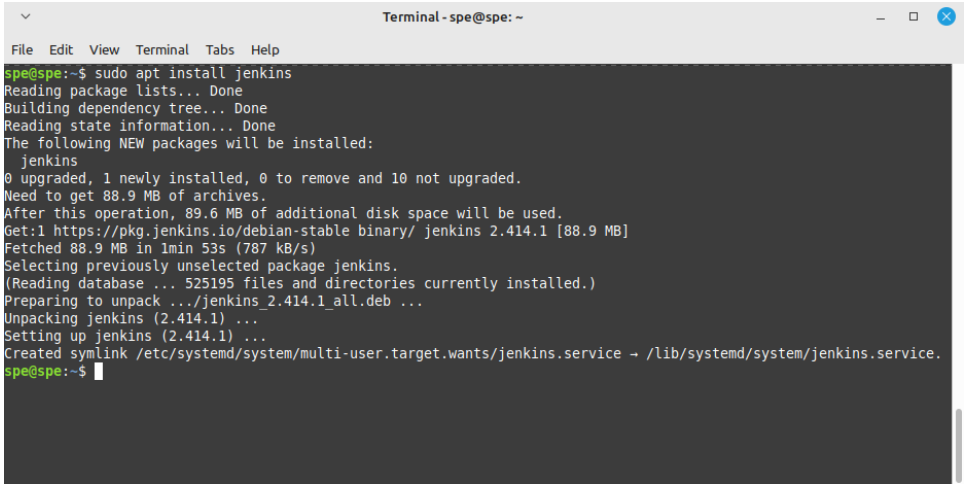


Fig 2.13 Installing Jenkins

Once we have installed it, we can start it and check its status using the following commands:

```
sudo systemctl start jenkins

sudo systemctl status jenkins
```

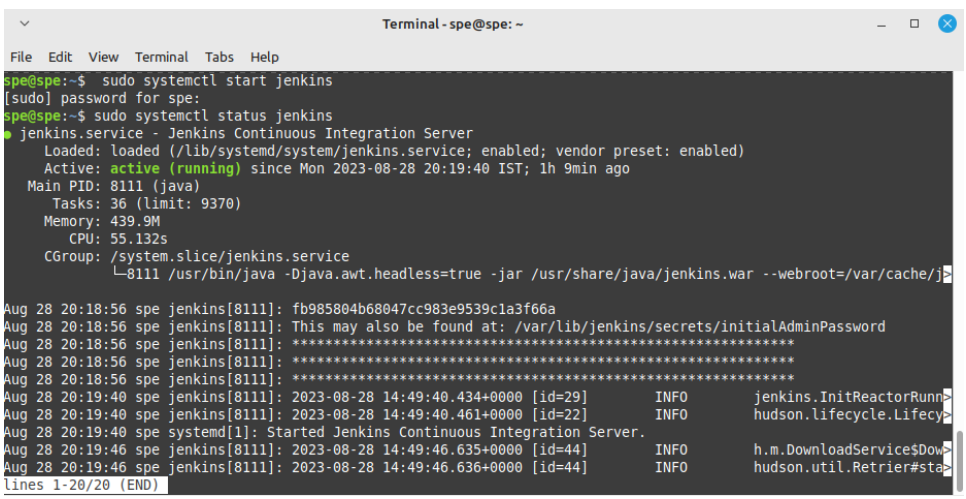


Fig 2.14 Starting and checking the status of Jenkins service

Once we have Jenkins up and running, we will need to configure it. We start by going to port 8080 on localhost. The URL is as follows:

http://localhost:8080/

When you visit the URL for the first time, Jenkins will ask for the password that has been written to the log file:

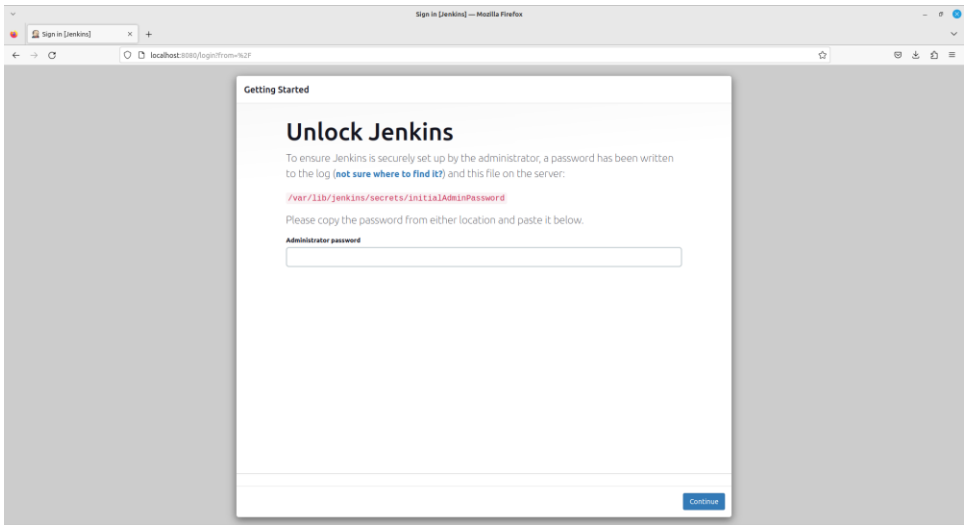


Fig 2.15 Unlocking Jenkins

To get this password, we will run the following command:

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

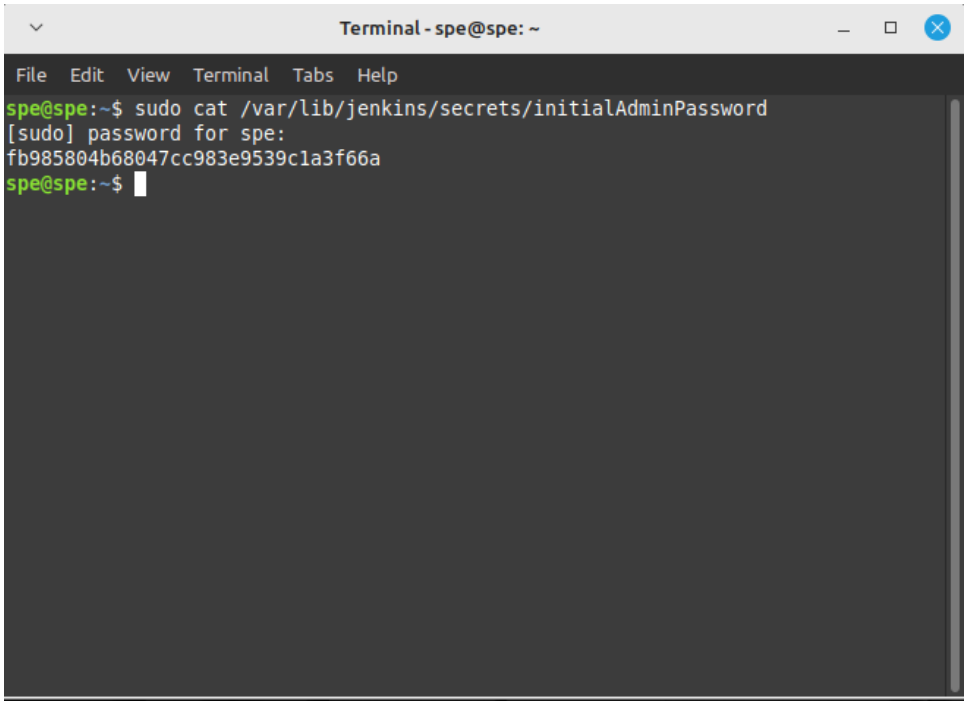


Fig 2.16 Getting Jenkins Admin Password

Copy and paste the password that shows on the terminal. Once we do this, Jenkins will ask us what plugins we want to install. We are going to choose the suggested plugins:

