

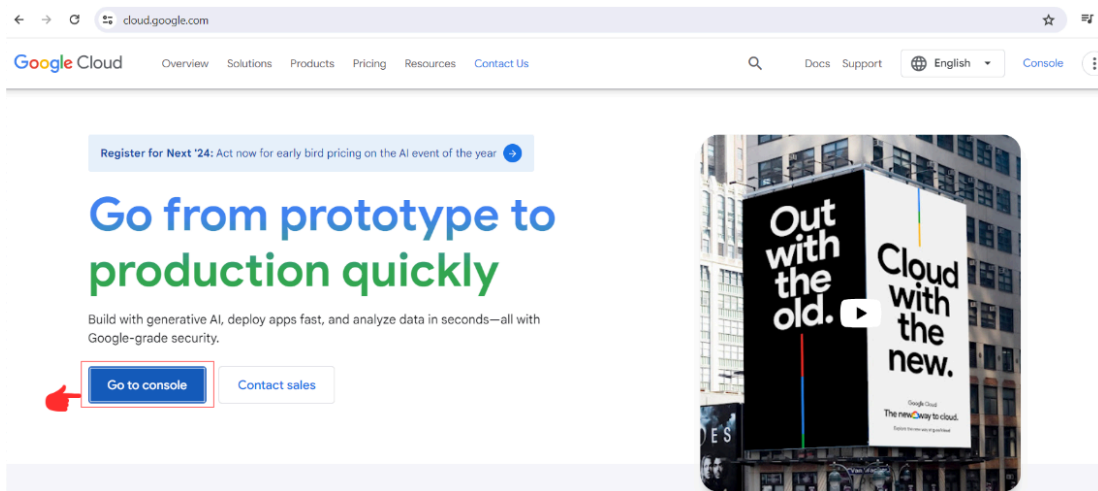
Deploying and Configuring Jenkins Server

(Usually these activities are performed by DevOps Engineers)

Go to cloud.google.com

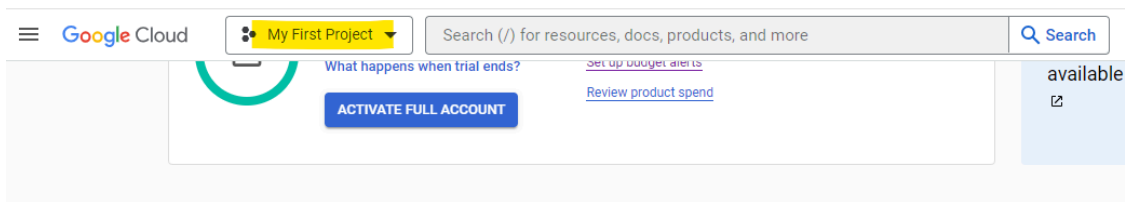
Create a GCP account (provides 300\$ free credits that would be valid for 3 months)

Once you create an account -> Go to Console =>
<https://console.cloud.google.com>




Create a new Project and Deploy Resources (Like - Jenkins)


To create the new project name as shown in the screenshot:





Recommended based on your interest in General

Pre-built solution templates

**Deploy a three-tier web app**
Web app, rich media site, ecommerce website, database-backed website

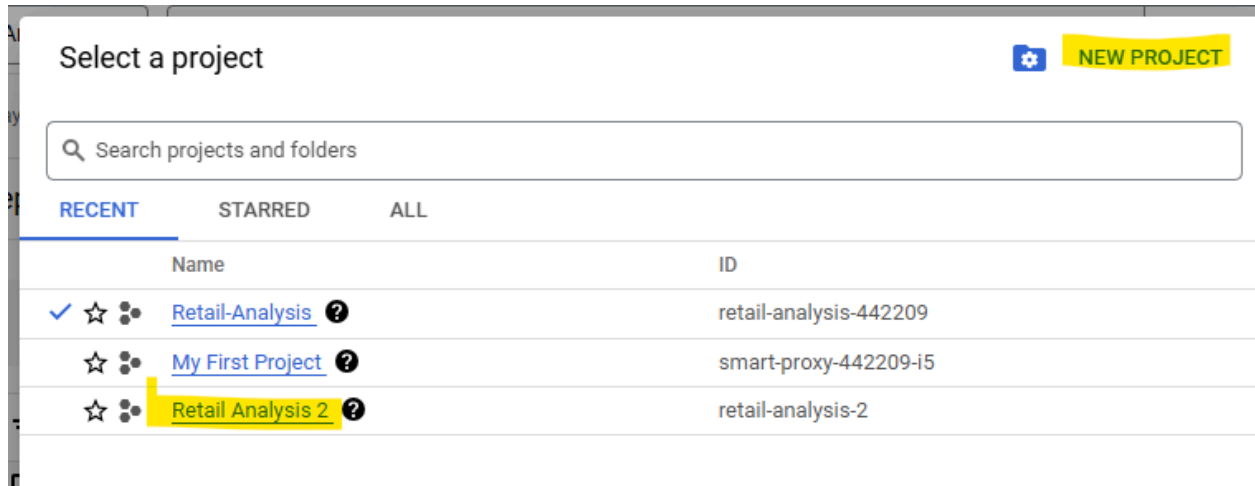
**Deploy load balanced managed VMs**
Data analysis, data pipeline, application logs, log management, log intelligence

**Create a data warehouse with BigQuery**
Data warehouse, dashboard, data analysis

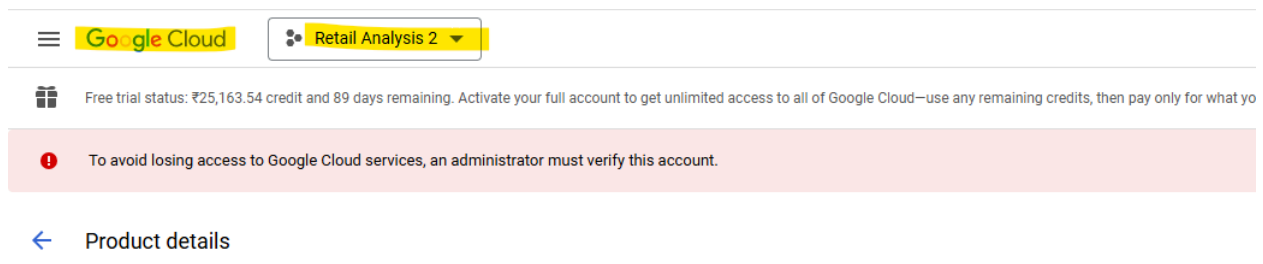
 [View all Solutions](#)

Now, the screen below will be visible. To initiate the creation of a new project, select the **"NEW PROJECT"** option highlighted in the screenshot below.

Now create the new project ex: Retail Analysis 2



And now select this project as shown below.



Cloud Deployment Manager V2 API

[Google Enterprise API](#)

The Google Cloud Deployment Manager v2 API provides services for configuring, deploying,...

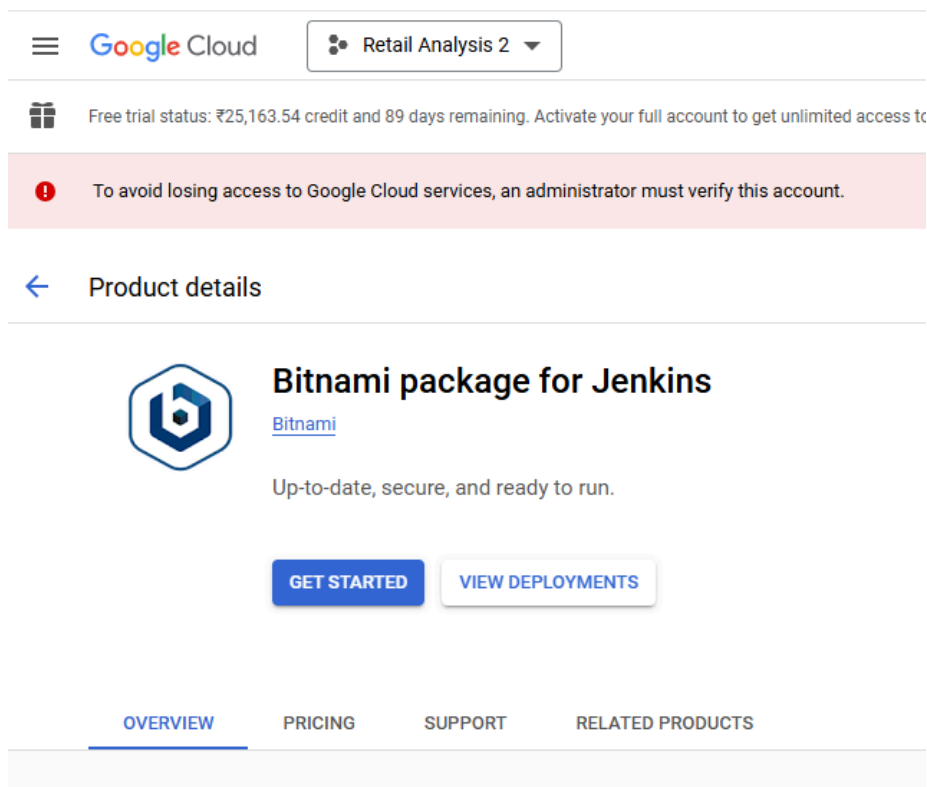
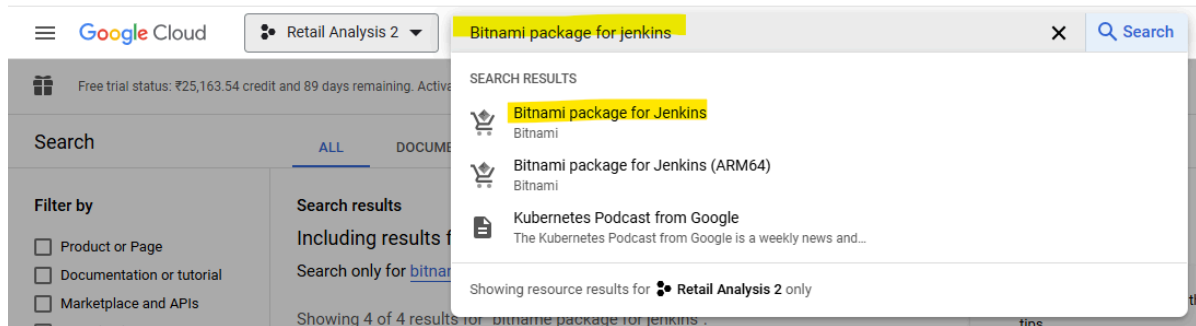
ENABLE

