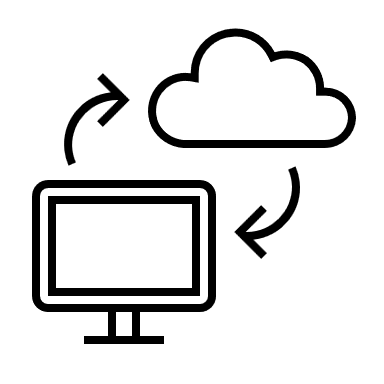
Installation guide for cloud recommendations



Ashok Das & Tinku Manivikesh Chukkapalli

# Introduction

The adaptation of cloud-based solutions has grown rapidly in recent times. Recent studies say that nearly 70% of organizations are migrating more workloads to the cloud and 50% of the organizations are developing their solutions with a cloud-first strategy. With this rapid growth trend in cloud migration, the organizations bare more OpEx costs and around 65% of the organizations which are working with cloud are looking for solutions or strategies to cut down their costs without impacting productivity.

Cloud Recommendations is a customizable solution to auto-schedule the recommendations provided by the cloud provider and apply the recommendations through a single click. The main problems that this cloud recommendation addresses to reduce are:

1. The need for constant checks for recommendations in the cloud platform.
2. Single point of approval and applying recommendations.

This solution can be used with non-production environments in the cloud.

## Purpose:

The main purpose of this document is to guide the installation and setup of the cloud Recommendations which can be used to achieve cost optimization on various cloud providers like GCP and AWS.

## Scope:

The Scope of the guide is:

* To cover the installation and setup of Google Cloud SDK and AWS CLI to actively manage scripts and pipelines.
* To cover the basic troubleshooting steps while setting up and running these pipelines through Jenkins.

The out-of-the-scope elements are:

* Installation and setup of pipelines through any other automation tool except Jenkins. However, some knowledge can be extended to other tools.
* Every other error while setting up and operating all these tools, scripts, templates, and pipelines. Basic troubleshooting steps are provided. Should you get any errors please refer to the relevant documents available online.

## Audience:

This guide is intended for end-users, cloud administrators, operations teams, and technical personnel who is willing to reduce and optimize the GCP and AWS compute costs without affecting the operations during the active period and spending extended time managing them.

# Environment Setup

This section describes the software required for the environment setup and the procedure to install them.

## Google Cloud SDK:

**Google Cloud SDK** (Software Development Kit) provides a set of tools that are used to manage resources hosted on the Google Cloud Platform (GCP). The SDK provides gcloud, gsutil, and bq commands with the ability to access the Google Cloud via the terminal.

If you are using instances in Google Cloud Platform Google Cloud SDK will be Preinstall in the instance.

Otherwise here are the steps to install Google Cloud SDK:

### Prerequisites to install Google Cloud SDK:

Login to your Ubuntu system with a sudo privileged account.

Open a terminal and execute the following commands to install the required packages on your system.

**$ sudo apt update**

**$ sudo apt install apt-transport-https ca-certificates gnupg**

### Installation of Google Cloud SDK using Apt Repository

Follow the below steps:

## Step 1: Import the GPG key

First, import the apt GPG key to your system with the below command.

**$ curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add**

#### Step 2: Create a PPA file

Then create a PPA file in your system by referring to the cloud-sdk in the Google packages repository

**$ echo "deb https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list**

#### Step 3: Install google-cloud-SDK

Finally, update the apt cache and install Google Cloud Packages on your Ubuntu system

**$ sudo apt update**

**$ sudo apt install google-cloud-sdk**

Press ‘y’ and hit enter for any confirmation asked by the installer.

### Initialize Cloud SDK

After you install Cloud SDK successfully on your system. The next step is to perform initializing the environment with **gcloud init** command. This authorizes Cloud SDK tools to use your Google account credentials to access Google Cloud and manage it.

To initialize the **gcloud CLI**:

#### Step 1: Run gcloud init:

**$ sudo gcloud init**

#### Step 2: Create or select a configuration if prompted

If you are initializing a new **gcloud CLI** installation**, gcloud init** creates a configuration named **default** for you and sets it as the active configuration. If you have existing configurations, **gcloud init** prompts you to choose between three options — re-initialize the active one, switch to another one, and re-initialize it, or create a new one.

#### Step 3: Complete the authorization step when prompted

Depending on whether you have previously authorized access to Google Cloud, you might be prompted to log in and grant access in a web browser or to select an existing account.

**$ sudo gcloud init –console-only**

If you used the **--console-only** flag and login is required, you must then copy and paste the provided authorization URL into another browser window and follow the prompts provided.

When this step is completed**, gcloud init** sets the **account**property in the configuration to the specified account.

#### Step 4: Choose a current Google Cloud project if prompted

If you only have access to one project, including the default project for your user account**, gcloud init** selects it for you.