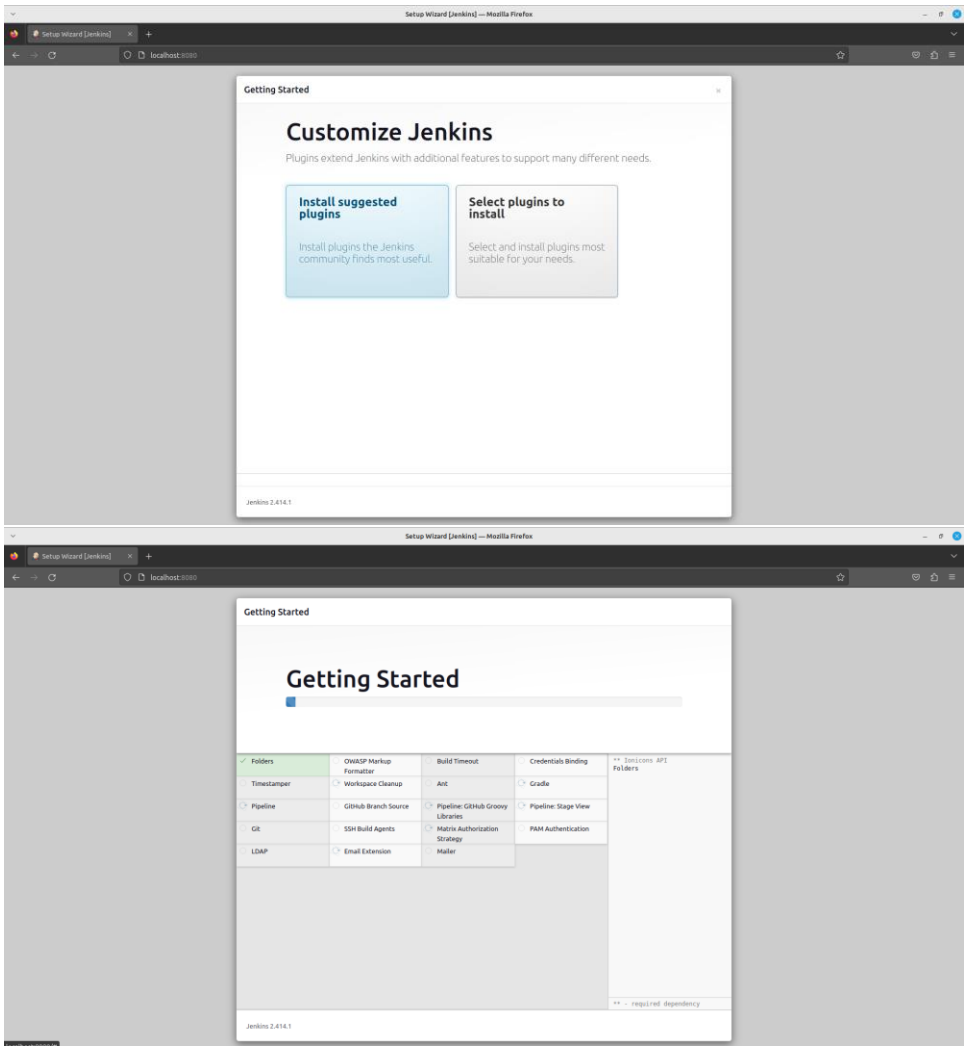


Fig 2.17 Installing suggested plugins

One of the known issues that many face is the failure to install plugins:

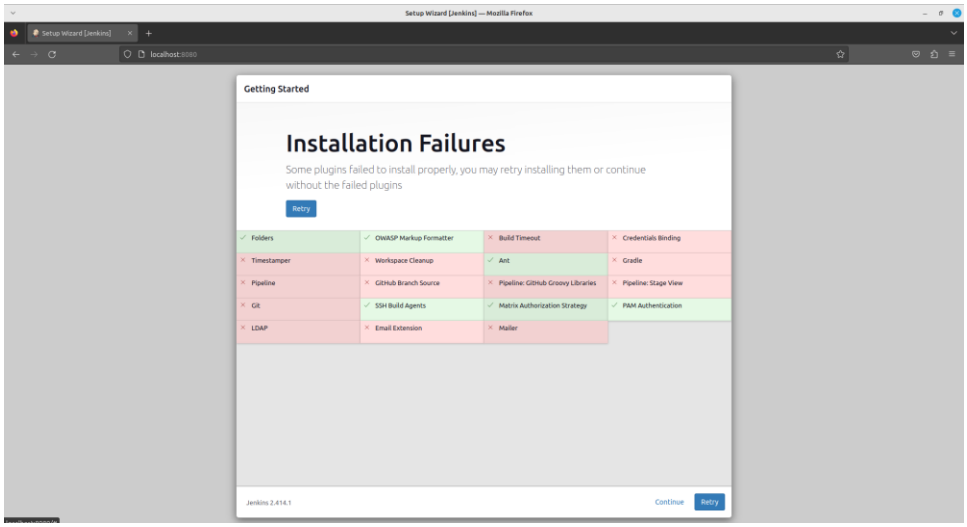


Fig 2.18 Failure to install few plugins

In such cases, keep retrying a few times as many of them resolve after a few attempts.

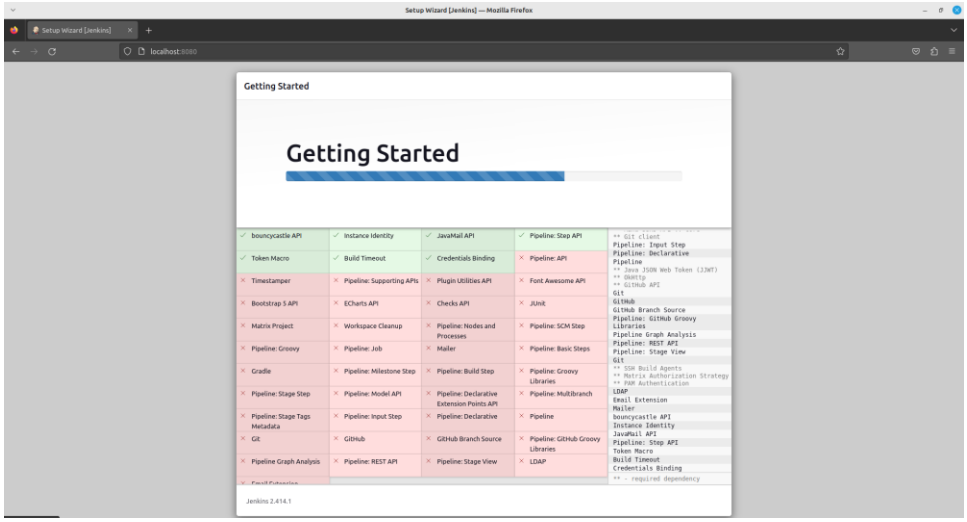


Fig 2.19 Retrying plugin installation

Once the errors resolve, the next step is the creation of Jenkins First User creation. You have the option of filling it. We will skip and continue as Jenkins Admin user:

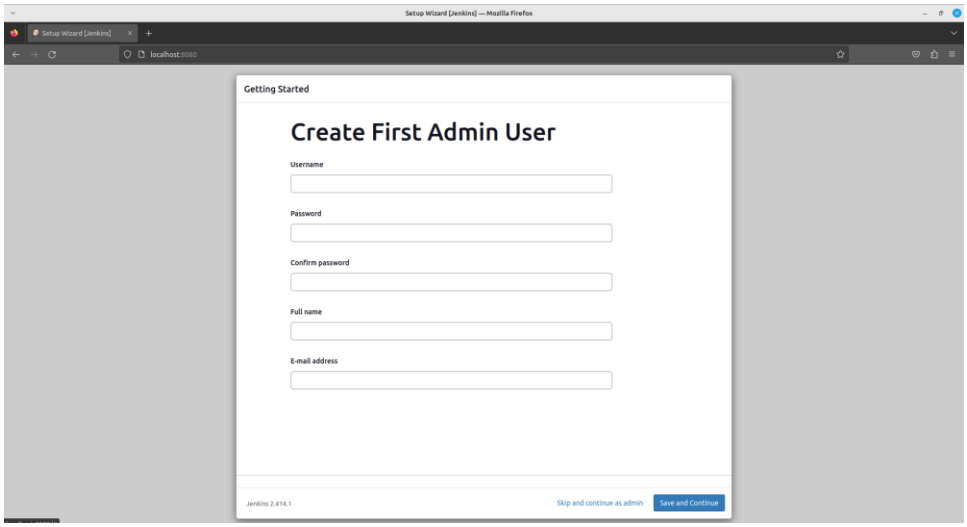


Fig 2.20 Optional First Admin User creation

The next step is the instance configuration. In this book we are going to keep the default configuration, which is to have Jenkins on localhost,