TRY THIS API [↗](#)

Deploy the necessary resources - Like Jenkins:

To deploy the jenkins follow below steps:

Search for “jenkin” in and in the marketplace you will get the option of “**Bitnami package for Jenkins**” refer attached screenshot.




Allow the installation that it will ask for. Then you will get the option to launch the package as shown in the screenshot below.

To avoid losing access to Google Cloud services, an administrator must verify this account.

←

Product details



Bitnami package for Jenkins

[Bitnami](#)

Up-to-date, secure, and ready to run.

LAUNCH

VIEW DEPLOYMENTS

OVERVIEW

PRICING

SUPPORT

RELATED PRODUCTS

Overview

Additional details

Jenkins is an open source automation server that helps you automate the building, testing, and deployment of your project across multiple platforms.

For package deployment, specify the zone and machine type as highlighted in the screenshot. Leave other configurations unchanged, and proceed to click on "Deploy."

Also click on "Enable" as shown below

Enable required APIs

1

The following APIs are required to deploy a VM product from Marketplace

[Compute Engine API](#)

Not enabled

[Cloud Deployment Manager V2 API](#)

Not enabled

[Cloud Runtime Configuration API](#)

Not enabled

ENABLE

SEND FEEDBACK

Google Cloud

CICD Demo

New Bitnami package for Jenkins deployment

Deployment name *

jenkins-1

Zone

asia-south1-a

Machine type

General purpose

Compute optimized

Memory optimized

Machine types for common workloads, optimized for cost and flexibility


Series

N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type

n1-standard-1 (1 vCPU, 3.75 GB memory)



vCPU

1

Memory

3.75 GB

Additional information

Bitnami package for Jenkins overview

Product provided by Bitnami

Bitnami Jenkins Usage Fee

Bitnami does not charge a usage fee.

INR 0.00/mo

Infrastructure fee

VM Instance: 1 vCPU + 3.75 GB memory (n1-standard-1)

INR 3,471.24/mo

Standard Persistent Disk: 10GB

INR 40.01/mo

Sustained use discount

- INR 1,041.37/mo

Estimated monthly total

INR 2,469.87/mo

All products are priced in USD and charged in the currency (INR) specified by your Billing Account. The price for this month is calculated with an exchange rate of 1 USD = 83.31 INR

Note: If you face this issue please try to deploy it again with a different zone.


The screenshot shows the Google Cloud Platform console with two panels. The left panel, titled 'jenkins-1', shows a deployment failure with a red error icon and the message 'jenkins-1 failed to deploy'. Below this is a tree view of the deployment components, including 'bitnami-package-for-jinja', 'bitnami-package-for-vm-tmpl', 'jenkins-1-vm', 'generated-password-0', 'software-status', 'jenkins-1-config', 'jenkins-1-software', 'software-status-script', 'jenkins-1-tcp-80', and 'jenkins-1-tcp-443'. The right panel, titled 'bitnami-package-for', shows a detailed error message: 'bitnami-package-for has resource level errors'. The error details for 'jenkins-1-vm' indicate a 'ZONE_RESOURCE_POOL_EXHAUSTED' error, stating that the zone 'projects/grand-guru-410310/zones/asia-south1-a' does not have enough resources available. Below the error message is a table of instance details: Site address (red error icon), Admin user (user), Admin password (Temporary) (X9nLGeWskJKv), Instance (jenkins-1-vm), Instance zone (asia-south1-a), and Instance machine type (n1-standard-1). A 'MORE ABOUT THE SOFTWARE' link is also present.

Once it is deployed you will get below screen. Now log into SSH.

The screenshot shows the Google Cloud Platform console with two panels. The left panel, titled 'jenkins-1', shows a deployment status with a yellow warning icon and the message 'jenkins-1 has been deployed, but contains warnings'. Below this is a tree view of the deployment components, including 'bitnami-package-for-jinja', 'bitnami-package-for-vm-tmpl', 'jenkins-1-vm', 'generated-password-0', 'software-status', 'jenkins-1-config', 'jenkins-1-software', 'software-status-script', 'jenkins-1-tcp-80', and 'jenkins-1-tcp-443'. The right panel, titled 'bitnami-package-for', shows the instance details: Instance (jenkins-1-vm), Instance zone (asia-east1-a), and Instance machine type (n1-standard-1). Below the instance details is a 'MORE ABOUT THE SOFTWARE' link. The panel also includes a 'Get started with Bitnami package for Jenkins' section with a 'VISIT THE SITE' button and an 'SSH' button. A 'Suggested next steps' section lists two steps: 'Change the temporary password' and 'Assign a static external IP address to your VM instance'. A 'Documentation' section includes a 'Getting Started' link.

Click on Site Address and username and password log into the site. Refer attached screenshot.

Close

 **Bitnami package for Jenkins**
Solution provided by Bitnami

Site address	http://35.201.205.191/
Admin user	user
Admin password (Temporary)	JEzgjzZPHKm9
Instance	jenkins-1-vm
Instance zone	asia-east1-a
Instance machine type	n1-standard-1

[MORE ABOUT THE SOFTWARE](#)

Get started with Bitnami package for Jenkins

[VISIT THE SITE](#) [SSH](#)

Suggested next steps

- Change the temporary password
For additional security, it is recommended that you change the password.

Note: In order to ssh into the cluster first click on SSH and it will ask you to "Authorize" and then you can access the cluster. Refer below command for more clarity.

Google Cloud

Retail Analysis 2

Search (/) for resources, docs, products, and more

Search

Free trial status: ₹25,163.54 credit and 89 days remaining. Activate your full account to get unlimited access to all of Google Cloud—use any remaining credits, then pay only for what you use.

Deployment Manager

jenkins-1

DELETE

Deployments

Type registry

jenkins-1 has been deployed, but contains warnings

VIEW DETAILS

Overview - jenkins-1

bitnami-package-for bitnami-package-for.jinja

bitnami-package-for-vm-tmpl vm_instance.py

jenkins-1-vm vm instance

generated-password-0 password.py

software-status software_status.py


jenkins-1-config config

jenkins-1-software config waiter

software-status-script software_status_script.py

jenkins-1-tcp-80 firewall

bitnami-package-for

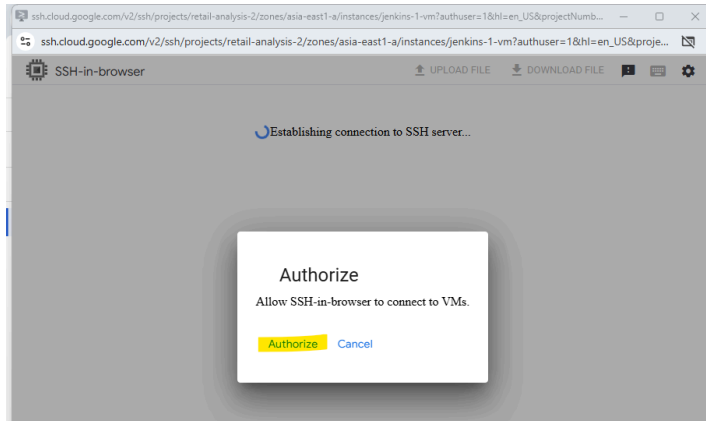
 **Bitnami package for Jenkins**
Solution provided by Bitnami

Site address	http://35.189.160.162/
Admin user	user
Admin password (Temporary)	trMBm7+XF39h
Instance	jenkins-1-vm
Instance zone	asia-east1-a
Instance machine type	n1-standard-1

[MORE ABOUT THE SOFTWARE](#)

Get started with Bitnami package for Jenkins

[VISIT THE SITE](#) [SSH](#)



Follow below steps before proceeding ahead.

