**Installing OpenJDK 8**

Java 8, the previous Java LTS version, is still widely used. If your application runs on Java 8, you can install it by typing the following commands:

sudo apt updatesudo apt install openjdk-8-jdk

Verify the installation by checking the Java version:

java -version

The output should look something like this:

openjdk version "1.8.0\_252"

OpenJDK Runtime Environment (build 1.8.0\_252-8u252-b09-1ubuntu1-b09)

OpenJDK 64-Bit Server VM (build 25.252-b09, mixed mode)

**Setting the Default Version**

If you have multiple Java versions installed on your Ubuntu system you can check which version is set as the default one by typing:

java -version

To change the default version, use the update-alternatives command:

sudo update-alternatives --config java

The output will look something like below:

There are 2 choices for the alternative java (providing /usr/bin/java).

Selection Path Priority Status

------------------------------------------------------------

\* 0 /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111 auto mode

1 /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111 manual mode

2 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java 1081 manual mode

Press <enter> to keep the current choice[\*], or type selection number:

You will be presented with a list of all installed Java versions. Enter the number of the version you want to be used as the default and press Enter.

You can uninstall Java like any other package installed with [apt](https://linuxize.com/post/how-to-use-apt-command/) .

For example, to uninstall the default-jdk package enter:

sudo apt remove openjdk-11-jdk

**Install Maven via Download**

It is possible that the packaged version of Maven provided by the Ubuntu repository is not the most up-to-date version. To obtain the most up-to-date version of Maven, it can be installed after being downloaded from the [Apache Maven website.](https://maven.apache.org/download.cgi)

Step #1: Download Maven

Download Maven into the /tmp directory utilizing the following command:

wget https://downloads.apache.org/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz -P /tmp

As of the writing of this article version, 3.6.3 is the latest version of Maven. To check if there is a newer version, visit the [Maven download page](http://apache.spinellicreations.com/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz).

Step #2: Extract Maven and Symlink

Next, extract the newly downloaded Maven archive to the /opt directory:

root@ubuntu:~# tar xf /tmp/apache-maven-3.6.3-bin.tar.gz -C /opt

To make it more straightforward to upgrade in the future, create a symlink to the Maven installation directory:

root@ubuntu:~# ln -s /opt/apache-maven-3.6.3 /opt/maven

In the future, to upgrade Maven, download the newer version, extract it, and point the symlink to the new version.

Step #3: Configure Environment Variables

Create a maven.sh file inside the /etc/profile.d/ directory:

root@ubuntu:~# vim /etc/profile.d/maven.sh

Copy the following configuration out and paste it into the maven.sh file:

export JAVA\_HOME= /usr/lib/jvm/java-1.8.0-openjdk-amd64

export M2\_HOME=/opt/maven

export MAVEN\_HOME=/opt/maven

export PATH=${M2\_HOME}/bin:${PATH}

Finally, to get these variables into play in the environment, run the source command on the file:

root@ubuntu:~# source /etc/profile.d/maven.sh

Step #4: Verify Maven Installation

To verify Maven is installed properly, run the version command and check the output:

root@ubuntu:~# mvn -version

Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)

Maven home: /opt/maven

Java version: 11.0.7, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64

Default locale: en, platform encoding: UTF-8

OS name: "linux", version: "4.15.0-91-generic", arch: "amd64", family: "unix"

That’s it! Maven is now installed. It’s now possible to use Maven to help make project management and definition simpler.