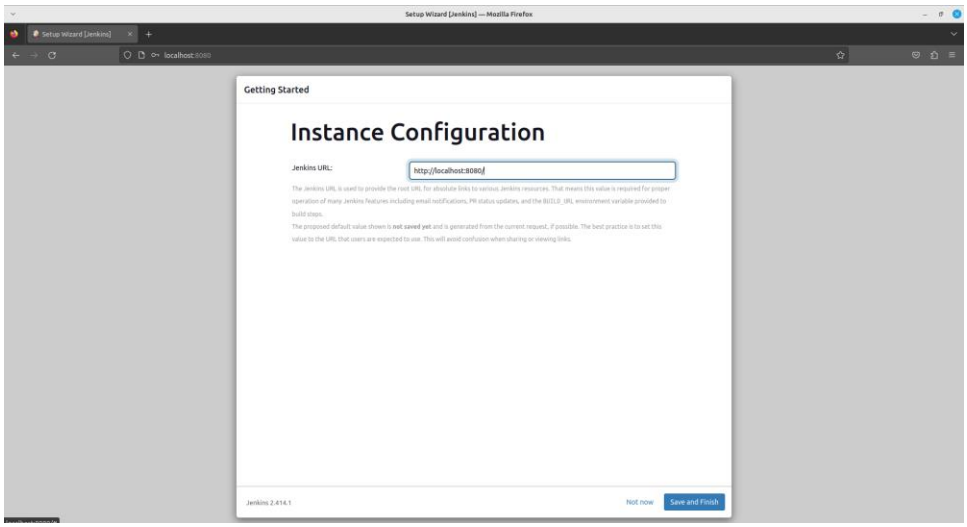


Fig 2.21 Jenkins URL Configuration

We then click on save and finish. We are now good to go

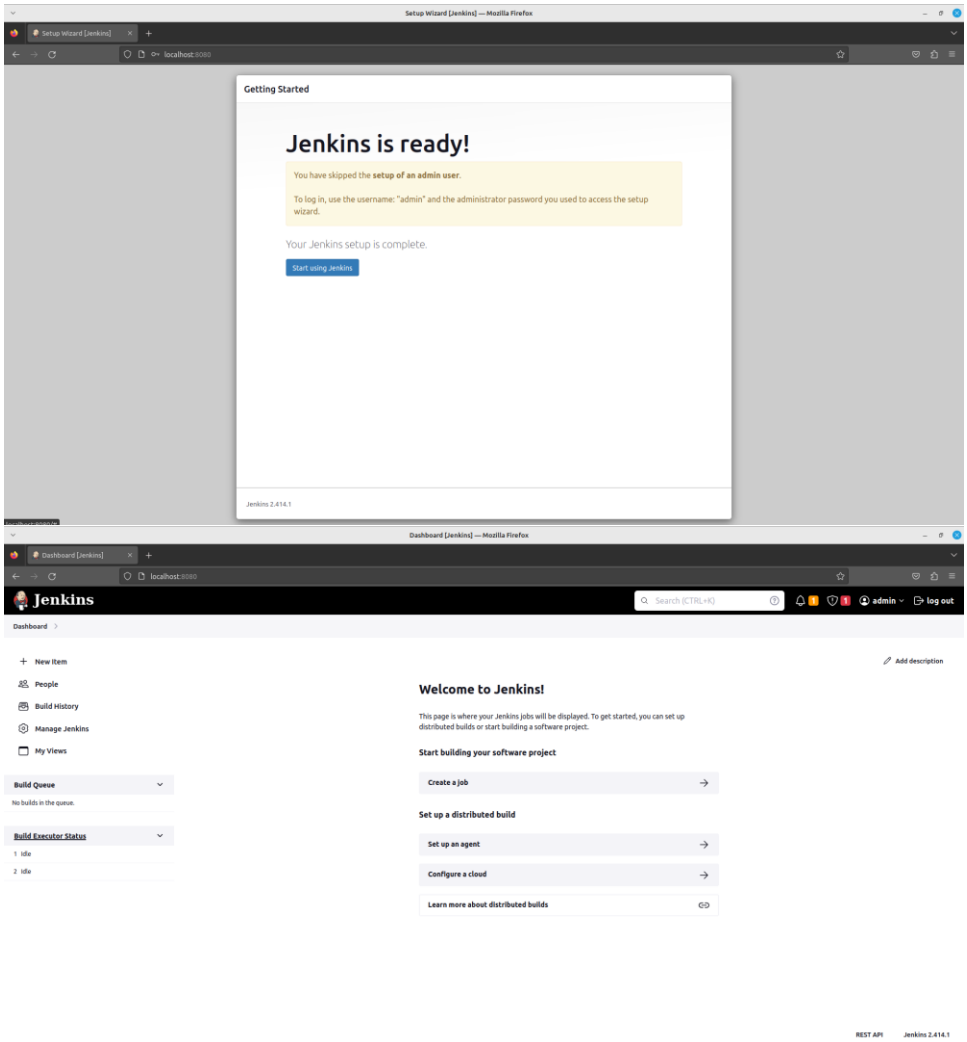


Fig 2.22 Jenkins Installation complete and dashboard

Installing and Initialising Docker

To install docker we are first going to make sure that curl is installed on your system. We do this using the following command:

```
sudo apt install curl
```

Once we do that, we are going to use curl to get the script needed to install docker using the following command:

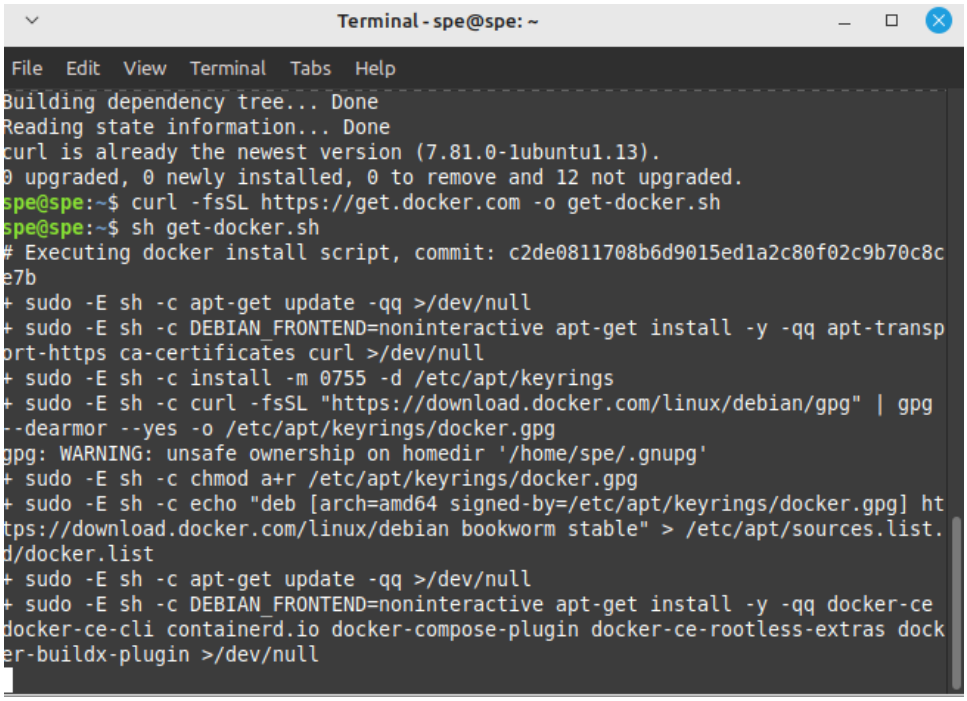
```
curl -fsSL https://get.docker.com -o get-docker.sh
```

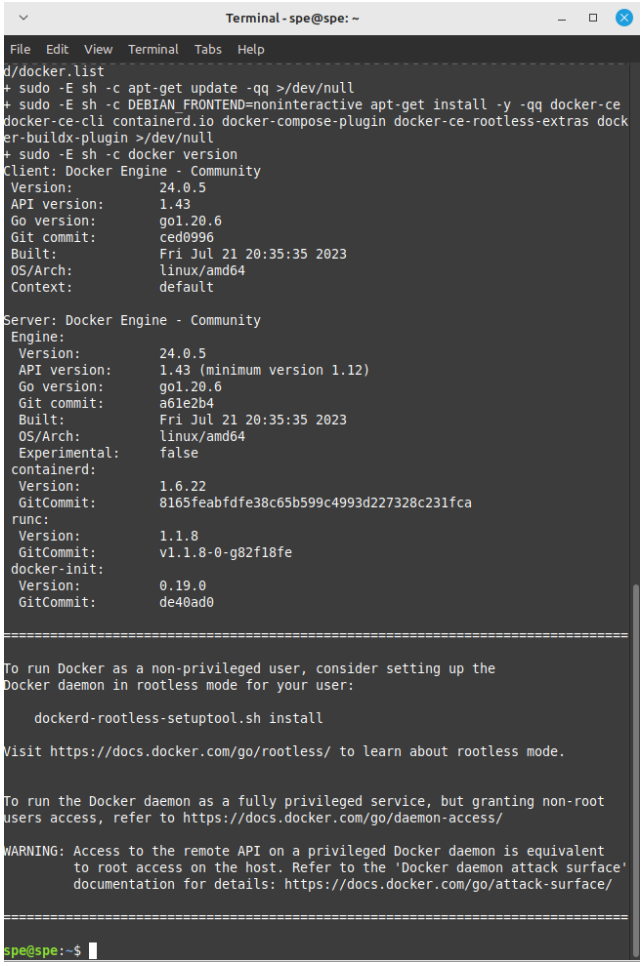
Once we have the script, we are going to run the script using the following command:

```
sh get-docker.sh
```