=> First check the version of Python that is present in the jenkins cluster. Note down this version of Python as later on this version we have to install using “pyenv” in the local system (i.e Windows) and in the virtual environment also in VS Code.

Use command `python3 --version` (Note normal `python --version` command will give error)

```
ssh.cloud.google.com/v2/ssh/projects/retail-analysis-2/zones/asia-east1-a/instances/jenkins-1-vm?authuser=1&hl=en_US&projectNumb...
ssh.cloud.google.com/v2/ssh/projects/retail-analysis-2/zones/asia-east1-a/instances/jenkins-1-vm?authuser=1&hl=en_US&proje...
SSH-in-browser
Linux jenkins-1-vm 6.1.0-26-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

[Bitnami Jenkins logo]

-> Welcome to Bitnami package for Jenkins 2.479.1
-> Documentation: https://docs.bitnami.com/google/apps/jenkins/
-> Bitnami Support: https://github.com/bitnami/vms/issues
jenkins-1-vm:~$ python --version
-bash: python: command not found
jenkins-1-vm:~$ python3 --version
Python 3.11.2
```

Here the Python version is 3.11.2.

=> Using the commands “`sudo apt-get install python3-pip`”, “`sudo apt-get install sshpass`” install pip and sshpass respectively in SSH.

sudo apt-get install python3-pip

```
ssh.cloud.google.com/v2/ssh/projects/retail-analysis-442209/zones/asia-east1-a/instances/jenkins-1-vm?authuser=1&hl=en_US...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

dhruthigowda12@jenkins-1-vm:~$ pip --version
-bash: pip: command not found
dhruthigowda12@jenkins-1-vm:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  build-essential bzip2 cpp cpp-12 dpkg-dev fakeroot g++ g++-12 gcc gcc-12 javascript-common libabsl20220623
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libaom3 libasan8 libatomic1
  libavif15 libc-dev-bin libc-devtools libc6-dev libc6-0 libcrypt-dev libdav1d6 libde265-0 libdeflate0
  libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgavl-1 libgcc-12-dev libgd3 libgomp1
  libheif1 libisl23 libitm1 libjbig0 libjpeg62-turbo libjs-jquery libjs-sphinxdoc libjs-underscore liblerc4
  liblocale-gettext-perl liblsan0 libmpc3 libmpfr6 libnsl-dev libnuma1 libpython3-dev libpython3.11
  libpython3.11-dev libquadmath0 libraw1e0 libstdc++-12-dev libsvtav1enc1 libtiff6 libtirpc-dev libtsan2
  libubsan1 libwebp7 libx11-6 libx11-data libx265-199 libxau6 libxcb1 libxdmcp6 libxpm4 libyuv0 linux-libc-dev
  make manpages-dev patch python3-dev python3-distutils python3-lib2to3 python3-setuptools python3-wheel
  python3.11-dev rpcsvc-proto zlib1g-dev
Suggested packages:
  bzip2-doc cpp-doc gcc-12-locales cpp-12-doc debian-keyring g++-multilib g++-12-multilib gcc-12-doc
  gcc-multilib autoconf automake libtool flex bison gdb gcc-doc gcc-12-multilib apache2 | lighttpd | httpd
  glibc-doc bsr libgd-tools libstdc++-12-doc make-doc ed diffutils-doc python-setuptools-doc
The following NEW packages will be installed:
  build-essential bzip2 cpp cpp-12 dpkg-dev fakeroot g++ g++-12 gcc gcc-12 javascript-common libabsl20220623
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libaom3 libasan8 libatomic1
  libavif15 libc-dev-bin libc-devtools libc6-dev libc6-0 libcrypt-dev libdav1d6 libde265-0 libdeflate0
  libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgavl-1 libgcc-12-dev libgd3 libgomp1
  libheif1 libisl23 libitm1 libjbig0 libjpeg62-turbo libjs-jquery libjs-sphinxdoc libjs-underscore liblerc4
  liblocale-gettext-perl liblsan0 libmpc3 libmpfr6 libnsl-dev libnuma1 libpython3-dev libpython3.11
  libpython3.11-dev libquadmath0 libraw1e0 libstdc++-12-dev libsvtav1enc1 libtiff6 libtirpc-dev libtsan2
  libubsan1 libwebp7 libx11-6 libx11-data libx265-199 libxau6 libxcb1 libxdmcp6 libxpm4 libyuv0 linux-libc-dev
  make manpages-dev patch python3-dev python3-distutils python3-lib2to3 python3-pip python3-setuptools
  python3-wheel python3.11-dev rpcsvc-proto zlib1g-dev
0 upgraded, 83 newly installed, 0 to remove and 3 not upgraded.
Need to get 11.4 MB/82.9 MB of archives.
After this operation, 329 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [30 B]
Get:2 https://deb.debian.org/debian bookworm/main amd64 libc-dev-bin amd64 2.36-9+deb12u9 [46.7 kB]
Get:3 https://deb.debian.org/debian bookworm/main amd64 linux-libc-dev amd64 6.1.115-1 [2,066 kB]
```

sudo apt-get install sshpass

```
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

Preparing to unpack .../14-python3-pip_20.3.4-4+deb11u1_all.deb ...
Unpacking python3-pip (20.3.4-4+deb11u1) ...
Setting up javascript-common (11+nmmul) ...
Setting up python3-wheel (0.34.2-1) ...
Setting up libexpat1-dev:amd64 (2.2.10-2+deb11u5) ...
Setting up python-pip-whl (20.3.4-4+deb11u1) ...
Setting up libjs-jquery (3.5.1+dfsg+~3.5.5-7) ...
Setting up python3-lib2to3 (3.9.2-1) ...
Setting up libjs-underscore (1.9.1-dfsg-3) ...
Setting up python3-distutils (3.9.2-1) ...
Setting up python3-setuptools (52.0.0-4) ...
Setting up libpython3.9-dev:amd64 (3.9.2-1) ...
Setting up python3-pip (20.3.4-4+deb11u1) ...
Setting up libjs-sphinxdoc (3.4.3-2) ...
Setting up python3.9-dev (3.9.2-1) ...
Setting up libpython3-dev:amd64 (3.9.2-3) ...
Setting up python3-dev (3.9.2-3) ...
Processing triggers for man-db (2.9.4-2) ...
arathishatti15@jenkins-1-vm:~$ sudo apt-get install sshpass
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libevent-2.1-7 libgnutls-dane0 libunbound8
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  sshpass
```


=> Install zip manually in jenkins using below command:

```
sudo apt-get install -y zip
```

Verify Installation using the command “ zip --version”. This should output the installed version of zip. Refer below command for more clarity:

```
@jenkins-1-vm:~$ sudo apt-get install -y zip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  unzip
The following NEW packages will be installed:
  unzip zip
0 upgraded, 2 newly installed, 0 to remove and 3 not upgraded.
Need to get 396 kB of archives.
After this operation, 1,019 kB of additional disk space will be used.
Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [30 B]
Get:2 https://deb.debian.org/debian bookworm/main amd64 unzip amd64 6.0-28 [166 kB]
Get:3 https://deb.debian.org/debian bookworm/main amd64 zip amd64 3.0-13 [230 kB]
Fetched 396 kB in 0s (801 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package unzip.
(Reading database ... 89166 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-28_amd64.deb ...
Unpacking unzip (6.0-28) ...
Selecting previously unselected package zip.
Preparing to unpack .../archives/zip_3.0-13_amd64.deb ...
Unpacking zip (3.0-13) ...
Setting up unzip (6.0-28) ...
Setting up zip (3.0-13) ...
Processing triggers for man-db (2.11.2-2) ...
@jenkins-1-vm:~$ zip --version
Copyright (c) 1990-2008 Info-ZIP - Type 'zip -L' for software license.
This is Zip 3.0 (July 5th 2008), by Info-ZIP.
Currently maintained by E. Gordon. Please send bug reports to
the authors using the web page at www.info-zip.org; see README for details.
```

In UI, install the following plugins:

Dashboard view -

This plugin allows you to create a customized dashboard view, providing a summary of information from various jobs or builds. It helps in creating a visual representation of the overall build health and status.