Otherwise, you can select a project from a list of projects for which you have **Owner**, **Editor** or **Viewer** permissions. **gcloud init** sets the **project**property in the configuration to the property you choose. If you have access to more than 200 projects, you will be prompted to enter a project id, create a new project, or list projects.

**This account has a lot of projects! Listing them all can take a while.**

**[1] Enter a project ID**

**[2] Create a new project**

**[3] List projects**

**Please enter your numeric choice:**

#### Step 5: Choose a default Compute Engine zone if prompted

If you don't have the Compute Engine API enabled or have a default zone in your project-level metadata, you will not see this step**. gcloud init** sets the region and zone properties in the configuration using the zone you choose.

When **gcloud init** finishes, it prints the properties in the active configuration to the terminal:

**[compute]**

**region = us-east1**

**zone = us-east1-b**

**[core]**

**account = user@google.com**

**disable\_usage\_reporting = False**

**project = example-project**

You can view these properties at any other time using the **[gcloud config list](https://cloud.google.com/sdk/gcloud/reference/config/list)**command.

# 2.2. AWS-CLI:

The AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell. With minimal configuration, the AWS CLI enables you to start running commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from the command prompt in your terminal program.

The installation of AWS-CLI can be done by executing the following commands:

### Step 1: Curl Command

Use the **curl** command – The **-o** option specifies the file name that the downloaded package is written to. The options on the following example command write the downloaded file to the current directory with the local name awscliv2.zip.

**$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**

### Step 2: Unzip the Installer

Unzip the installer. If your Linux distribution doesn't have a built-in unzip command, use an equivalent to unzip it. The following example command unzips the package and creates a directory named **aws**under the current directory.

**$ unzip awscliv2.zip**

### Step 3: Install Program

Run the install program. The installation command uses a file named install in the newly unzipped **aws** directory. By default, the files are all installed to **/usr/local/aws-cli,** and a symbolic link is created in **/usr/local/bin**. The command includes **sudo** to grant write permissions to those directories.

**$ sudo ./aws/install**

### Step 4: Move the file

Move the downloaded file to the path, The default value is **/usr/local/bin**.

**$ ./aws/install -i /usr/local/aws-cli -b /usr/local/bin**

### Step 5: Check the version

Now, Check the version of the AWS-CLI by running the following command. And the output is given below.

**$ aws –version**

**aws-cli/2.4.5 Python/3.8.8 Linux/4.14.133-113.105.amzn2.x86\_64 botocore/2.4.5**

### Step 6: Authenticate AWS Account through AWS-CLI

To create and manage any AWS resource through AWS-CLI, you must authenticate first and this AWS account must be having related permissions for the services.

Run the below command and provide the necessary details to authenticate.

**$ aws configure**

**AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE**

**AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY**

**Default region name [None]: us-west-2**

**Default output format [None]: json**

For more information on authenticating via AWS-CLI, please follow the official documentation from AWS (link is below):

<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html>

## Jenkins Pipeline Setup:

Please refer to “To appendix A: Installation of Jenkins in Ubuntu” for the steps to install and configure Jenkins in Ubuntu. For other distributions or OS, please refer to official documentation.

### Creating Pipeline Jobs in Jenkins:

**Step 1:** In Jenkins, click on**‘New Item’** to create a new project.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 1: Jenkins - New Job

**Step 2: Give your project a name, then choose ‘Pipeline’ and finally, click on ‘OK’.**

Graphical user interface, text

Description automatically generated with medium confidence

Figure 2: Jenkins - Job Description

**Step3:** In the Definition choose Pipeline script from SCM from the Drop Down and Git in the SCM from the drop-down.

**Step4:** Input the GitHub private SSH repo URL in the Repository URL. In the credentials, field chooses the GitHub credentials from the drop-down.

Graphical user interface, text, application, email

Description automatically generated

Figure 3: Jenkins - Pipeline

**Step 5**: Specify the branch name of GitHub and the pipeline script path.

Graphical user interface, text, application, email, Teams

Description automatically generated

Figure 4: Jenkins - Git repo

# Cloud Recommendations:

For the cloud recommendations, you must store the code in the GitHub repository. And create two pipeline jobs in Jenkins.

## Job 1 Getting recommendations:

Please follow the steps to create the Jen Jenkins pipeline explained earlier. Kindly place the Jenkins file for the job (recommendation. Jenkins) in the GitHub private repo. Please configure your GitHub credential in Jenkins to authenticate.

For getting the recommendations and creating the pull request kindly place the shell script file in the GitHub repo. (recommendation.sh)

## Job 2 Applying recommendations:

To apply the recommendation we have to create the Jenkins pipeline with (recommendations\_apply.jenkins) file. Place the file in the GitHub private repo.