# How to Install Jenkins on Ubuntu 20.04

Posted on [January 11, 2021](https://www.liquidweb.com/kb/how-to-install-jenkins-on-ubuntu-20-04/) by [Margaret Fitzgerald](https://www.liquidweb.com/kb/author/mfitzgerald/)  
Category: [Tutorials](https://www.liquidweb.com/kb/category/tutorials/) | Tags: [Ansible](https://www.liquidweb.com/kb/tag/ansible/), [Automation](https://www.liquidweb.com/kb/tag/automation/), [CentOS](https://www.liquidweb.com/kb/tag/centos/), [Chef](https://www.liquidweb.com/kb/tag/chef/), [Chef-Manage](https://www.liquidweb.com/kb/tag/chef-manage/), [CI/CD](https://www.liquidweb.com/kb/tag/ci-cd/), [Cloud](https://www.liquidweb.com/kb/tag/cloud/), [Code](https://www.liquidweb.com/kb/tag/code/), [Container](https://www.liquidweb.com/kb/tag/container/), [Containerization](https://www.liquidweb.com/kb/tag/containerization/), [Containers](https://www.liquidweb.com/kb/tag/containers/), [Cookbook](https://www.liquidweb.com/kb/tag/cookbook/), [Cookbooks](https://www.liquidweb.com/kb/tag/cookbooks/), [Deploy](https://www.liquidweb.com/kb/tag/deploy/), [Deployment](https://www.liquidweb.com/kb/tag/deployment/), [Developer](https://www.liquidweb.com/kb/tag/developer/), [Development](https://www.liquidweb.com/kb/tag/development/), [DevOps](https://www.liquidweb.com/kb/tag/devops/), [Docker](https://www.liquidweb.com/kb/tag/docker/), [Git](https://www.liquidweb.com/kb/tag/git/), [GitHub](https://www.liquidweb.com/kb/tag/github/), [Helm](https://www.liquidweb.com/kb/tag/helm/), [IaC](https://www.liquidweb.com/kb/tag/iac/), [Infrastructure](https://www.liquidweb.com/kb/tag/infrastructure/), [Infrastructure as Code](https://www.liquidweb.com/kb/tag/infrastructure-as-code/), [Jenkins](https://www.liquidweb.com/kb/tag/jenkins/), [Kubelet](https://www.liquidweb.com/kb/tag/kubelet/), [Kubernetes](https://www.liquidweb.com/kb/tag/kubernetes/), [Linux](https://www.liquidweb.com/kb/tag/linux/), [MiniKube](https://www.liquidweb.com/kb/tag/minikube/), [Modules](https://www.liquidweb.com/kb/tag/modules/), [Monitoring](https://www.liquidweb.com/kb/tag/monitoring/), [Nagios](https://www.liquidweb.com/kb/tag/nagios/), [Nodes](https://www.liquidweb.com/kb/tag/nodes/), [Open Source](https://www.liquidweb.com/kb/tag/open-source/), [Organization](https://www.liquidweb.com/kb/tag/organization/), [Parent](https://www.liquidweb.com/kb/tag/parent/), [Parent Server](https://www.liquidweb.com/kb/tag/parent-server/), [Platform](https://www.liquidweb.com/kb/tag/platform/), [Prometheus](https://www.liquidweb.com/kb/tag/prometheus/), [Provisioning](https://www.liquidweb.com/kb/tag/provisioning/), [Puppet](https://www.liquidweb.com/kb/tag/puppet/), [Remote Management](https://www.liquidweb.com/kb/tag/remote-management/), [Salt](https://www.liquidweb.com/kb/tag/salt/), [SaltStack](https://www.liquidweb.com/kb/tag/saltstack/), [Server Administration](https://www.liquidweb.com/kb/tag/server-administration/), [Tools](https://www.liquidweb.com/kb/tag/tools/), [Vagrant](https://www.liquidweb.com/kb/tag/vagrant/), [Virtual Machines](https://www.liquidweb.com/kb/tag/virtual-machines/), [VirtualBox](https://www.liquidweb.com/kb/tag/virtualbox/), [Virtualization](https://www.liquidweb.com/kb/tag/virtualization/), [YAML](https://www.liquidweb.com/kb/tag/yaml/)

Reading Time: 7 minutes

By mfitzgerald



## Introduction

In this tutorial, we are going to examine [Jenkins](https://www.liquidweb.com/kb/how-to-install-jenkins-on-centos-8/) and what its purpose is. We will begin by installing it on an Ubuntu 20.04 server and then start the software, enable the service, configure it, and finally, create a user with administrative privileges.

## What is Jenkins?

[Jenkins](https://www.liquidweb.com/kb/installing-jenkins-on-ubuntu-16-04/) is an open-source program written in the Java programming language. It is designed to provide stable and balanced Continuous Integration and Continuous Deployment (CI/CD) processes.