Github branch Source -

The GitHub plugin in Jenkins enables integration with GitHub repositories. It allows Jenkins to trigger builds automatically when changes are pushed to a GitHub repository, and it provides the ability to specify branches for building.

pipeline declarative -

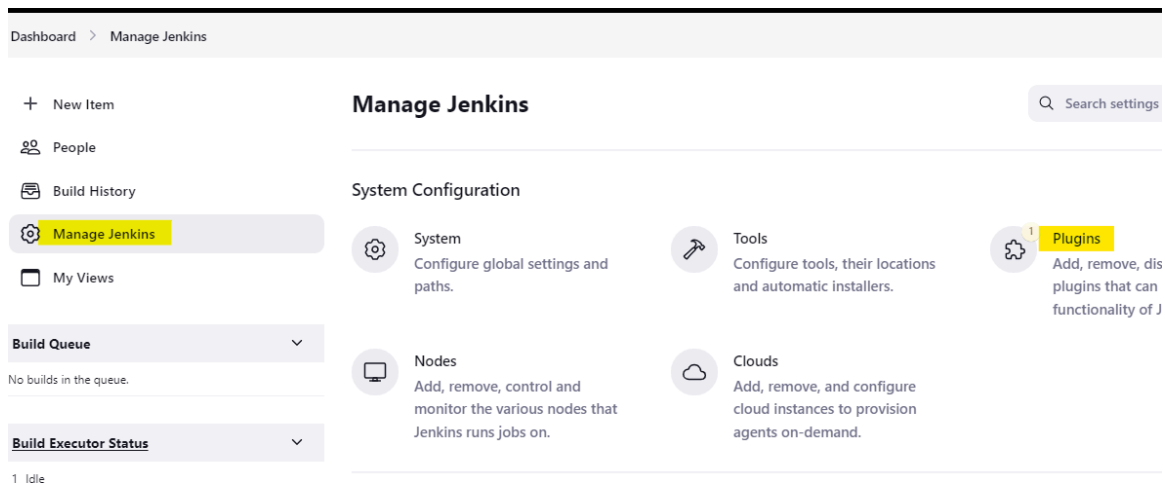
Declarative Pipeline is a feature within the Pipeline plugin that provides a simplified and opinionated syntax for defining build pipelines. It allows you to express pipelines in a more structured and readable format.

pipeline stage view -

The Stage View plugin enhances the visualization of Jenkins Pipelines. It provides a graphical representation of the stages in your pipeline, showing the progress and status of each stage.

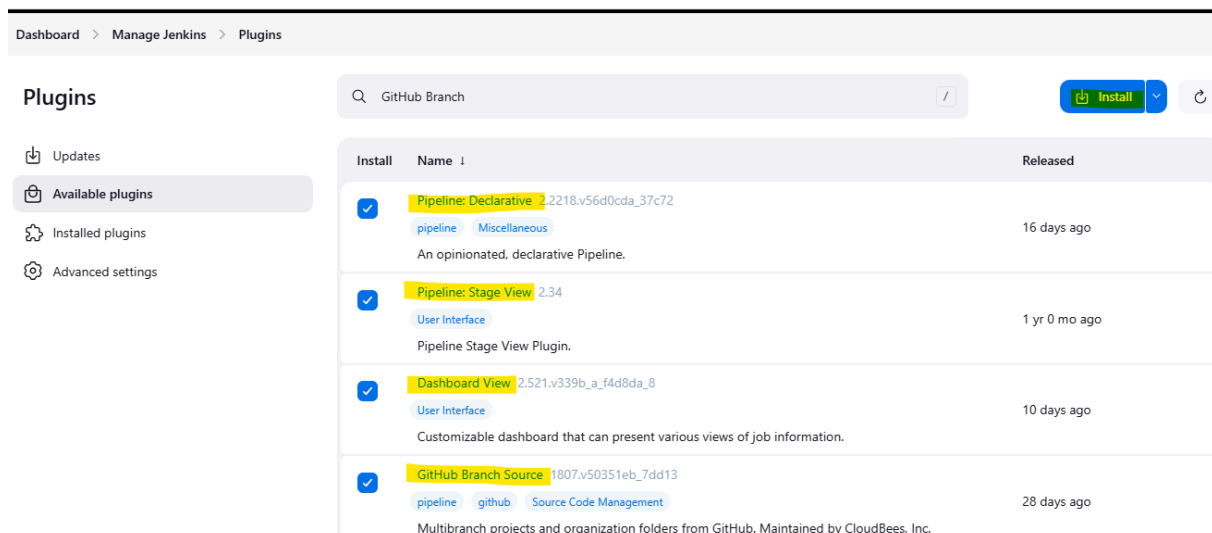
To install the plugins follow below steps:

In dashboard select “manage jenkins” => Plugins



Now, go to **Available plugins** => select mentioned plugin => click on install

Refer attached screenshot.



Steps for creating a project Retail Analysis, initializing Git, and establishing branches on GitHub are as follows:

Prerequisites:

=> In the Jenkins cluster please check the version of python using the below command.

Ex: python 3.11.2 in this Jenkins Cluster

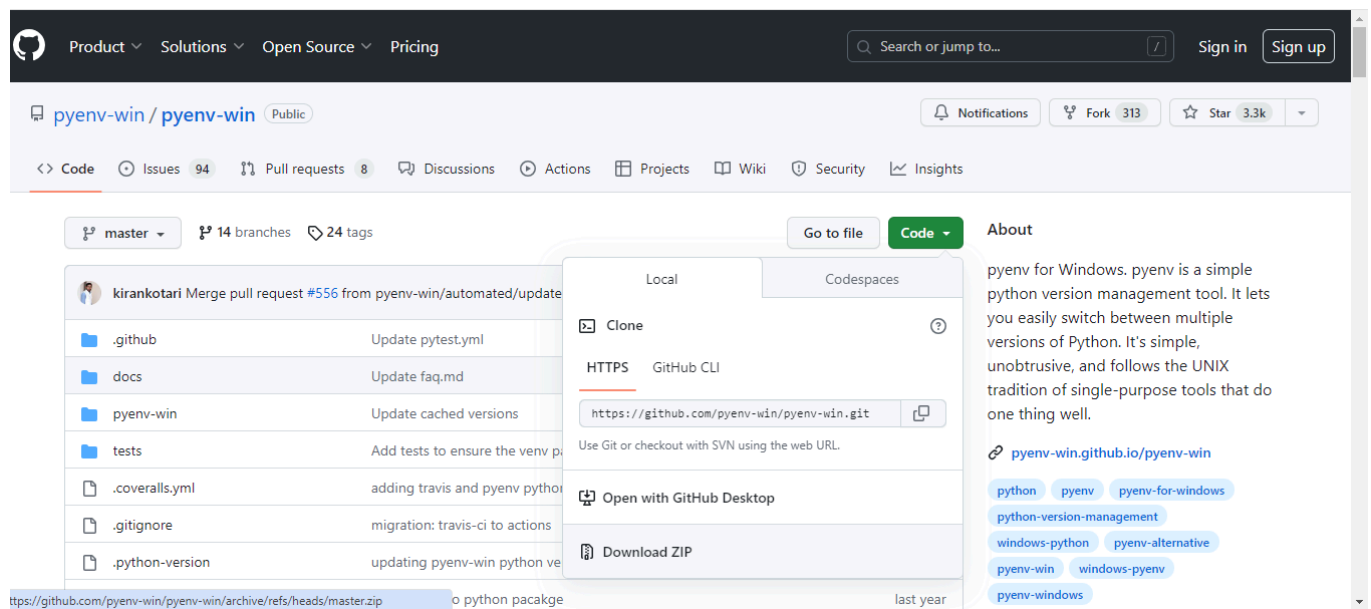
=> Now using pyenv install same version

Follow below steps for installing Pyenv in your system(Local):

Please check the below link to download the pyenv

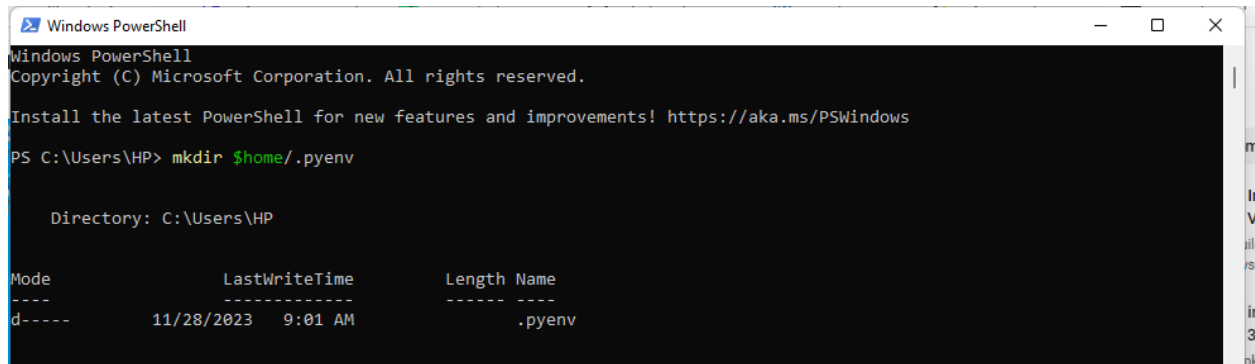
<https://github.com/pyenv-win/pyenv-win>