To apply the recommendations kindly place the shell script file in the GitHub repo. (recommendation\_apply.sh)

# Troubleshooting Guide

## Issues with AWS-CLI setup

While setting up AWS-CLI make sure you are installing the latest version. And you may also phase some common errors like not moving the downloaded binary file to the correct location. And check the version after moving the file to a particular location.

Correct location: **$ ./aws/install -i /usr/local/aws-cli -b /usr/local/bin**

## Issues with AWS- Authentication

While running the **$ aws configure** command need to double-check whether we are pasting the right credentials like access key, secret access key, and region code.

## Issues with Google Cloud-CLI Authentication

Once Google Cloud SDK in installed make sure you initialize the Gcloud CLI by using the **init** command. If your system does not support browser, then use **init** **--console-only**.

Make sure you login with the correct username to view the list of projects you have access.

When prompted for the project id and default zone make sure you have selected the correct project id and default zone same as the Jenkins worker node project and zone.

# Appendix A: Installation of Jenkins in Ubuntu

## A.1. Prerequisites for Jenkins installation:

Minimum hardware requirements:

* 256 MB of RAM
* 1 GB of drive space

Recommended hardware configuration for a small team:

* 4 GB+ of RAM
* 50 GB+ of drive space

## A.2. Installation of Java:

Java is required to run Jenkins.

To install Java, run following in terminal:

**$ sudo apt update**

**$ sudo apt install openjdk-11-jre**

## A.3. Jenkins Installation:

The version of Jenkins included with the default Ubuntu packages is often behind the latest available version from the project itself. To ensure you have the latest fixes and features, use the project-maintained packages to install Jenkins.

First, add the repository key to the system:

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

After the key is added the system will return with OK.

Next, let’s append the Debian package repository address to the server’s ‘**sources.list’**:

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

After both commands have been entered, you’ll run update so that apt will use the new repository.

**$ sudo apt update**

Finally, you’ll install Jenkins and its dependencies.

**$ sudo apt install jenkins**

Now that Jenkins and its dependencies are in place, you’ll start the Jenkins server.

## A.4. Starting Jenkins:

Let’s start Jenkins by using [systemctl](https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units):

**$ sudo systemctl start Jenkins**

Since **systemctl** doesn’t display status output, you’ll use the status command to verify that Jenkins started successfully:

**$ sudo systemctl status Jenkins**

Now that Jenkins is up and running, let’s adjust your firewall rules so that you can reach it from a web browser to complete the initial setup.

## A.5. Setting Up Jenkins

To set up your installation, visit Jenkins on its default port, 8080, using your server domain name or IP address: http://your\_server\_ip\_or\_domain:8080

You should receive the Unlock Jenkins screen, which displays the location of the initial password:

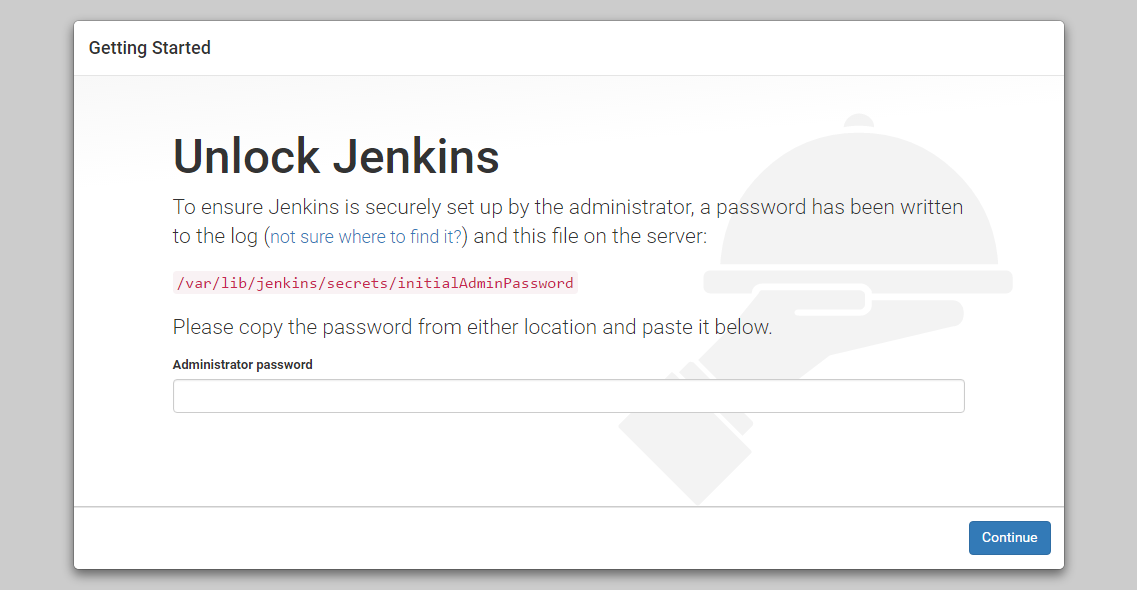


Figure 4: Jenkins - Unlock Jenkins

In the terminal window, use the **cat** command to display the password:

**$ sudo cat /var/lib/Jenkins/secrets/initialAdminPassword**

Copy the 32-character alphanumeric password from the terminal and paste it into the Administrator password field, then click Continue.