## Why is it Needed?

Let’s try to understand why this type of software is needed. Previously, when programmers wrote code and created applications, the next steps were to:

* Build a cross-platform application that can be utilized on and transferred to any server.
* Perform the necessary performance testing and evaluation.
* Transfer, install, and configure the application on the servers.

These are just some of the tasks that Jenkins solves. These steps can now be automated to stabilize and speed up the development process while reducing human error. Because of this, Jenkins is one of the most widely used [CI/CD solutions](https://www.liquidweb.com/kb/the-best-devops-tools-for-infrastructure-automation/) in use today.

## Advantages

Jenkins is famous for such advantages as:

* Increasing the reliability of the CI/CD process.
* Ease of installation, configuration, and usage of the software.
* A wide variety of Plugins which extends its capabilities.
* Integration with other [CI/CD tools](https://www.liquidweb.com/kb/devops-a-new-perspective-on-shared-automation/).
* Distribution of services. A developer can build, test, and deploy applications across multiple servers and platforms.
* Being a free and open-source application.

## Prerequisites

To install Jenkins on Ubuntu 20.04, the following requirements must be met.

* A server with at least 2 GB RAM and 2 cores.
* The Ubuntu 20.04 OS installed.
* All commands are executed as the root user. If you are a normal user, you must have access to and use the sudo command.

## Installation

First, we should always update Ubuntu and all installed applications.

root@host:~# apt update &&  apt upgrade -y

### Install JDK

In order to install, configure, and use Jenkins, we need the Java Development Kit (JDK) installed. This tutorial will use JDK version 11 since Jenkins can use both JDK 8 and 11 versions by default. Let’s begin by installing the latest version of the JDK software.

First, we check if we already have the [JDK software installed](https://www.liquidweb.com/kb/how-to-install-openjdk-11-on-centos-8/) on the server using the version command.

root@host:~# java –version

-bash: java: command **not** found

root@host:~#

Now we can install the JDK.

root@host:~# apt install openjdk-11-jdk -y

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following additional packages will be installed:

  at-spi2-core ca-certificates-java fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common libatk-bridge2.0-0

  libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data

  libavahi-common3 libcups2 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libgif7

  libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2

  libllvm10 libnspr4 libnss3 libpciaccess0 libpcsclite1 libpthread-stubs0-dev libsensors-config libsensors5 libsm-dev libsm6

  libvulkan1 libwayland-client0 libx11-6 libx11-data libx11-dev libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0

  libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-sync1 libxcb1 libxcb1-dev libxcomposite1 libxdamage1

  libxdmcp-dev libxext6 libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxmuu1 libxpm4 libxrandr2 libxrender1

  libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 mesa-vulkan-drivers openjdk-11-jdk-headless

  openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev

Suggested packages:

  default-jre cups-common libice-doc liblcms2-utils pcscd lm-sensors libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-11-demo

  openjdk-11-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei

  fonts-indic mesa-utils

The following NEW packages will be installed:

  at-spi2-core ca-certificates-java fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common libatk-bridge2.0-0

  libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data

  libavahi-common3 libcups2 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libgif7

  libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2

  libllvm10 libnspr4 libnss3 libpciaccess0 libpcsclite1 libpthread-stubs0-dev libsensors-config libsensors5 libsm-dev libsm6

  libvulkan1 libwayland-client0 libx11-6 libx11-data libx11-dev libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0

  libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-sync1 libxcb1 libxcb1-dev libxcomposite1 libxdamage1

  libxdmcp-dev libxext6 libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxmuu1 libxpm4 libxrandr2 libxrender1

  libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 mesa-vulkan-drivers openjdk-11-jdk

  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils x11proto-core-dev x11proto-dev

  xorg-sgml-doctools xtrans-dev

0 upgraded, 93 newly installed, 0 to remove **and** 0 **not** upgraded.

Need to get 309 MB of archives.

After this operation, 815 MB of additional disk space will be used.