Now click on code and download the zip file



Go to powershell and run the following command to create pyenv directory in user directory

mkdir \$home/.pyenv



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

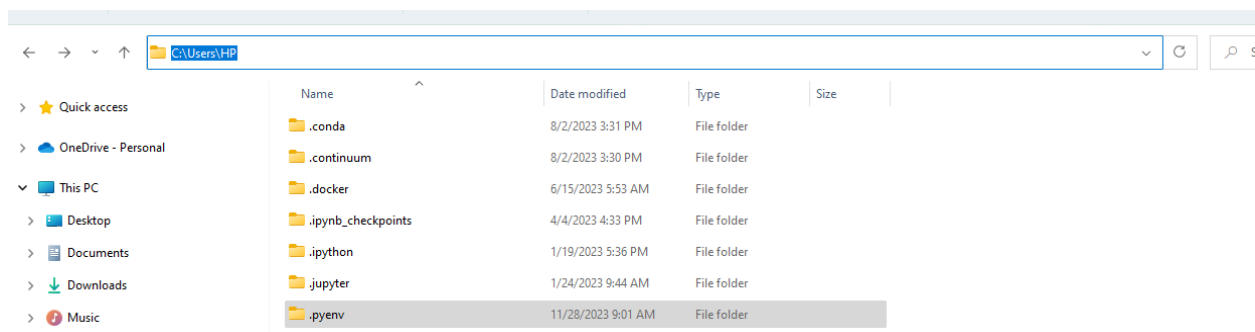
PS C:\Users\HP> mkdir $home/.pyenv

Directory: C:\Users\HP

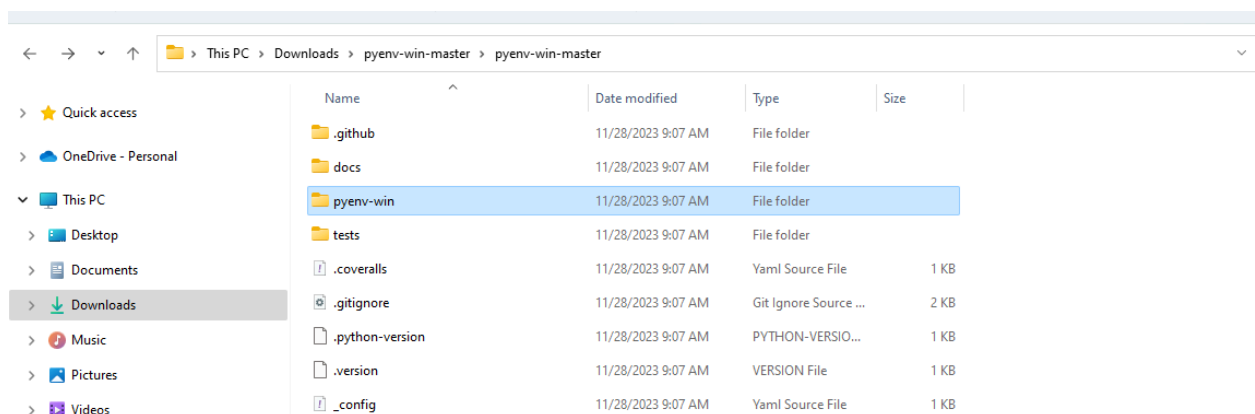
Mode                LastWriteTime         Length Name
----                -
d-----         11/28/2023   9:01 AM             .pyenv
```

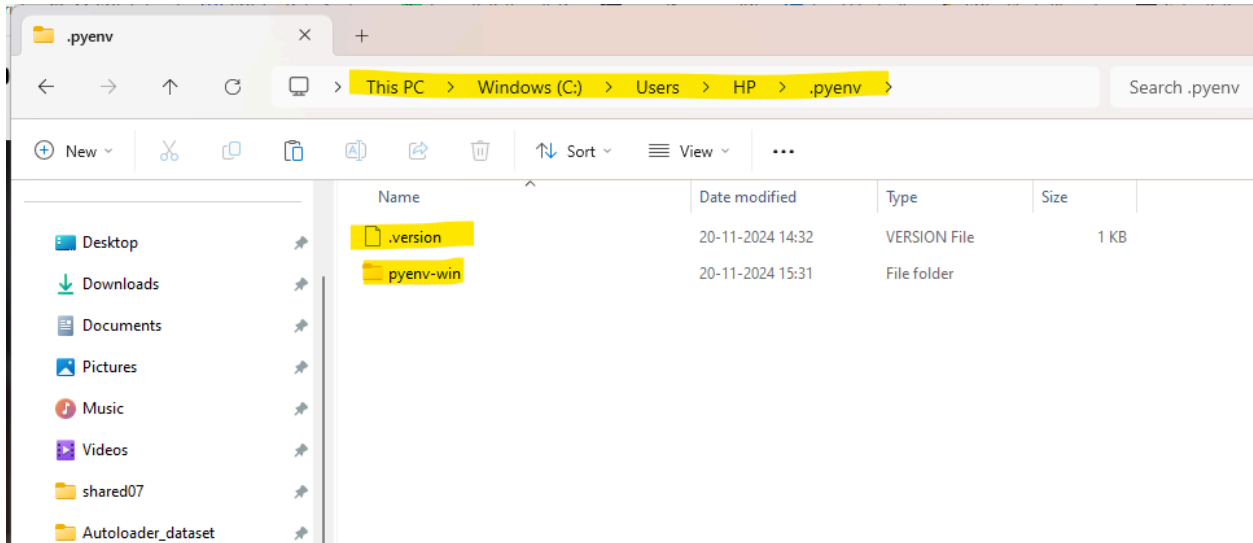
Now go to C => user => <username> and you can see .pyenv folder created.

For below ss username is HP check for your system using above this



And copy files pyenv-win from the extracted files to .pyenv folder refer attached screenshots





Set the environment variables PYENV and PYENV_HOME that point to the installation folder:

PYENV =>

```
[System.Environment]::SetEnvironmentVariable('PYENV',$env:USERPROFILE +  
"\\.pyenv\pyenv-win\","User")
```

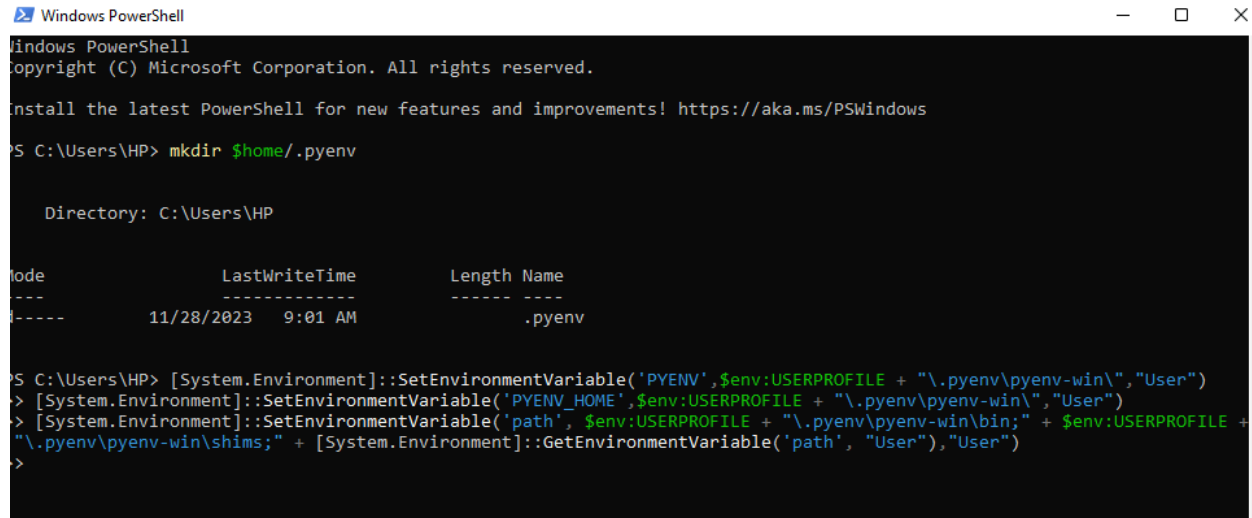
PYENV_HOME =>

```
[System.Environment]::SetEnvironmentVariable('PYENV_HOME',$env:USERPR  
OFILE + "\\pyenv\pyenv-win\","User")
```

Add the bin folder to the PATH variable. Such that pyenv can be found when using the command line.

```
[System.Environment]::SetEnvironmentVariable('path', $env:USERPROFILE +  
"\\.pyenv\pyenv-win\bin;" + $env:USERPROFILE + "\\pyenv\pyenv-win\shims;" +  
[System.Environment]::GetEnvironmentVariable('path', "User"),"User")
```

Refer the below screenshot



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP> mkdir $home/.pyenv

Directory: C:\Users\HP

Mode                LastWriteTime         Length Name
----                -
d-----          11/28/2023   9:01 AM             .pyenv

PS C:\Users\HP> [System.Environment]::SetEnvironmentVariable('PYENV',$env:USERPROFILE + "\.pyenv\pyenv-win\","User")
> [System.Environment]::SetEnvironmentVariable('PYENV_HOME',$env:USERPROFILE + "\.pyenv\pyenv-win\","User")
> [System.Environment]::SetEnvironmentVariable('path', $env:USERPROFILE + "\.pyenv\pyenv-win\bin;" + $env:USERPROFILE +
"\.pyenv\pyenv-win\shims;" + [System.Environment]::GetEnvironmentVariable('path', "User"), "User")
>
```

Close the currently open powershell, start a new PowerShell with admin privileges by right-clicking on the PowerShell icon in the start menu and choose Run as administrator.