After the script finishes execution and installs Docker, we are going to check the version of Docker using the following command:

```
sudo docker --version
```





```
Terminal - spe@spe: ~
File  Edit  View  Terminal  Tabs  Help
d/docker.list
+ sudo -E sh -c apt-get update -qq >/dev/null
+ sudo -E sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce
docker-ce-cli containerd.io docker-compose-plugin docker-ce-rootless-extras dock
er-buildx-plugin >/dev/null
+ sudo -E sh -c docker version
Client: Docker Engine - Community
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.6
 Git commit:        ced0996
 Built:             Fri Jul 21 20:35:35 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          24.0.5
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.6
  Git commit:       a61e2b4
  Built:            Fri Jul 21 20:35:35 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.22
  GitCommit:        8165feabfdfe38c65b599c4993d227328c231fca
 runc:
  Version:          1.1.8
  GitCommit:        v1.1.8-0-g82f18fe
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0

================================================================================

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.


To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
         to root access on the host. Refer to the 'Docker daemon attack surface'
         documentation for details: https://docs.docker.com/go/attack-surface/

================================================================================

spe@spe:~$
```

Fig 2.23 Installing Docker

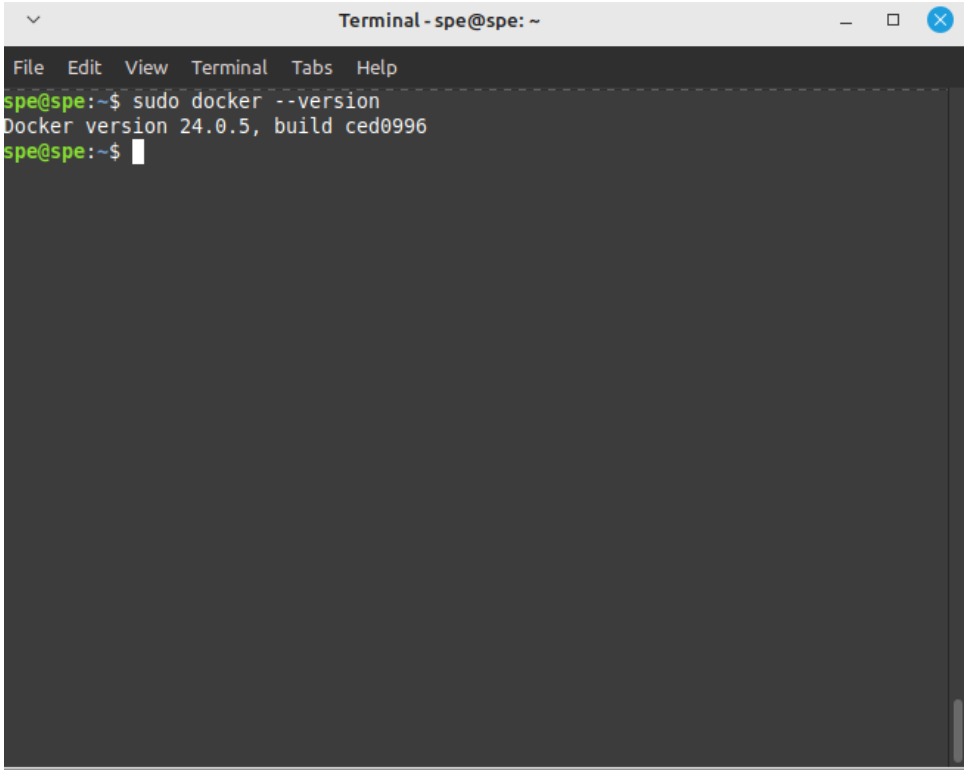


Fig 2.24 Checking Docker version post installation

Once Docker is installed, we also need to make sure that Docker is in the same user group as Jenkins. We can check it using the following command:

```
sudo tail /etc/gshadow
```

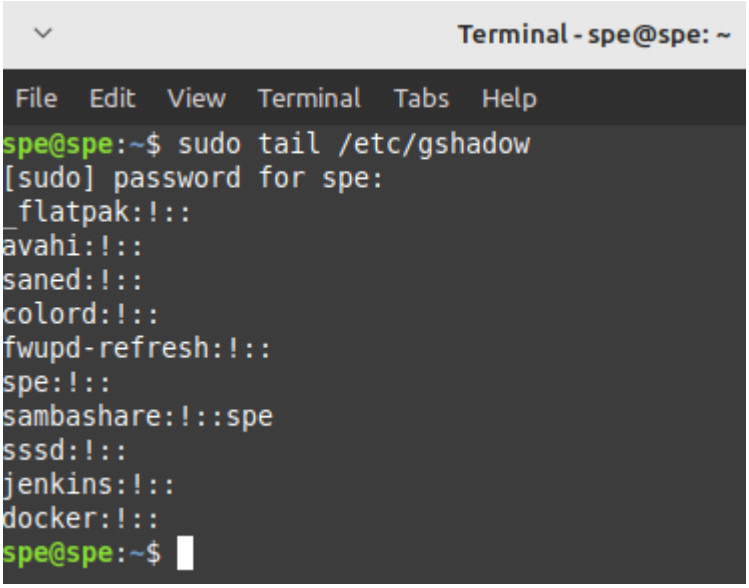


Fig 24-2 Checking if Jenkins & Docker are in the same user group

As you can see, both Jenkins and docker are not in the same line in the list. We then run the following commands to add it:

```
sudo usermod -aG docker jenkins

sudo systemctl restart jenkins
```

After running these commands, try running sudo tail /etc/gshadow to verify that both Jenkins and Docker are in the same line in the list:

```
spe@spe:~$ sudo usermod -aG docker jenkins
spe@spe:~$ sudo systemctl restart jenkins
spe@spe:~$ sudo tail /etc/gshadow
_flatpak:!::
avahi:!::
saned:!::
colord:!::
fwupd-refresh:!::
spe:!::
sambashare:!::spe
sssd:!::
jenkins:!::
docker:!::jenkins
spe@spe:~$
```

Fig 2.24.2.1 Successfully added Docker to Jenkin's user group

Creating an account in Docker Hub

Once we create Docker images, we will want to store them remotely so that they can later be pulled from the remote location to be deployed anywhere. To create an account, visit https://hub.docker.com/ and follow the sign-up procedure to create an account. Remember the username and password as we will need those credentials to be able to push and pull from Docker Hub.