…

…

…

done.

done.

root@host:~#

### Verify Java Version.

Lastly, we verify the Java version using this command.

root@host:~# java -version

openjdk version "11.0.9.1" 2020-11-04

OpenJDK Runtime Environment (build 11.0.9.1+1-Ubuntu-120.04)

OpenJDK 64-Bit Server VM (build 11.0.9.1+1-Ubuntu-120.04, mixed mode)

root@host:~#

## Install Jenkins

Next, we install Jenkins itself. The Jenkins version contained in the standard Ubuntu packages is different from the latest available version. Therefore, we will install the [newest version](https://pkg.jenkins.io/debian/) from the official site.

First, we add the gpg key to verify the authenticity of the software from the repository.

root@host:~# wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | apt-key add -

OK

root@host:~#

After this, we add the repository address to our */etc/apt/sources.list.d* file. This list contains information on where to download and update the application.

root@host:~# sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \

e> /etc/apt/sources.**list**.d/jenkins.**list**'

root@host:~#

Next, let’s update our package list again so that the apt package manager can find the software new repository.

root@host:~# apt-get **update**

Hit:1 http://archive.ubuntu.**com**/ubuntu focal InRelease

Hit:2 http://archive.ubuntu.**com**/ubuntu focal-updates InRelease

Hit:3 http://archive.ubuntu.**com**/ubuntu focal-backports InRelease

Hit:4 http://security.ubuntu.**com**/ubuntu focal-security InRelease

Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease

Get:6 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]

Get:7 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]

Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [18.9 kB]

Fetched 21.8 kB in 1s (38.1 kB/s)

Reading package lists... Done

root@host:~#

Now we can install Jenkins.

root@host:~# apt install jenkins -**y**

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following additional packages will **be** installed:

  daemon net-tools

The following NEW packages will **be** installed:

  daemon jenkins net-tools

0 upgraded, 3 newly installed, 0 **to** remove and 0 not upgraded.

Need **to** get 67.2 MB of archives.

After this operation, 68.5 MB of additional disk space will **be** used.

Get:1 http://archive.ubuntu.**com**/ubuntu focal/universe amd64 daemon amd64 0.6.4-1build2 [96.3 kB]

Get:2 http://archive.ubuntu.**com**/ubuntu focal/main amd64 net-tools amd64 1.60+git20180626.aebd88e-1ubuntu1 [196 kB]

Get:3 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.263.1 [66.9 MB]

Fetched 67.2 MB in 9s (7,112 kB/s)

Selecting previously unselected package daemon.

(Reading database ... 50086 **files** and directories currently installed.)

Preparing **to** unpack .../daemon\_0.6.4-1build2\_amd64.**deb** ...

Unpacking daemon (0.6.4-1build2) ...

Selecting previously unselected package net-tools.

Preparing **to** unpack .../net-tools\_1.60+git20180626.aebd88e-1ubuntu1\_amd64.**deb** ...

Unpacking net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...

Selecting previously unselected package jenkins.

Preparing **to** unpack .../jenkins\_2.263.1\_all.**deb** ...

Unpacking jenkins (2.263.1) ...

Setting **up** net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...

Setting **up** daemon (0.6.4-1build2) ...

Setting **up** jenkins (2.263.1) ...

Processing triggers **for** man-db (2.9.1-1) ...

Processing triggers **for** systemd (245.4-4ubuntu3.3) ...

root@host:~#

### Start and Verify the Service

We have installed Jenkins. Now we will start it using the systemctl system command.

root@host:~# systemctl start jenkins

root@host:~#

Next, we check to see if everything has started correctly using the systemctl status command.

**root**@**host**:~# **systemctl** **status** **jenkins**

● **jenkins**.service **-** **LSB**: **Start** **Jenkins** **at** **boot** **time**

**Loaded**: **loaded** (/etc/init.d/jenkins; generated)

**Active**: **active** (exited) **since** **Sat** **2020-12-19** **19**:13:43 **UTC**; **4min** **35s** **ago**

**Docs**: **man**:systemd-sysv-generator(8)

**Tasks**: **0** (limit: 4620)

**Memory**: **0B**

**CGroup**: /**system**.slice/**jenkins**.service

**Dec** **19** **19**:13:42 **host** **systemd**[1]: **Starting** **LSB**: **Start** **Jenkins** **at** **boot** **time**...