Enter the following command into the PowerShell to enable the execution of scripts:

Set-ExecutionPolicy unrestricted

pyenv install --list

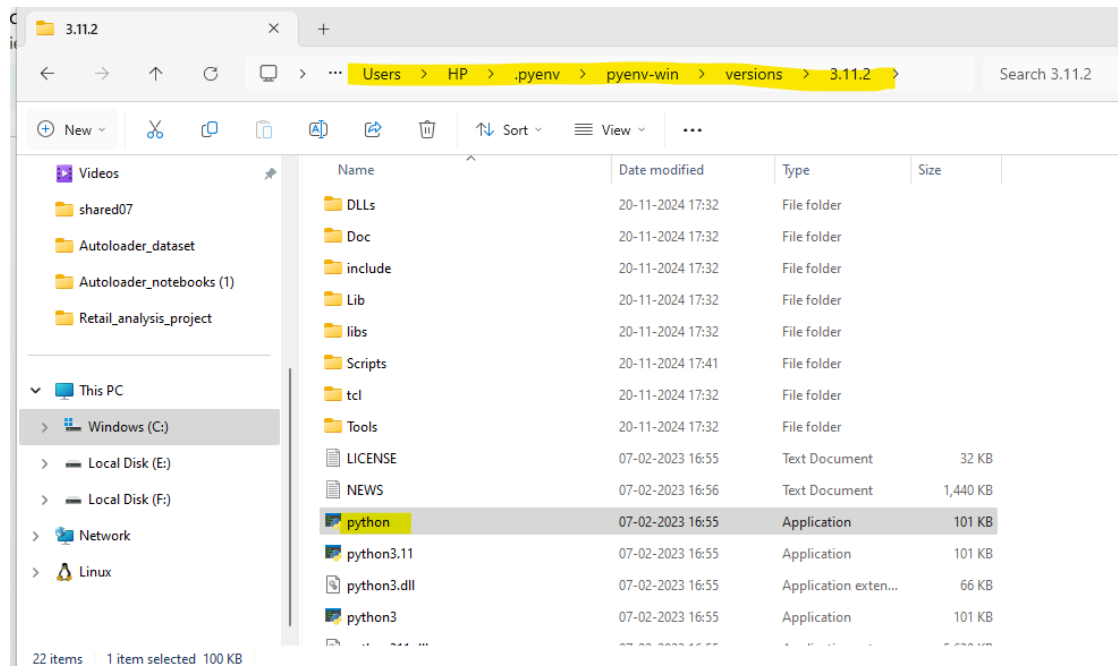
pyenv install 3.11.2

To install the python 3.11.2 version using pyenv use the below command.

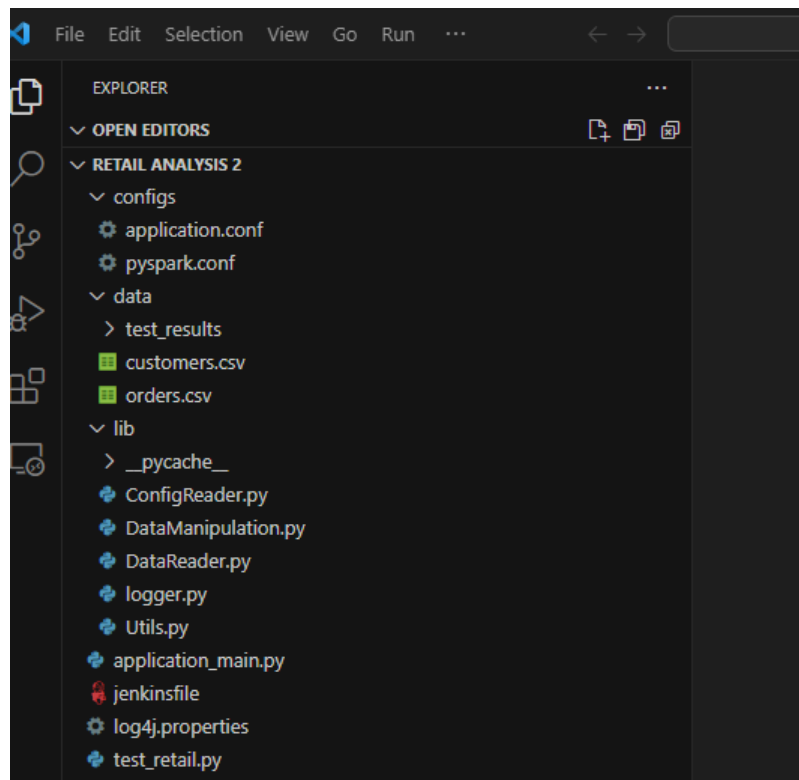
pyenv install 3.11.2

This will be installed in user => <username> (here HP) => .pyenv => pyenv-win
=> versions =>

Ex: "C:\Users\HP\.pyenv\pyenv-win\versions\3.11.2\python.exe"



Create a project Retail Analysis in VS code if you do not have



Sample Code for jenkins file:

```
pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        echo "build completed successfully"
      }
    }

    stage('Test') {
      steps {
        echo "test completed successfully"
      }
    }

    stage('Package') {
      steps {
        echo "package completed successfully"
      }
    }

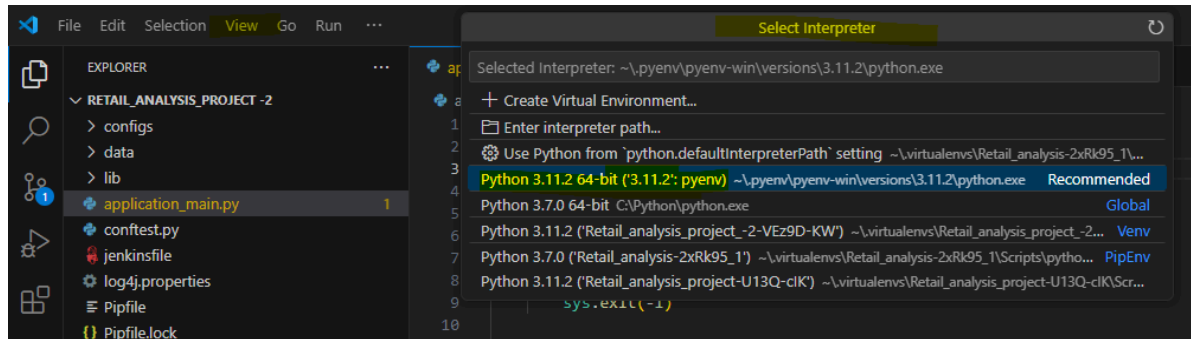
    stage('Deploy') {
      steps {
        echo "deploy completed successfully"
      }
    }
  }
}
```


Follow below steps to set path of virtual environment in VS code:

Now in vs code set path of this python.exe file as interpreter To set the virtual interpreter follow below steps:

Click on "View" > select "Command Palette" > Type "Select Interpreter"

Refer below screenshot for more clarity



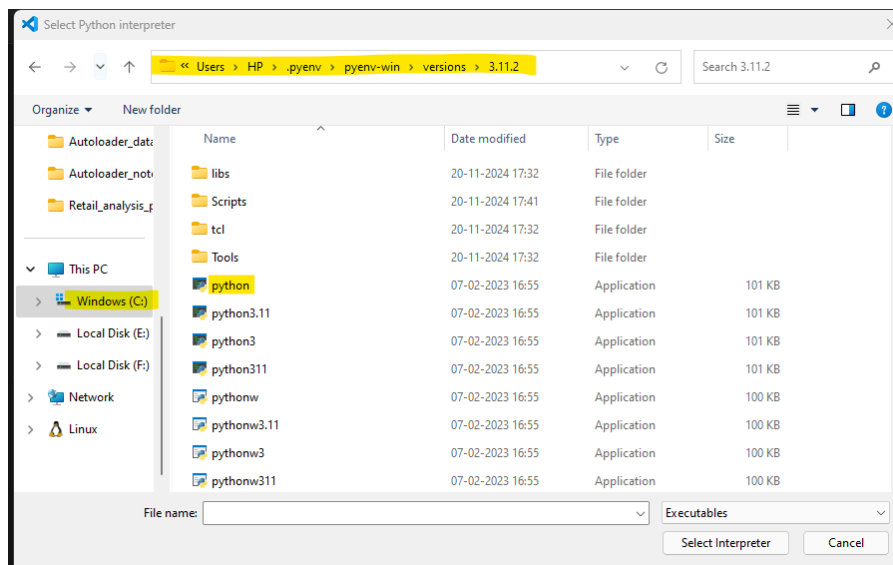
In the list of available interpreters, you should see the Python interpreter. Select the desired Python interpreter from the list (Here pyenv 3.11.2)

If it is not listed you can browse that path refer below steps for more clarity.