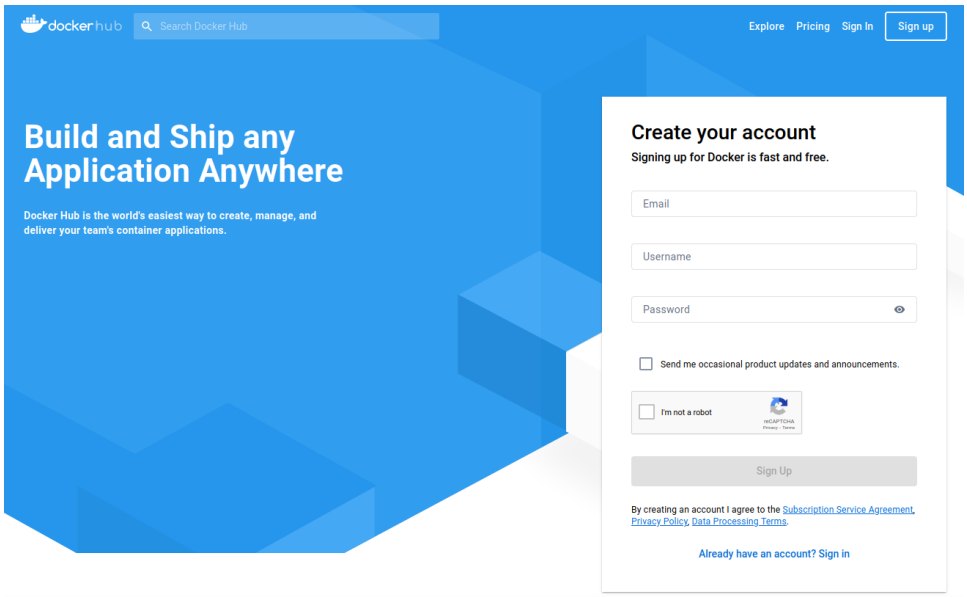


Fig 24-3 Docker Hub Sign up page

Installing and Initialising Ansible

To install Ansible, we need to add its repository using the following command:

```
sudo apt-add-repository ppa:ansible/ansible
```

Then we are going to perform apt update to refresh the links that we added from the repository:

```
sudo apt update
```

Finally, we are going to run the apt install command:

```
sudo apt install ansible
```

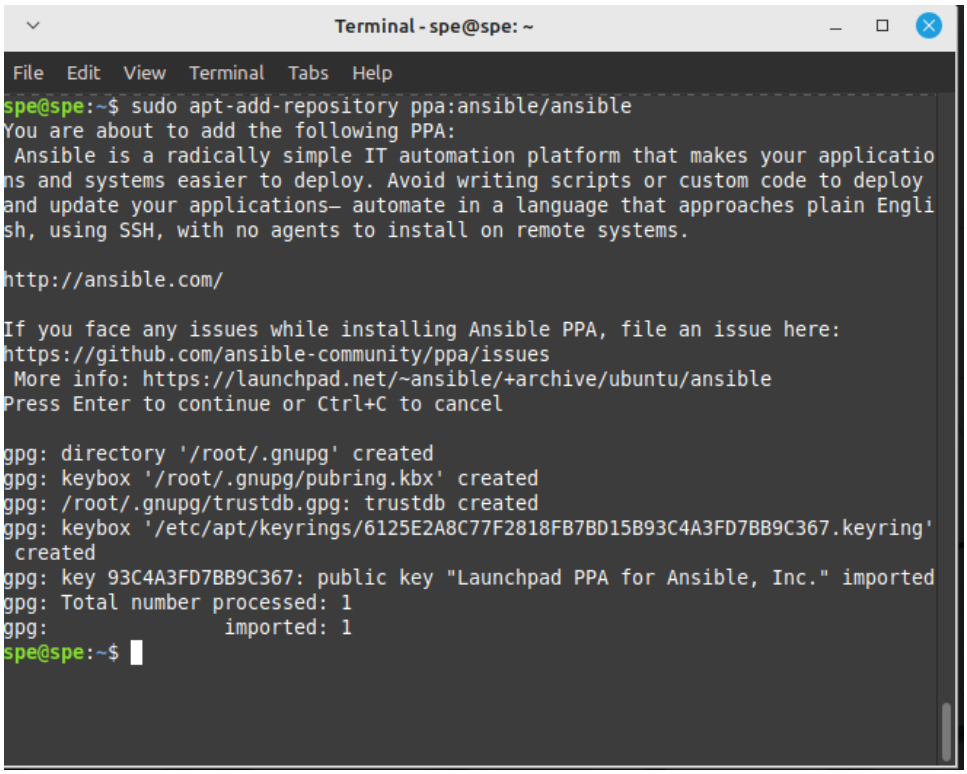


Fig 2.25 Adding Ansible Repository

```
Terminal - spe@spe: ~
File  Edit  View  Terminal  Tabs  Help
spe@spe:~$ sudo apt update
Hit:1 https://download.docker.com/linux/debian bookworm InRelease
Ign:2 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:3 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:5 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:8 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy InRelease [1
8.0 kB]
Ign:9 http://packages.linuxmint.com victoria InRelease
Get:10 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadat
a [11.2 kB]
Hit:12 http://packages.linuxmint.com victoria Release
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Met
adata [16.6 kB]
Get:15 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy/main i386 P
ackages [1,144 B]
Get:16 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy/main amd64
Packages [1,144 B]
Get:17 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy/main Transl
ation-en [752 B]
Fetched 387 kB in 3s (130 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
12 packages can be upgraded. Run 'apt list --upgradable' to see them.
spe@spe:~$
```

Fig 2.26 Updating library references after adding Ansible Repository

```
Terminal - spe@spe: ~
File  Edit  View  Terminal  Tabs  Help
spe@spe:~$ sudo apt install ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ansible-core python-babel-localedata python3-babel python3-bcrypt
  python3-jinja2 python3-jmespath python3-kerberos python3-ntlm-auth
  python3-paramiko python3-requests-kerberos python3-requests-ntlm
  python3-resolvelib python3-tz python3-winrm python3-xmltodict sshpass
Suggested packages:
  python-jinja2-doc python3-gssapi python3-invoke
The following NEW packages will be installed:
  ansible ansible-core python-babel-localedata python3-babel python3-bcrypt
  python3-jinja2 python3-jmespath python3-kerberos python3-ntlm-auth
  python3-paramiko python3-requests-kerberos python3-requests-ntlm
  python3-resolvelib python3-tz python3-winrm python3-xmltodict sshpass
0 upgraded, 17 newly installed, 0 to remove and 12 not upgraded.
Need to get 22.6 MB of archives.
After this operation, 307 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Fig 2.27 Installing Ansible

Once we install Ansible we can check its version using the following commands:

```
sudo ansible --version
```

However, a common error that pops us is the following:

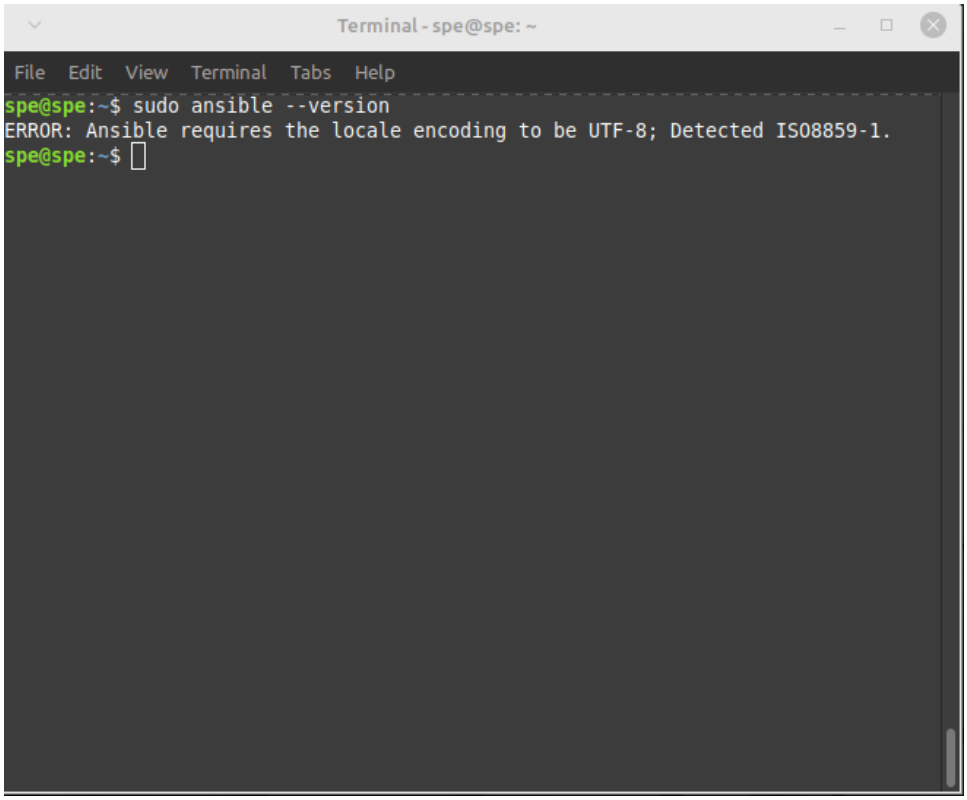


Fig 2.28 Possible error

To address this, we are going to change the locale encoding using the following commands:

```
sudo nano /etc/default/locale
```

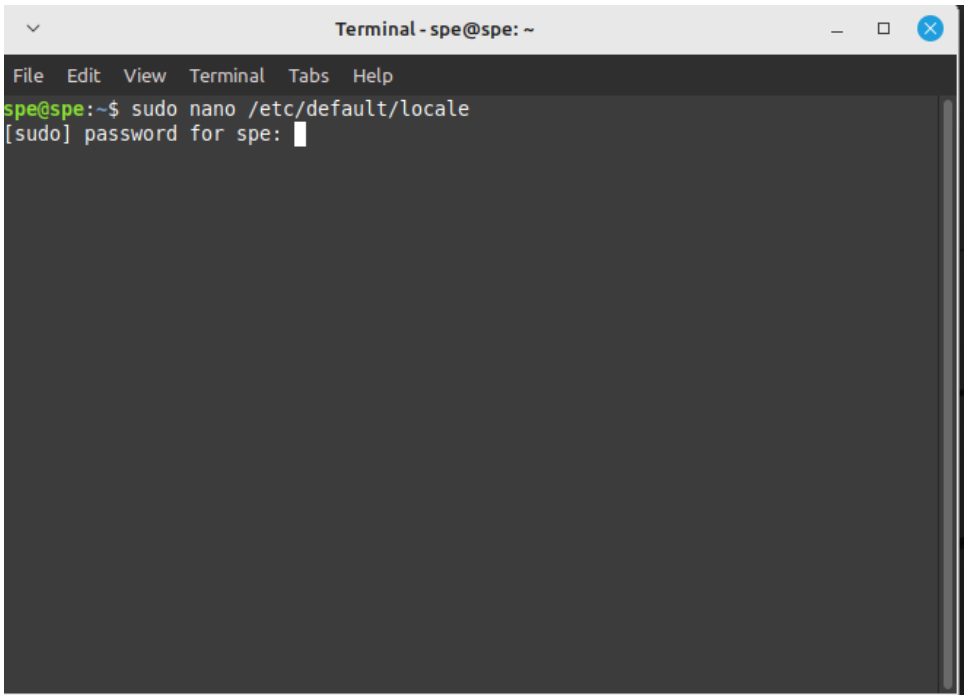


Fig 2.29 Opening locale file

This will open the locale file in the terminal. We are going to replace LANG and LC_TYPE values as follows:

```
LANG="en_US.UTF-8"

LC_CTYPE="en_US.UTF-8"
```

Before:

Fig 2.30 How locale file might look

After:

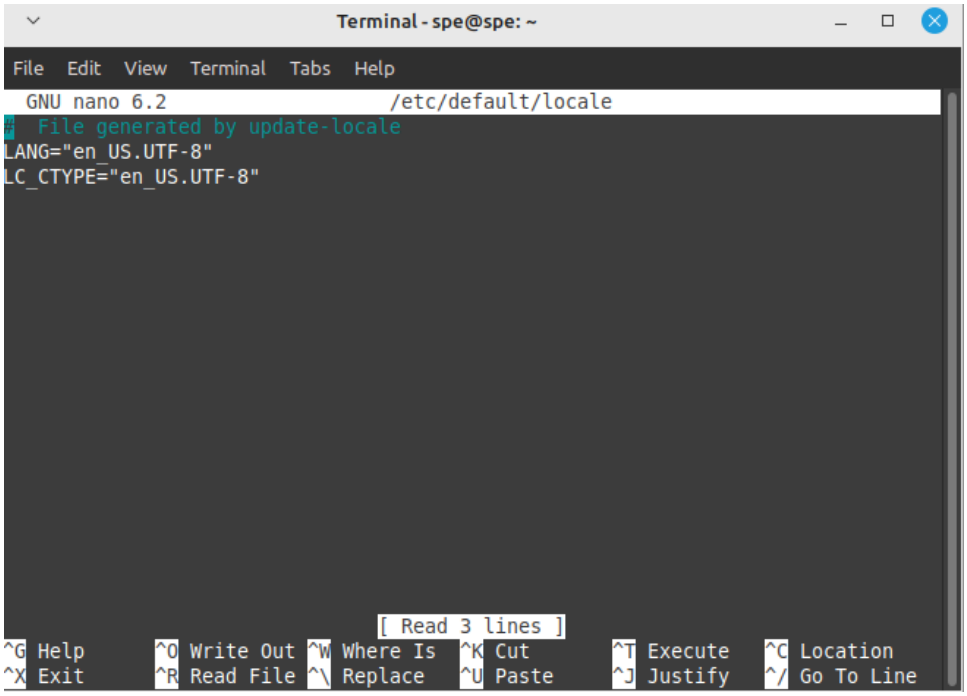


Fig 2.31 How locale file should look

Once we replace it, we perform Ctrl + O and press Enter to write the changes to the file and save it. Then we press Ctrl + X to exit.

Then we need to run the following command:

```
sudo update-locale LANG=en_US.UTF-8 LC_CTYPE=en_US.UTF-8
```

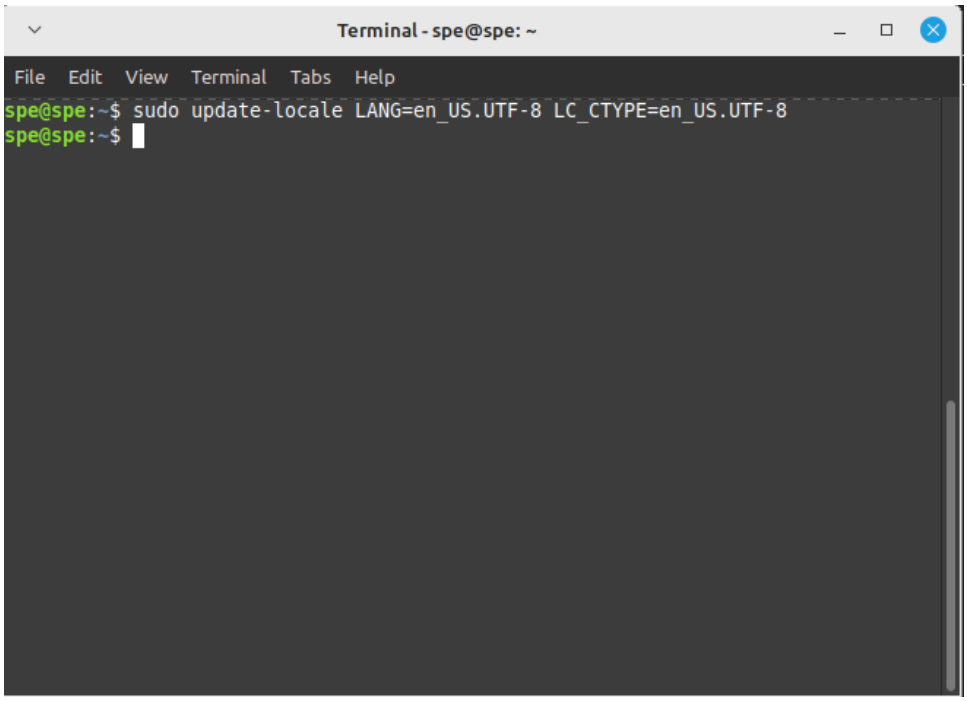


Fig 2.32 Updating locale

Once the command is executed, we are going to restart our system to let the changes take effect. Then when we try to run the command to check ansible version we will get the following output:

```
sudo ansible --version
```

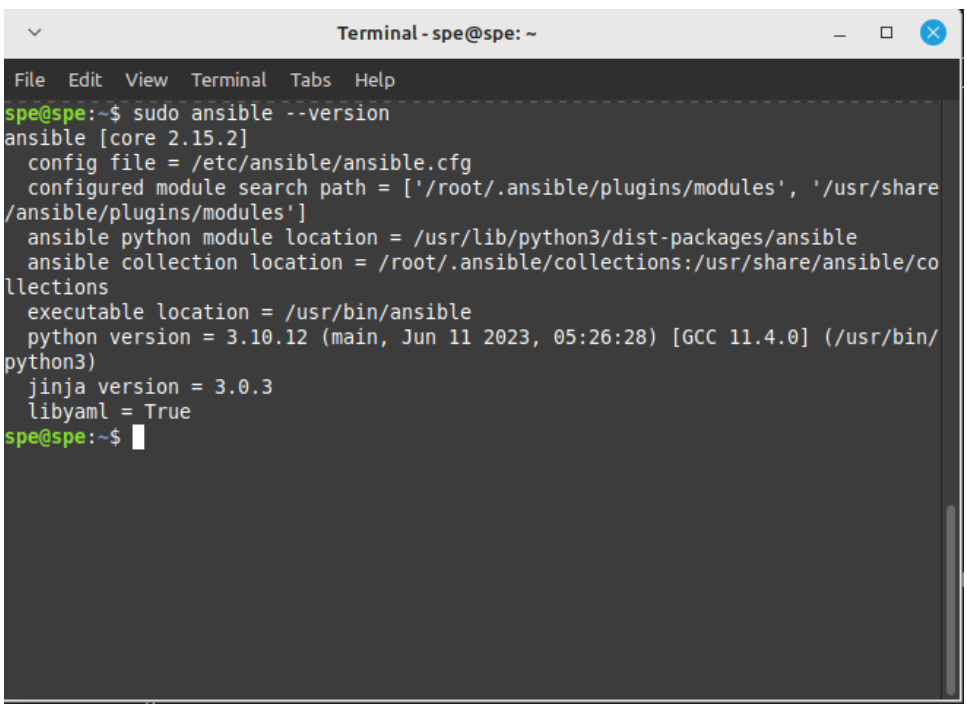


Fig 2.34 Verifying Ansible version

Installing ngrok

We visit https://ngrok.com/ and go to the download section. From there we can download the zipfile and extract it. The instructions are given in the download page so follow along. Once you have ngrok available in command line, we must sign up for a free account in nrok and visit the dashboard of our account. Over there we can see our auth token. You then use the ngrok config commands to add the authtoken in command line, thereby linking your ngrok installation with your account.

Once you have downloaded ngrok, you run the following command in the same directory that you have downloaded ngrok, to extract it to the destination:

```
sudo tar xvzf ~/Downloads/ngrok-v3-stable-linux-amd64.tgz -C
/usr/local/bin
```

Ensure you replace *ngrok-v3-stable-linux-amd64.tgz* with the appropriate version of the zip you downloaded. Once you do that, you run the following command:

```
ngrok config add-authtoken
2N3Rtt6nqyqkaADQyBwrJs6TzOW_2onhu7Vz6WJrnApNGSBah
```

Replace the auth token with what is shown in your account dashboard.

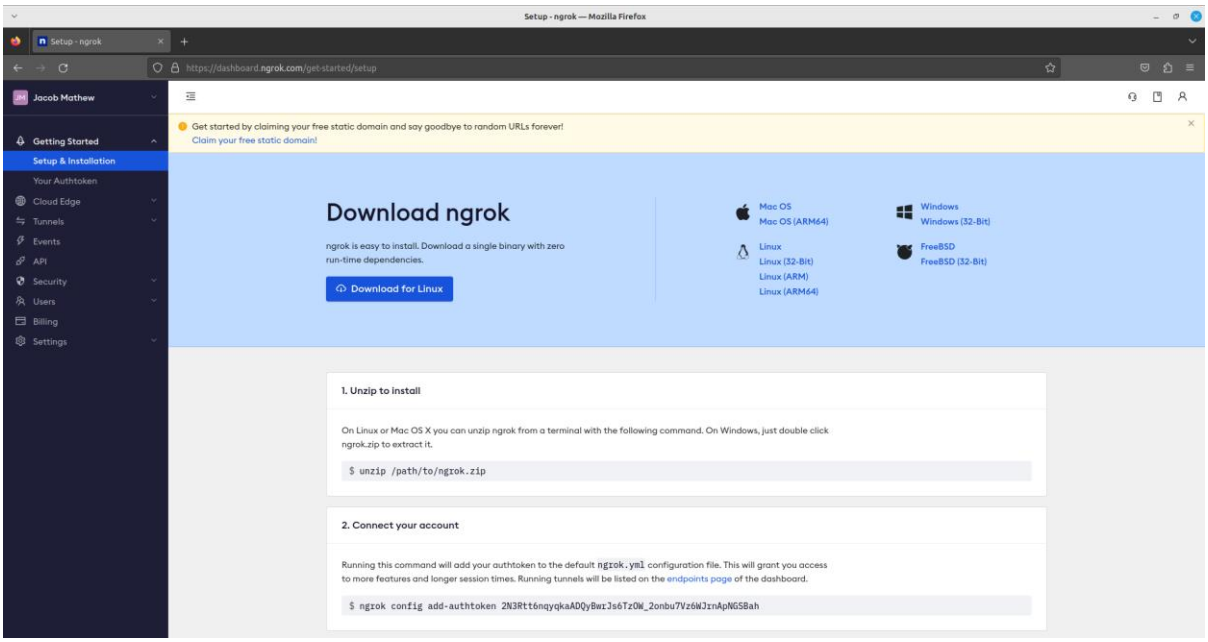


Fig 2.39 Installation guide once you sign up and reach user dashboard

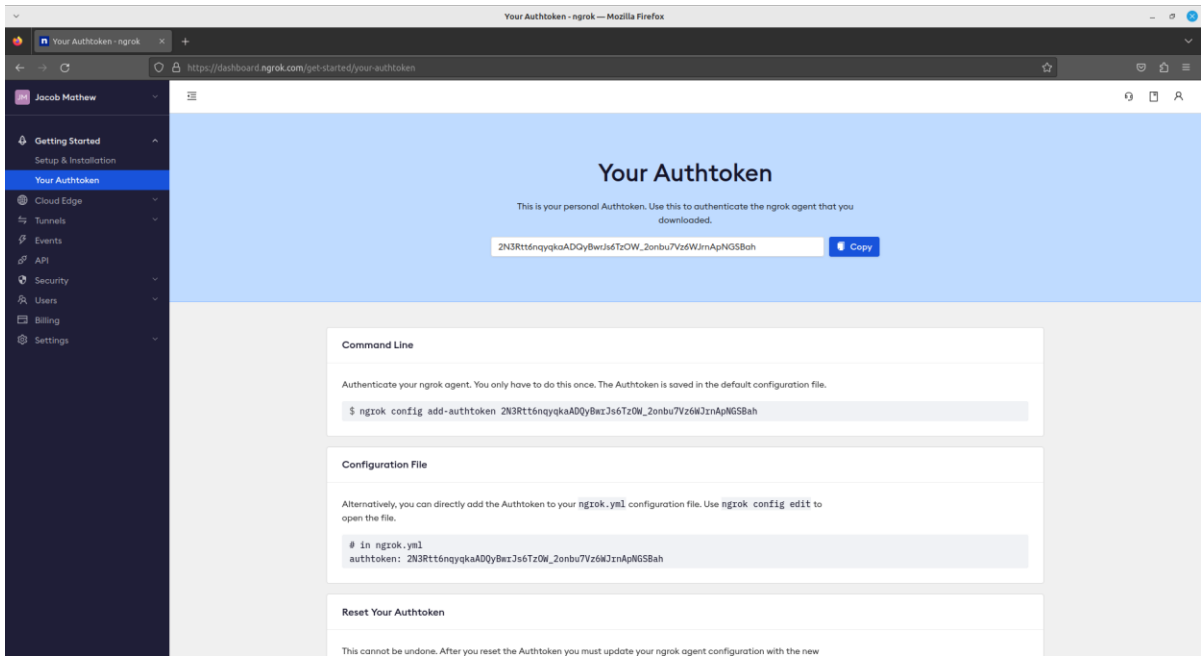


Fig 2.40 Auth token and instructions in user dashboard

Installing Elasticsearch and Kibana

First, we are going to copy the public GPG key of elastic search using the following command:

```
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch
|sudo gpg --dearmor -o /usr/share/keyrings/elastic.gpg
```

Then we add the source of elastic search so that apt will use it to install it:

```
echo "deb [signed-by=/usr/share/keyrings/elastic.gpg]
https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo
tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Fig 2.41 Adding keys and sources of elastic search

Then we perform apt update and install elastic search using the following command:

```
sudo apt update
```

```
sudo apt install elasticsearch
```

```
spe@spe:~$ sudo apt install elasticsearch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  elasticsearch
0 upgraded, 1 newly installed, 0 to remove and 98 not upgraded.
Need to get 323 MB of archives.
After this operation, 540 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 elasticsea
rch amd64 7.17.14 [323 MB]
49% [1 elasticsearch 198 MB/323 MB 61%]                          14.4 MB/s 8s
```

Fig 2.42 Installing elastic search after apt update

Once you have installed elastic search, you can run it with the following command:

```
sudo systemctl start elasticsearch
```

In case you want elastic search to run on system start up itself, then run the following command:

```
sudo systemctl enable elasticsearch
```