**Dec** **19** **19**:13:42 **host** **jenkins**[54912]: **Correct** **java** **version** **found**

**Dec** **19** **19**:13:42 **host** **jenkins**[54912]:  \* **Starting** **Jenkins** **Automation** **Server** **jenkins**

**Dec** **19** **19**:13:42 **host** **su**[54967]: (to jenkins) **root** **on** **none**

**Dec** **19** **19**:13:42 **host** **su**[54967]: **pam\_unix**(su-l:session): **session** **opened** **for** **user** **jenkins** **by** (uid=0)

**Dec** **19** **19**:13:42 **host** **su**[54967]: **pam\_unix**(su-l:session): **session** **closed** **for** **user** **jenkins**

**Dec** **19** **19**:13:43 **host** **jenkins**[54912]:    ...done.

**Dec** **19** **19**:13:43 **host** **systemd**[1]: **Started** **LSB**: **Start** **Jenkins** **at** **boot** **time**.

**root**@**host**:~#

Here we can see the status

Active: active (exited) since Sat 2020-12-19 19:13:43 UTC; 4min 35s ago

## Configure Firewall

So, now that our software is installed and has been launched successfully, we will begin the configuration of the server. When Jenkins is running, we must open port 8080 in the firewall so that we can access Jenkins through the browser. First, activate the [UFW firewall](https://www.liquidweb.com/kb/installing-using-ufw-ubuntu-16-04-lts/).

root@host:~# ufw enable

Firewall is active **and** enabled on system startup

root@host:~#

Next, add a rule to open and use port 8080 in the firewall.

root@host:~# ufw allow 8080

Rule added

Rule added (v6)

root@host:~#

Now, let’s check the status of the firewall.

root@host:~# ufw status

Status: active

To                         Action      From

--                         ------      ----

8080                       ALLOW       Anywhere

OpenSSH                    ALLOW       Anywhere

8080 (v6)                  ALLOW       Anywhere (v6)

OpenSSH (v6)               ALLOW       Anywhere (v6)

root@host:~#

## Set up Jenkins Users

Now let’s start setting up our main user. Go to a browser and enter **http://YOUR\_IP\_OR\_DOMAIN:8080**

**YOUR\_IP\_OR\_DOMAIN –**list your domain name or IP address here. If you don’t know your IP address, you can verify it using the following command.

root@host:~# ip a

When you enter the url into the browser, you will see the Unlock Jenkins screen.

Here we need to enter the password. To obtain the password, run the following command in the terminal to locate the password that is stored in the *initialAdminPassword* file.

root@host:~# cat /var/lib/jenkins/secrets/initialAdminPassword

9654\*\*\*\*\*\*\*d9549b\*\*\*\*\*\*2195\*\*\*\*\*\*18

root@host:~#

Copy that password that the command output received in the terminal, and enter it in the Administrator password window below and click continue.

[Home](https://linoxide.com/) » [Ubuntu](https://linoxide.com/category/ubuntu-how-to/) » How to Install Terraform on Ubuntu 20.04

# How to Install Terraform on Ubuntu 20.04

Updated January 20, 2021

by [Pratik Gautam](https://linoxide.com/author/pratikg/" \o "View all posts by Pratik Gautam) Categories[Ubuntu](https://linoxide.com/category/ubuntu-how-to/)

[Terraform](https://www.terraform.io/) is an infrastructure as a code platform developed by HashiCorp. You can simply write code in the human-readable format following HashiCorp Configuration Language (HCL) and deploy it to get the infrastructure in the cloud. Terraform is supported in many cloud providers like Google, Amazon, Alibaba, etc.

Here in this article, we are going to **install** the latest version of **terraform** on Ubuntu. We are performing terraform installation on Ubuntu 20.04 however you can do the same procedure on all Linux platforms.

Also, learn how to **use terraform** with simple example by launch an ec2 instance and create s3 bucket.