Click on "Enter Interpreter path" => Find =>

Using browse option go to C drive => Users => HP => .pyenv => pyenv-win => versions => 3.11.2 => python.exe (select file)

Refer below screenshot for more clarity.

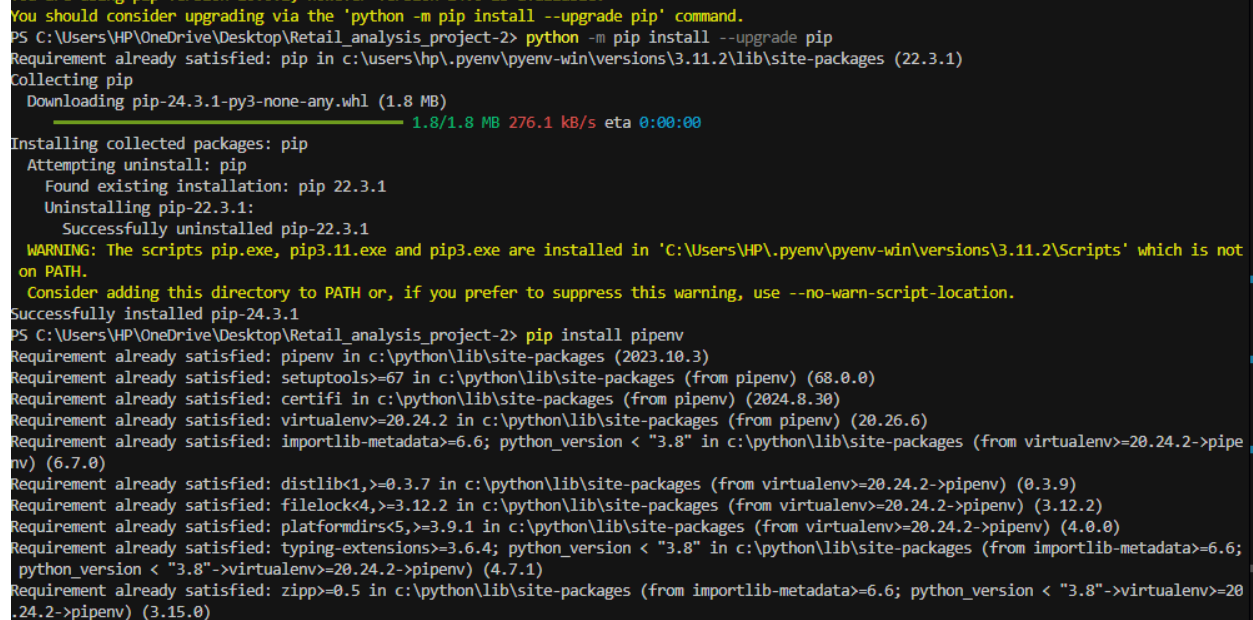


Now first update the “**pip**” and install “**pipenv**” using below commands:

```
pip => python -m pip install --upgrade pip
```

```
pipenv => pip install pipenv
```

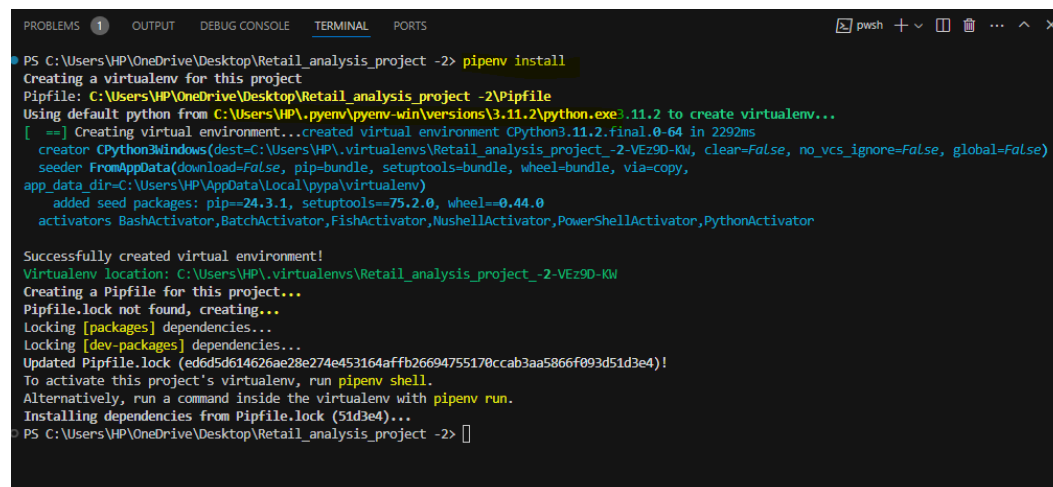
Refer below screenshot for more clarity.



```
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project-2> python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\hp\pyenv\pyenv-win\versions\3.11.2\lib\site-packages (22.3.1)
Collecting pip
  Downloading pip-24.3.1-py3-none-any.whl (1.8 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 276.1 kB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
  WARNING: The scripts pip.exe, pip3.11.exe and pip3.exe are installed in 'C:\Users\HP\pyenv\pyenv-win\versions\3.11.2\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pip-24.3.1
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project-2> pip install pipenv
Requirement already satisfied: pipenv in c:\python\lib\site-packages (2023.10.3)
Requirement already satisfied: setuptools>=67 in c:\python\lib\site-packages (from pipenv) (68.0.0)
Requirement already satisfied: certifi in c:\python\lib\site-packages (from pipenv) (2024.8.30)
Requirement already satisfied: virtualenv>=20.24.2 in c:\python\lib\site-packages (from pipenv) (20.26.6)
Requirement already satisfied: importlib-metadata>=6.6; python_version < "3.8" in c:\python\lib\site-packages (from virtualenv>=20.24.2->pipenv) (6.7.0)
Requirement already satisfied: distlib<1,>=0.3.7 in c:\python\lib\site-packages (from virtualenv>=20.24.2->pipenv) (0.3.9)
Requirement already satisfied: filelock<4,>=3.12.2 in c:\python\lib\site-packages (from virtualenv>=20.24.2->pipenv) (3.12.2)
Requirement already satisfied: platformdirs<5,>=3.9.1 in c:\python\lib\site-packages (from virtualenv>=20.24.2->pipenv) (4.0.0)
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8" in c:\python\lib\site-packages (from importlib-metadata>=6.6; python_version < "3.8"->virtualenv>=20.24.2->pipenv) (4.7.1)
Requirement already satisfied: zipp>=0.5 in c:\python\lib\site-packages (from importlib-metadata>=6.6; python_version < "3.8"->virtualenv>=20.24.2->pipenv) (3.15.0)
```

Now **create virtual environment** in it using command : “**pipenv install**”

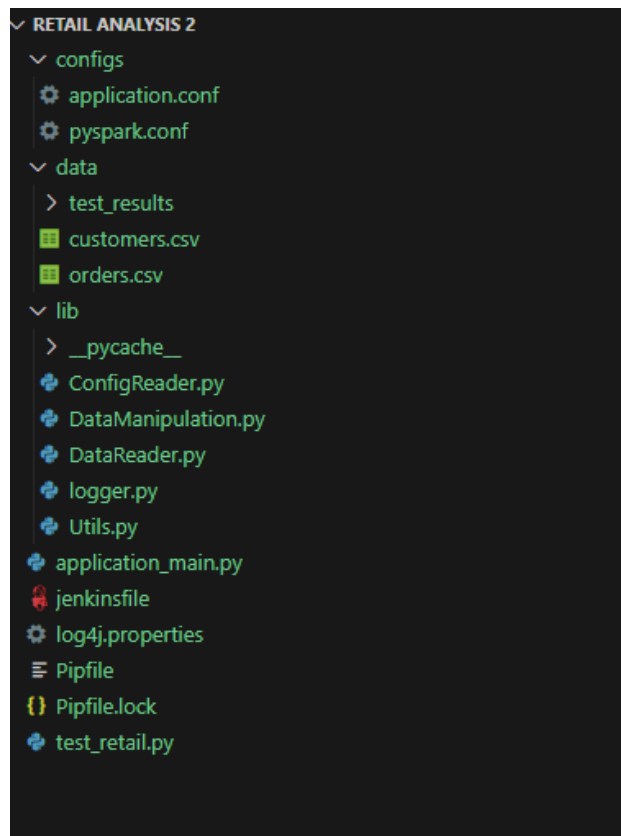
Note: Here as we have to set python version as 3.11.2 so while creating virtual environment python 3.11.2 will be considered as base interpreter.