We can also check the status of elastic search to see if it is running or not using the following command:

```
sudo systemctl status elasticsearch
```

```
Terminal - spe@spe: ~
File  Edit  View  Terminal  Tabs  Help
spe@spe:~$ sudo systemctl start elasticsearch
spe@spe:~$ sudo systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; disabled; vendo>
     Active: active (running) since Sun 2023-10-15 12:42:01 IST; 11s ago
       Docs: https://www.elastic.co
   Main PID: 6556 (java)
      Tasks: 80 (limit: 5861)
     Memory: 2.5G
        CPU: 1min 5.014s
     CGroup: /system.slice/elasticsearch.service
             ├─6556 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.net>
             └─6747 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x>

Oct 15 12:41:20 spe systemd[1]: Starting Elasticsearch...
Oct 15 12:41:33 spe systemd-entrypoint[6556]: Oct 15, 2023 12:41:32 PM sun.util>
Oct 15 12:41:33 spe systemd-entrypoint[6556]: WARNING: COMPAT locale provider w>
Oct 15 12:42:01 spe systemd[1]: Started Elasticsearch.
lines 1-16/16 (END)
```

Fig 2.43 Starting and status of elastic search

Now that we have install elastic search, we can install kibana by running:

```
sudo apt install kibana
```

Again, to start kibana we can use the following command:

```
sudo systemctl start kibana
```

And to check if it is running or not, we will use the following command:

```
sudo systemctl status kibana
```

```
spe@spe:~$ sudo apt install kibana
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  kibana
0 upgraded, 1 newly installed, 0 to remove and 98 not upgraded.
Need to get 298 MB of archives.
After this operation, 769 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 kibana amd
64 7.17.14 [298 MB]
47% [1 kibana 176 MB/298 MB 59%]                                6,273 kB/s 19s
```

Fig 2.44 Installing Kibana

```
spe@spe:~$ sudo systemctl start kibana
spe@spe:~$ sudo systemctl status kibana
● kibana.service - Kibana
     Loaded: loaded (/etc/systemd/system/kibana.service; disabled; vendor prese>
     Active: active (running) since Sun 2023-10-15 12:52:04 IST; 4s ago
       Docs: https://www.elastic.co
   Main PID: 7475 (node)
      Tasks: 7 (limit: 5861)
     Memory: 118.9M
        CPU: 2.855s
     CGroup: /system.slice/kibana.service
             └─7475 /usr/share/kibana/bin/../node/bin/node /usr/share/kibana/bi>

Oct 15 12:52:04 spe systemd[1]: Started Kibana.
Oct 15 12:52:04 spe kibana[7475]: Kibana is currently running with legacy OpenS>
lines 1-13/13 (END)
```

Fig 2.45 Starting and checking Kibana status

Once Kibana is running, we are going to setup a username and password for kibana. The user is going to be **kadmin**. Once you run the following command, it will prompt you for a password which you will enter as well:

```
echo "kadmin:`openssl passwd -apr1`" | sudo tee -a
/etc/nginx/htpasswd.users
```

```
spe@spe:~$ echo "kadmin:`openssl passwd -apr1`" | sudo tee -a /etc/nginx/htpassw
d.users
Password: tee: /etc/nginx/htpasswd.users: No such file or directory

Verifying - Password:
kadmin:$apr1$yWxq5Vru$jiLFfPNscdMpiWcsSjQsc1
spe@spe:~$
```

Fig 2.46 Initialising Kibana user and password

Visit http://localhost:5601/ to access the Kibana dashboard and http://localhost:5601/status to see the status of your Kibana service.

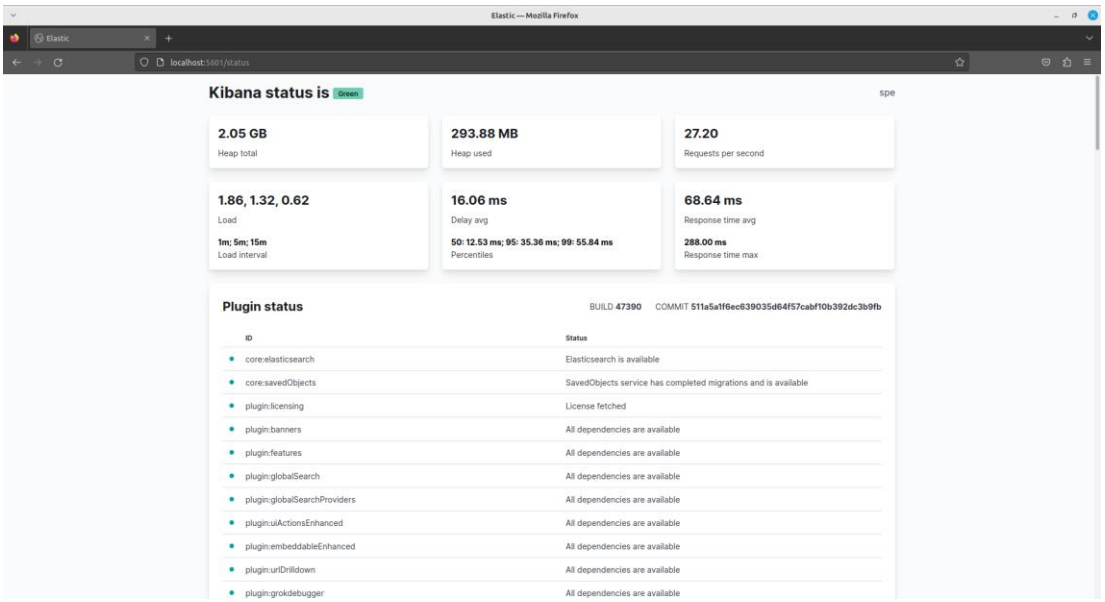


Fig 2.47 Kibana status

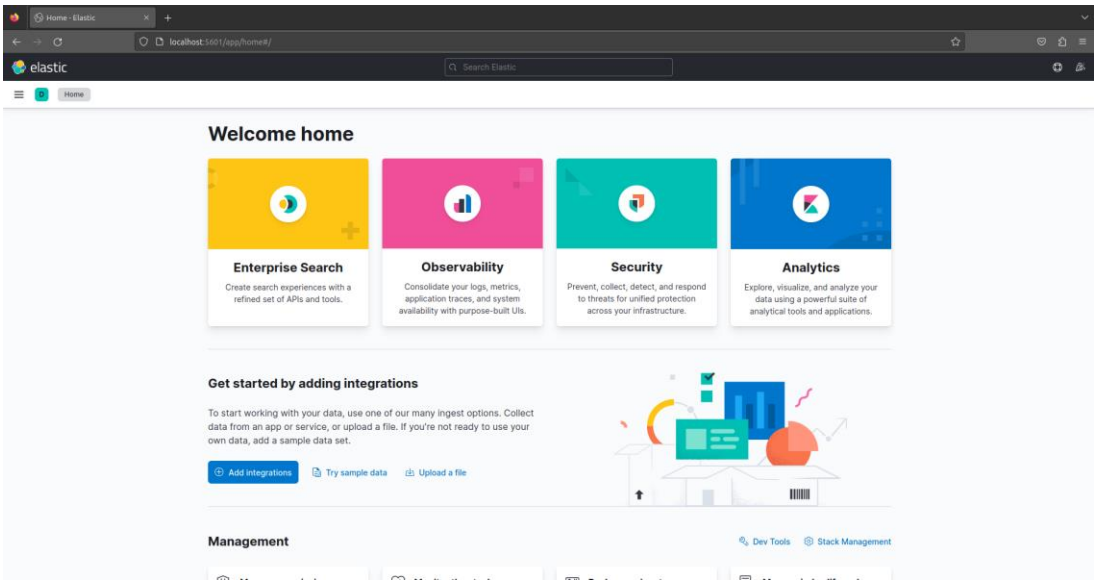


Fig 2.48 Kibana dashboard

In case you face 503 error, then ensure that elastic search and kibana services are running. In case you face an error starting elastic search then try to manually set the localhost IP and port in the YML file using the following command:

```
sudo nano /etc/elasticsearch/elasticsearch.yml
```

In this file, add the following lines:

```
network.host: 127.0.0.1
http.port: 9200
```

Save and exit, then try to launch elastic search service again.

Closing thoughts

We have installed all the generous size tools that we need for the following chapters. However, there are a few more things like Maven which we have not installed as they need to be configured to work with the project that you are working on. Hence the book will go over the installation of any other tools or packages as and when they are needed.