The next screen presents the option of installing suggested plugins or selecting specific plugins:

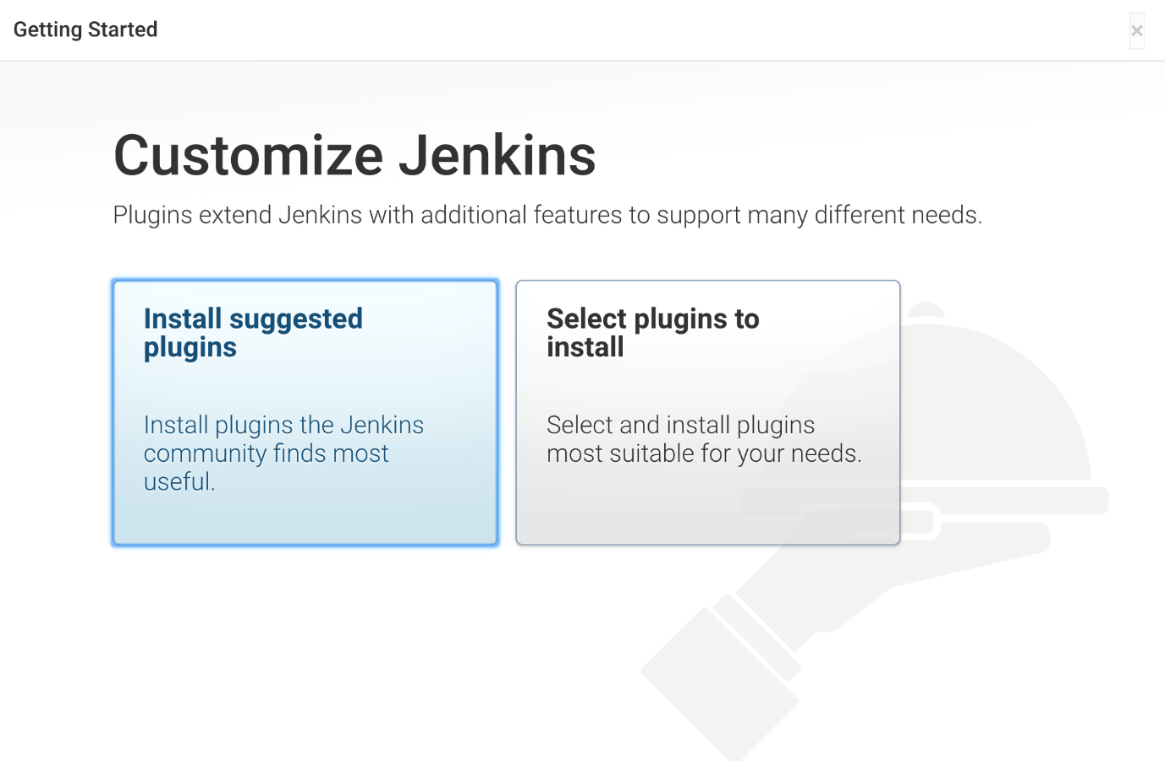


Figure 5: Jenkins - Select Plugin to Install

Click the Install suggested plugins option, to begin the installation process.