## Install terraform on Ubuntu 20.04

[Download](https://www.terraform.io/downloads.html) the latest version of Terrafrom. At the time of writing article, the latest version is 0.14.3 .

$ wget https://releases.hashicorp.com/terraform/0.14.3/terraform\_0.14.3\_linux\_amd64.zip

Now, unzip the download file.

$ sudo apt install zip -y

$ sudo unzip terraform\_0.14.3\_linux\_amd64.zip

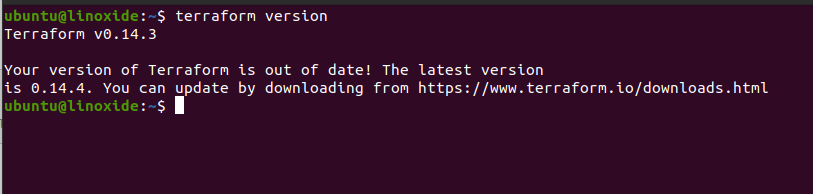
This will output you a terraform file just move it to /usr/local/bin/ to execute the command.

Moto Z4 Hands On

$ sudo mv terraform /usr/local/bin/

Check the version

$ terraform version

Terraform version

## How to use Terraform

Let's explain how to use terraform with basic examples.

## Launching ec2 instance and creating an s3 bucket

I would like to create a folder and do everything inside it.

$ mkdir aws && cd aws

Create a **configuration file** for terraform having extension 'tf'

vi configuration.tf

Now you need to provide the following information:

* Provider: Cloud provider, AWS in our case
* Access, Secret key: Access to AWS resources
* Region: The region where you are going to provision your infrastructure. I am doing it in Oregon.

On the second block of the code define AWS instance, ie ami ( check amazon EC2 [AMI Locator](https://cloud-images.ubuntu.com/locator/ec2/) ), instance type, and tag.

The last part of the following code will create s3 bucket named 'bucket-launched-using-terrafrom-20210106'. Remember that the bucket name must be unique over the AWS.

Copy the following content paste in the file configuration.tf file. Provide access, secret key, region, bucket name of your own.

**#Define keys and region**

provider "aws" {

access\_key = "YOUR-ACCESS-KEY"

secret\_key = "YOUR-SECRET-KEY"

region = "us-west-2"

}

**#Define ec2 instance**

resource "aws\_instance" "instance1" {

ami = "ami-089668cd321f3cf82"

instance\_type = "t2.micro"

tags = {

Name = "ubuntu-20.04"

}

}

**#Define s3 bucket**

resource "aws\_s3\_bucket" "bucket1" {

bucket = "bucket-launched-using-terrafrom-20210106"

acl = "private" # or can be "public-read"

tags = {

Name = "Bucket"

Environment = "Production"

}

}

Now, initialize, plan, and execute your code. **Initializing terraform** will make necessary configuration, planning is like you see what will happen in actuality. It will also find out your syntax error. And, finally applying means you will deploy the code in the cloud. Let's execute the following command one by one.

Initializing terraform will make the necessary configuration. So, execute the following command,

$ terraform init

Planning is like you see what will happen in actuality. It will also find out your syntax error.

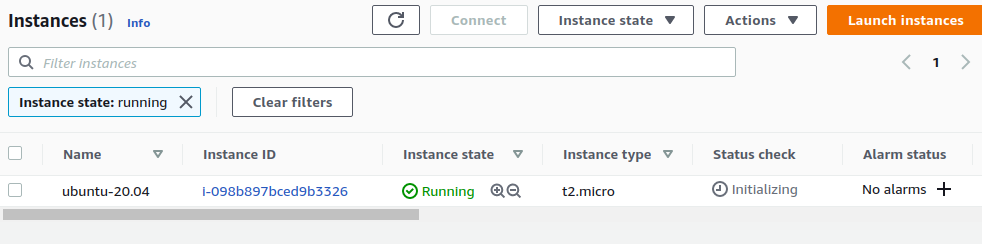
$ terraform plan

Applying means you will **deploy** the code in the cloud. Do it just by executing the command,

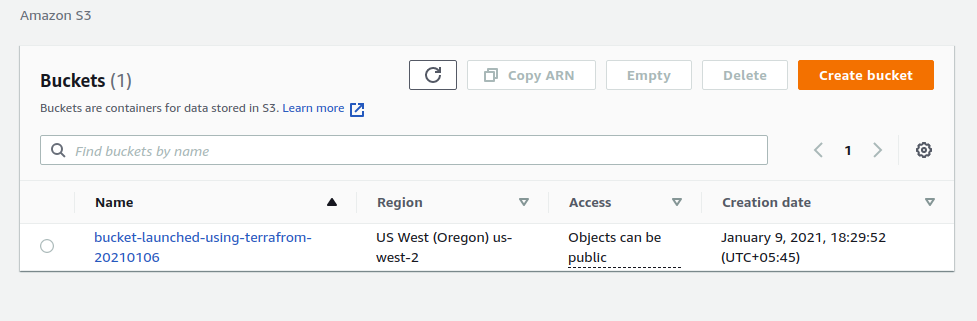
$ terraform apply

It will ask you for confirmation. Just type 'yes' and hit enter. Within few second your infrastructure will be ready.

You can now login to the AWS console and goto the service ec2. You will find ec2 is launched.

ec2 instance launched from terraform script

Similarly, goto s3 and search your bucket.

S3 bucket launched from terraform script

## Destroy infrastructure

If you want to **destroy** the above infrastructure, you can simply type 'terraform destroy'.

$ terraform destroy

It asks you for the confirmation just say 'yes' and hit enter. The ec2 instance and s3 bucket you create above should be removed from your AWS account. You may verify by logging into the AWS console.

## Remove terraform

If you want to **remove** terraform you can simply delete the 'terraform' file kept at /usr/local/bin/

$ sudo rm -f /usr/local/bin/terraform

Also, you can clean your directory where you have initialized terraform. In our case, we can delete all the content of the folder 'aws'.

Related Read:

* [How to Upload Files to S3 using Terraform](https://linoxide.com/tools/upload-files-to-s3-using-terraform/)
* [Provision AWS EC2 Cloud Instance](https://linoxide.com/devops/install-terraform-provision-aws-ec2-instance/)

## Conclusion

This article has simply guided you to install terraform, launching ec2 instance, and creating s3 bucket using it.  We also showed you to destroy your infrastructure created from terraform.

To install the Atom snap, open your terminal (Ctrl+Alt+T) and run the following command:

sudo snap install atom --classic

That’s it. Atom has been installed on your Ubuntu desktop, and you can start using it.

Whenever a new version is released, the Atom snap package will be automatically updated in the background.

If you are not comfortable with the command line, open Ubuntu Software, search for “Atom” and install the application.

## Installing Atom with apt

Atom is available from the Packagecloud Apt repositories. To install it, follow the steps below:

1. Update the packages list and install the dependencies:

sudo apt updatesudo apt install software-properties-common apt-transport-https wget

1. Import the repository GPG key and enable the [Apt repository](https://linuxize.com/post/how-to-add-apt-repository-in-ubuntu/) :

wget -q https://packagecloud.io/AtomEditor/atom/gpgkey -O- | sudo apt-key add -sudo add-apt-repository "deb [arch=amd64] https://packagecloud.io/AtomEditor/atom/any/ any main"

1. Once the repository is enabled, [install](https://linuxize.com/post/how-to-use-apt-command/) the latest version of Atom:

sudo apt install atom

When a new version is released, you can update Atom through the command-line or your desktop Software Update tool.

## Starting Atom

In the Activities search bar type “Atom” and click on the icon to launch the application.

When you start the Atom editor for the first time, a window like the following should appear:

From here, you start installing themes and extensions and configuring the editor according to your preferences.

## Conclusion

We’ve shown you how to install Atom on Ubuntu 20.04. To learn more about how to use Atom, from beginner basics to advanced techniques, visit their official [documentation](https://atom.io/docs) page.

If you have any questions, please leave a comment below.

[atom](https://linuxize.com/tags/atom/)