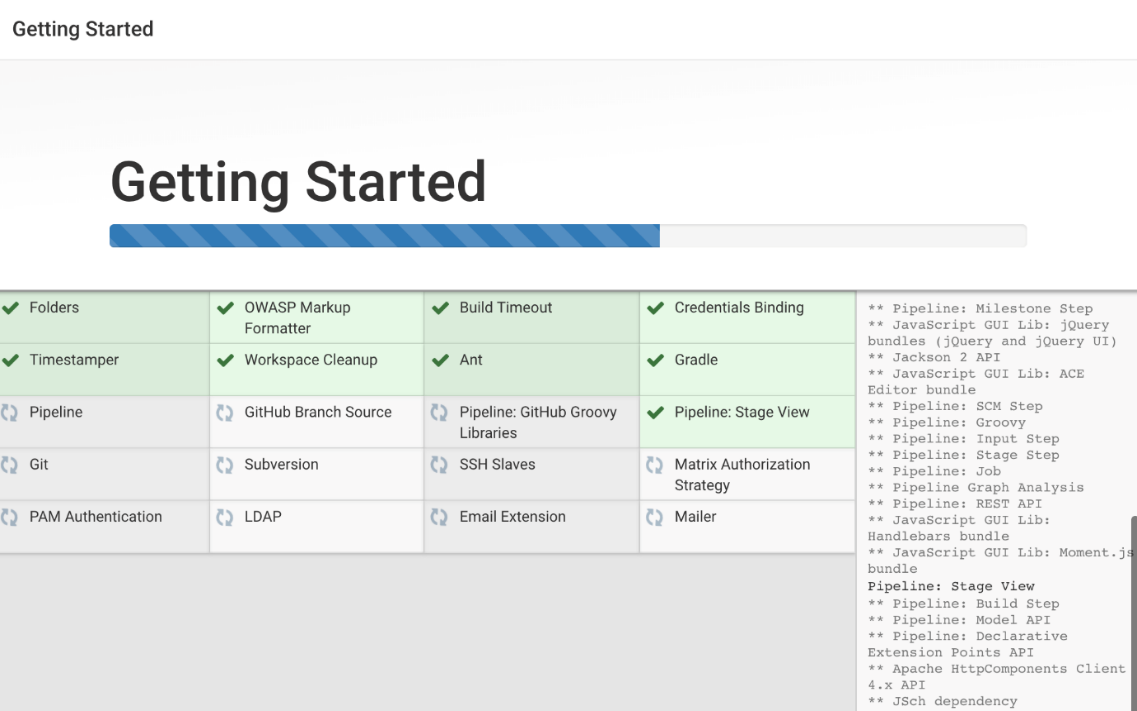


Figure 6: Jenkins - Plugin Installation

When the installation is complete, you’ll be prompted to set up the first administrative user. It’s possible to skip this step and continue as admin using the initial password you used above, but you’ll take a moment to create the user.

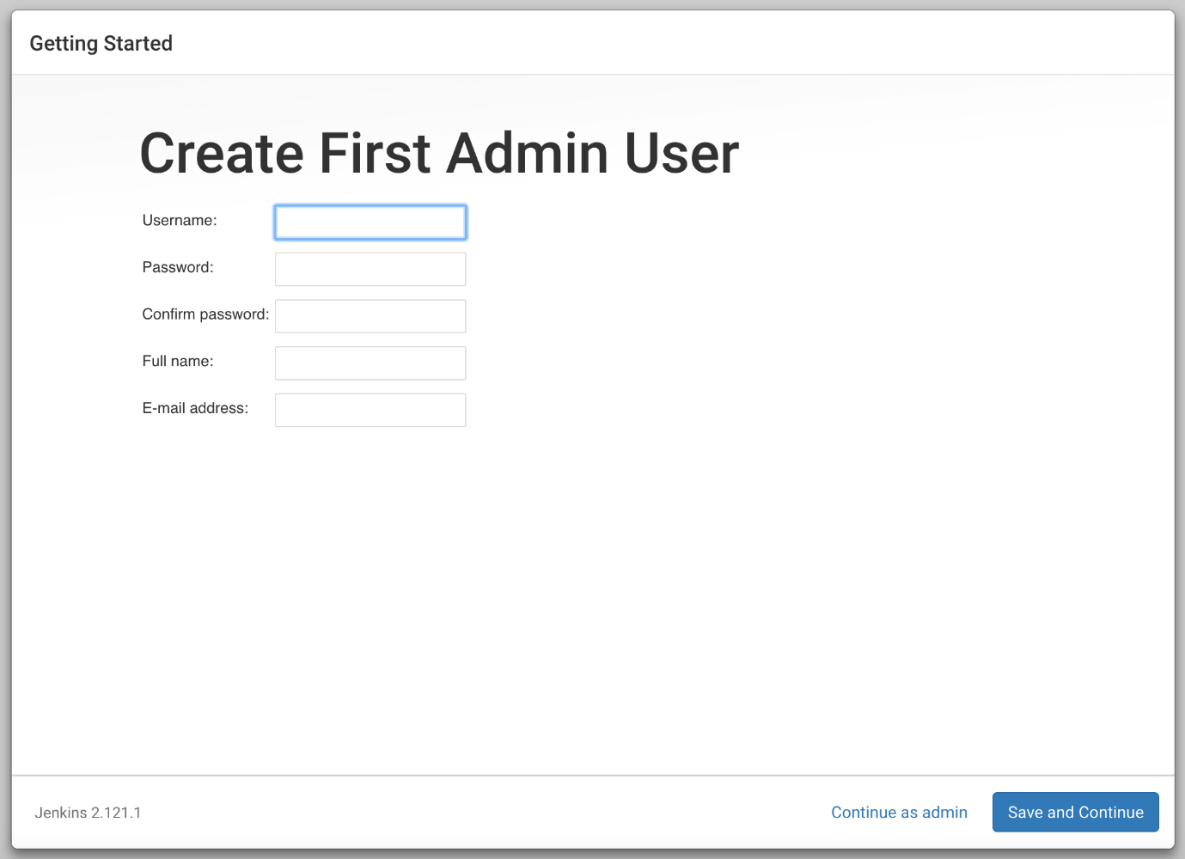


Figure 7: Jenkins - Create Admin User

Enter the name and password for your user:

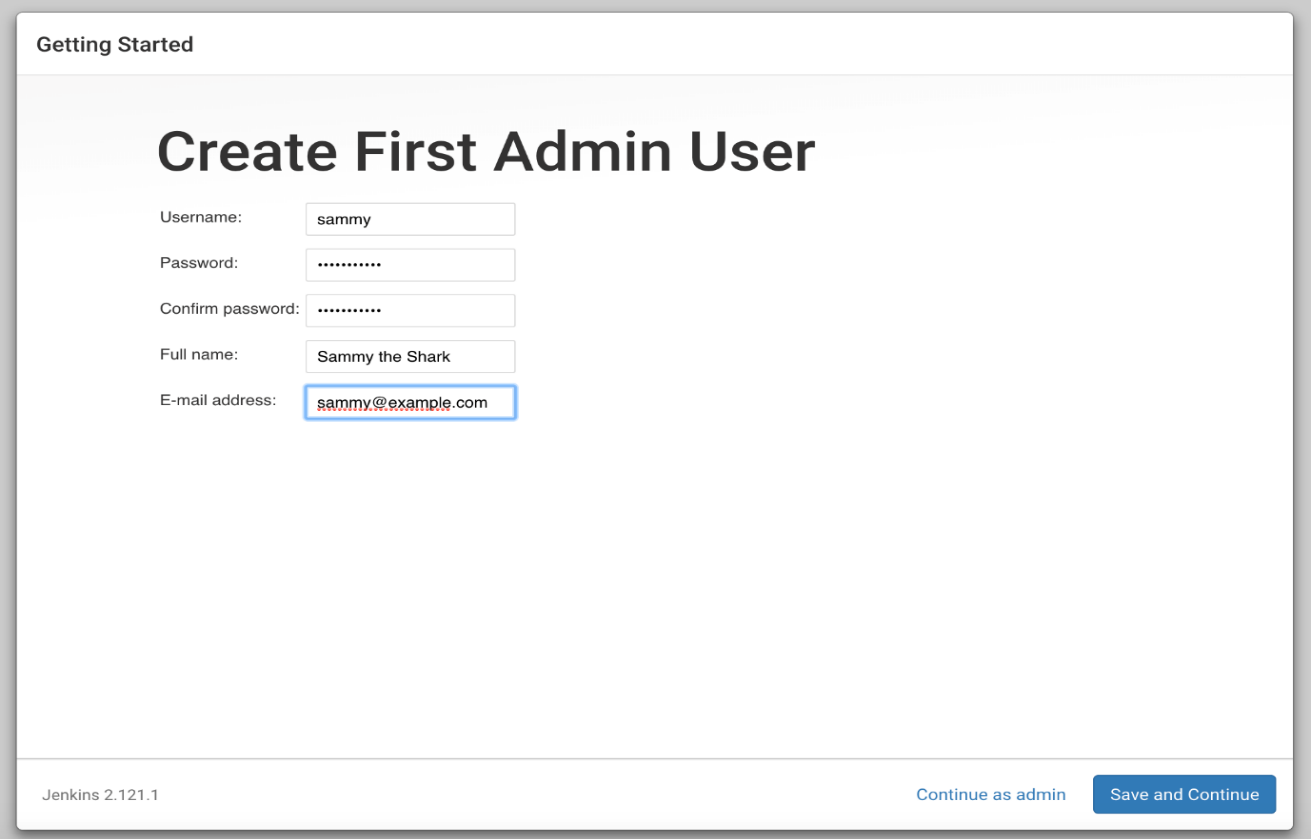


Figure 8: Jenkins - Admin User Details

You’ll receive an Instance Configuration page that will ask you to confirm the preferred URL for your Jenkins instance. Confirm either the domain name for your server or your server’s IP address:

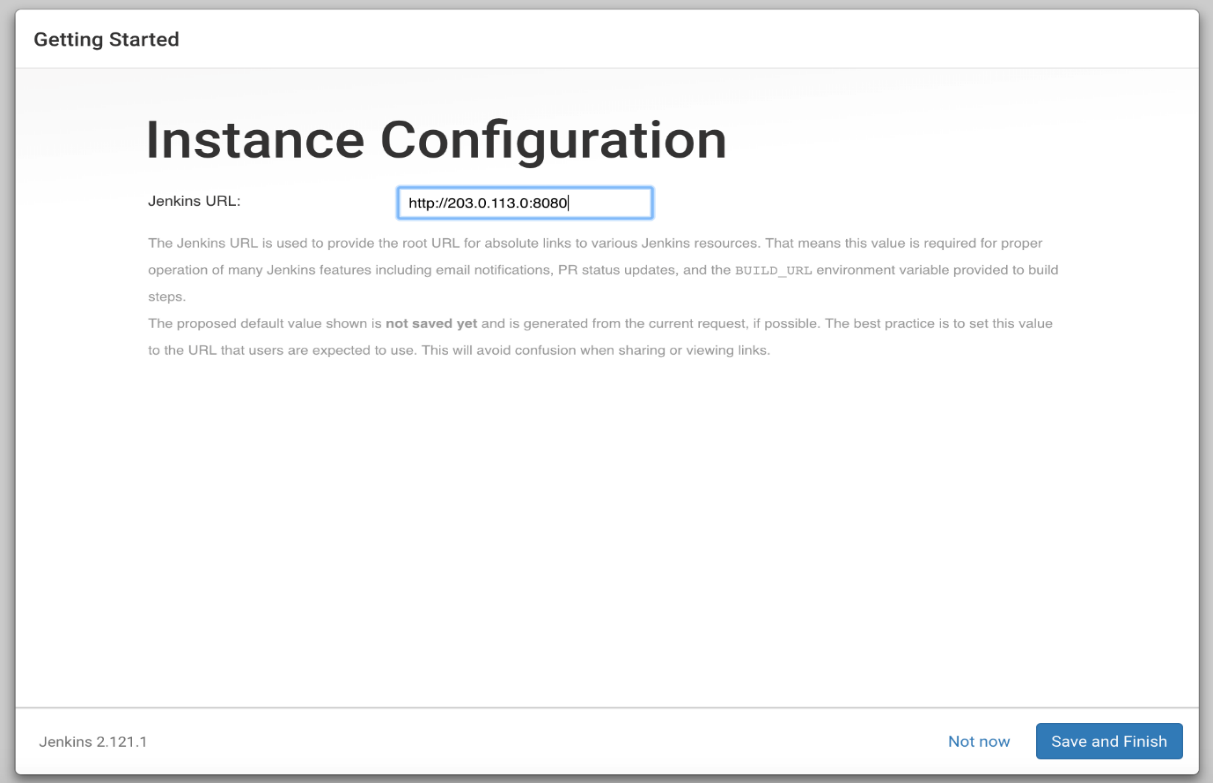


Figure 9: Jenkins - Jenkins URL

After confirming the appropriate information, click **Save and Finish**. You’ll receive a confirmation page confirming that **“Jenkins is Ready!”:**

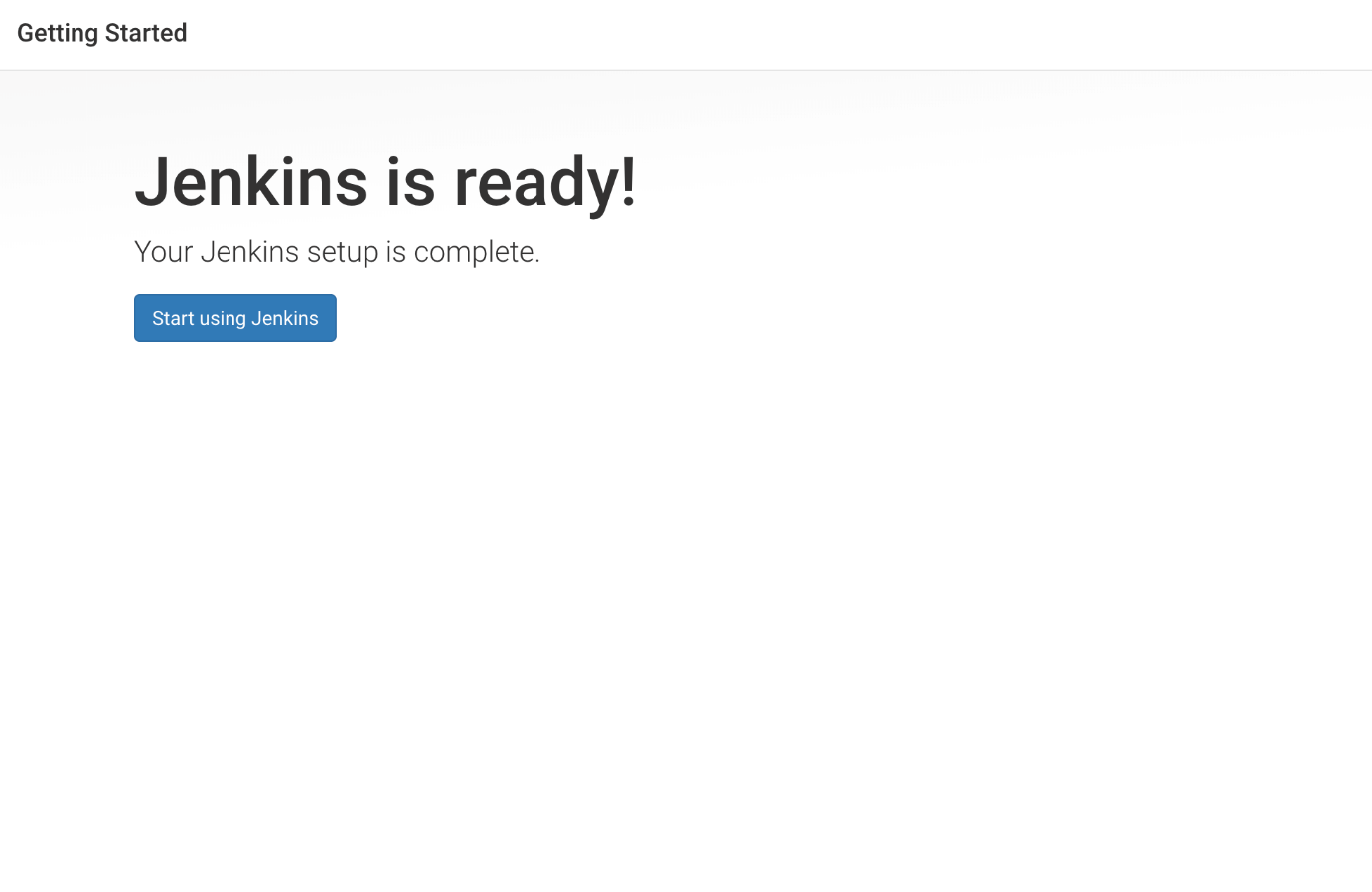


Figure 10: Jenkins - Installation Finish

Click **Start** using **Jenkins** to visit the main Jenkins dashboard:

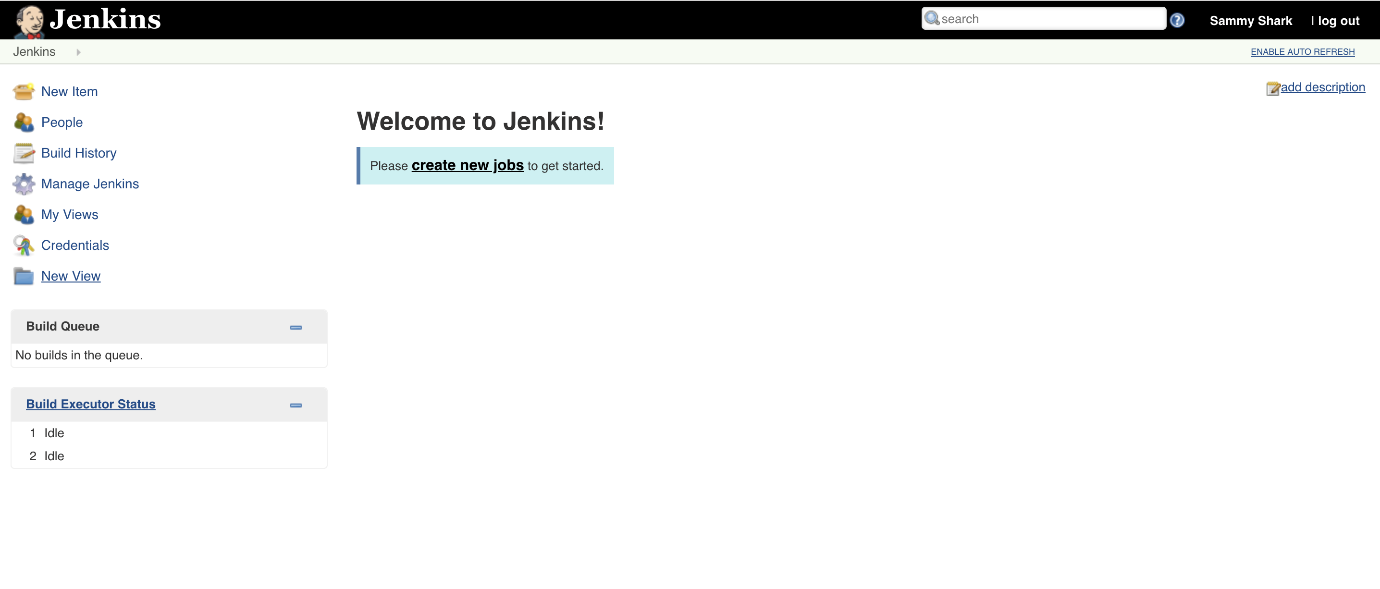


Figure 11: Jenkins - Welcome Page

At this point, you have completed a successful installation of Jenkins.

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| GCP | Google Cloud Platform |
| AWS | Amazon Web Services |
| AWS-CLI | AWS-Command Line Interface |
| SBX | Sand Box Environment |
| VM | Virtual Machine |
| SDK | Software Development Kit |

# Glossary

# Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Version** | **Section Changed** | **Description** | **Author** |
| 30 Sep. 2022 | 1.0 | All | Initial Draft | Ashok Das, Tinku Manivikesh Chukkapalli |
|  |  |  |  |  |