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project-2> pipenv install
Creating a virtualenv for this project
Pipfile: C:\Users\HP\OneDrive\Desktop\Retail_analysis_project-2\Pipfile
Using default python from C:\Users\HP\pyenv\pyenv-win\versions\3.11.2\python.exe\3.11.2 to create virtualenv...
[ ==] Creating virtual environment...created virtual environment CPython3.11.2.final.0-64 in 2292ms
creator CPythonWindows(dest=C:\Users\HP\virtualenvs\Retail_analysis_project-2-VEz90-KW, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy,
app_data_dir=C:\Users\HP\AppData\Local\pypa\virtualenv)
added seed packages: pip==24.3.1, setuptools==75.2.0, wheel==0.44.0
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

Successfully created virtual environment!
Virtualenv location: C:\Users\HP\virtualenvs\Retail_analysis_project-2-VEz90-KW
Creating a Pipfile for this project...
Pipfile.lock not found, creating...
Locking [packages] dependencies...
Locking [dev-packages] dependencies...
Updated Pipfile.lock (ed6d5d614626ae28e274e453164affb26694755170ccab3aa5866f093d51d3e4)!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (51d3e4)...
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project-2> 
```

Note : It will create pipfile and pipfile.lock



Install **pyspark** using the command: `pipenv install pyspark`

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> pipenv install pyspark
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing pyspark...
Installation Succeeded
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (51d3e4)...
All dependencies are now up-to-date!
Upgrading pyspark in dependencies.
Building requirements...
Resolving dependencies...
Success!
Building requirements...
Resolving dependencies...
Success!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (10e123)...
All dependencies are now up-to-date!
Installing dependencies from Pipfile.lock (10e123)...
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> 
```

Also install **pytest** using command : “**pipenv install pytest**”

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> pipenv install pytest
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing pytest...
Installation Succeeded
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (10e123)...
All dependencies are now up-to-date!
Upgrading pytest in dependencies.
Building requirements...
Resolving dependencies...
Success!
Building requirements...
Resolving dependencies...
Success!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (1fc039)...
All dependencies are now up-to-date!
Installing dependencies from Pipfile.lock (1fc039)...
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> []
```

If your existing project is not under version control, use "**git init**" to start tracking changes.

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git init
Reinitialized existing Git repository in C:/Users/HP/OneDrive/Desktop/Retail_analysis_project -2/.git/
```

To **rename** branch **master** to **main** use command : “**git branch -M main**”

Using the commands “**git add .**” add all the changes

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git branch -M main
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git add .
warning: in the working copy of 'Pipfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Pipfile.lock', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'data/customers.csv', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'data/orders.csv', LF will be replaced by CRLF the next time Git touches it
warning: adding embedded git repository: Retail_analysis_project
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> Retail_analysis_project
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached Retail_analysis_project
hint:
hint: See "git help submodule" for more information.
hint: Disable this message with "git config advice.addEmbeddedRepo false"
```

And using command “git commit -m “performing initial commit”” commit all the changes

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git commit -m "performing initial commit"
[main 73d2b12] performing initial commit
6 files changed, 4 insertions(+), 3 deletions(-)
create mode 160000 Retail_analysis_project
delete mode 100644 __pycache__/conftest.cpython-311-pytest-8.3.3.pyc
delete mode 100644 __pycache__/conftest.cpython-37-pytest-7.4.4.pyc
delete mode 100644 __pycache__/test_retail.cpython-311-pytest-8.3.3.pyc
delete mode 100644 __pycache__/test_retail.cpython-37-pytest-7.4.4.pyc
```

Go to github and create a repo with the name Retail_analysis_2

Refer attached screenshot.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

[aratishatti15](#) / **Retail_analysis_2**

✔ Retail_analysis_2 is available.

Great repository names are short and memorable. Need inspiration? How about [miniature-couscous](#) ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

☐ **Add .gitignore**

Also copy the ssh link as highlighted in the screenshot.

Retail_analysis_2 Public

Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Get started with GitHub Copilot

Add collaborators to this repository

Search for people using their GitHub username or email address.

Invite collaborators

Quick setup — if you've done this kind of thing before

☒ Set up in Desktop or ☒ **HTTPS** ☐ **SSH** **https://github.com/aratishatti15/Retail_analysis_2.git**

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

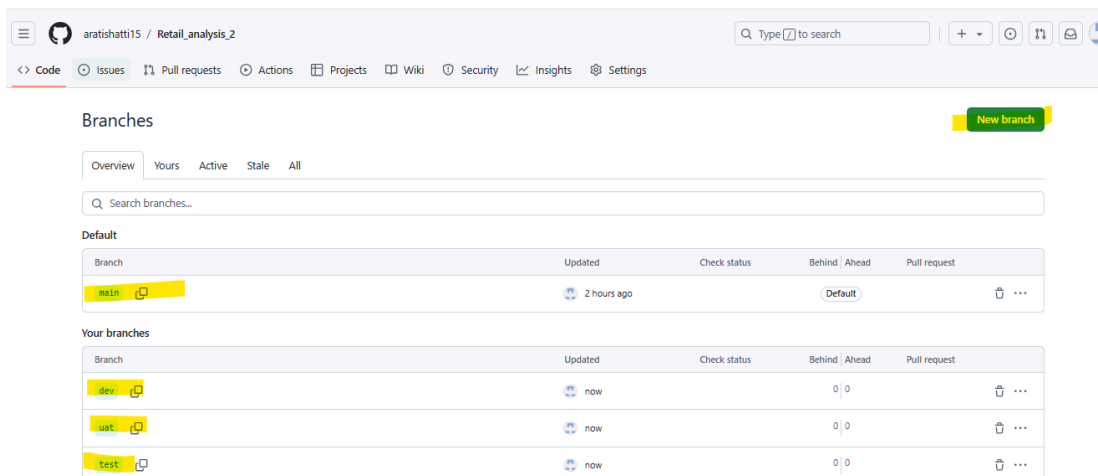
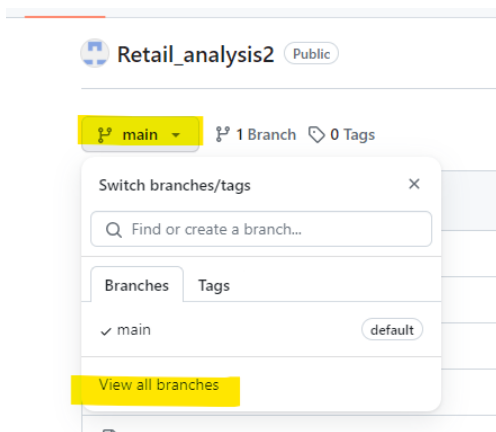
git remote add origin <url of project>

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git remote add origin https://github.com/aratishatti15/Retail_analysis_2.git
```

To push the code to the origin main run the command: **“git push origin main”**

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git push origin main
Enumerating objects: 100, done.
Counting objects: 100% (100/100), done.
Delta compression using up to 8 threads
Compressing objects: 100% (99/99), done.
Writing objects: 100% (100/100), 729.85 KiB | 3.82 MiB/s, done.
Total 100 (delta 41), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (41/41), done.
To https://github.com/aratishatti15/Retail_analysis_2.git
* [new branch]      main -> main
```

Click on view all branches and create the new branches : **main, dev, uat, test**



All have the same code

Create and switch to a new branch in a Git repository on your local machine using the command :

git checkout -b feature-rp-50001

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git checkout -b feature-rp-50001
Switched to a new branch 'feature-rp-50001'
```

After making modifications, commit all the changes, and then push the changes to the remote repository using the command

git push origin feature-rp-50001

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git add .
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git commit -m "performing initial commit in feature"
[feature-rp-50001 a80fe5e] performing initial commit in feature
1 file changed, 1 deletion(-)
delete mode 160000 Retail_analysis_project
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> git push origin feature-rp-50001
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 237 bytes | 237.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature-rp-50001' on GitHub by visiting:
remote:   https://github.com/aratishatti15/Retail_analysis_2/pull/new/feature-rp-50001
remote:
To https://github.com/aratishatti15/Retail_analysis_2.git
 * [new branch]      feature-rp-50001 -> feature-rp-50001
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project -2> 
```

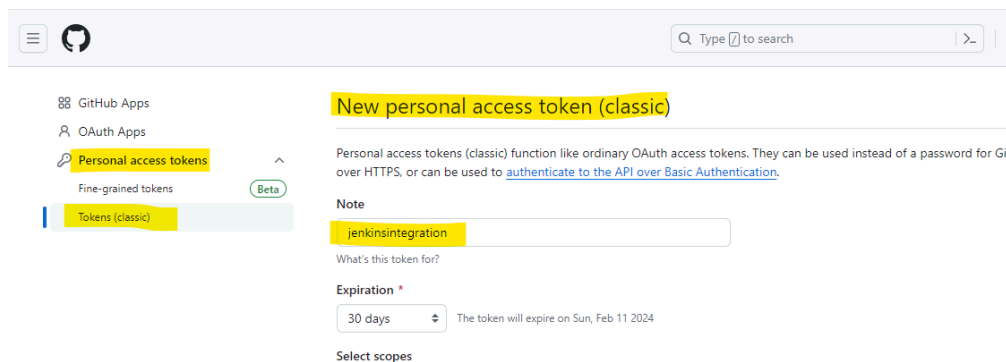
when you execute the above a new feature branch will be created in github

Configuring Jenkins to interpret GitHub events.

Making configurations in Jenkins so that it can read the GitHub events like branch creation, git push, pull request ..etc To establish the connection between GitHub and Jenkins, it has to be done from both ways.

Pre-step: For creation of token in Github:

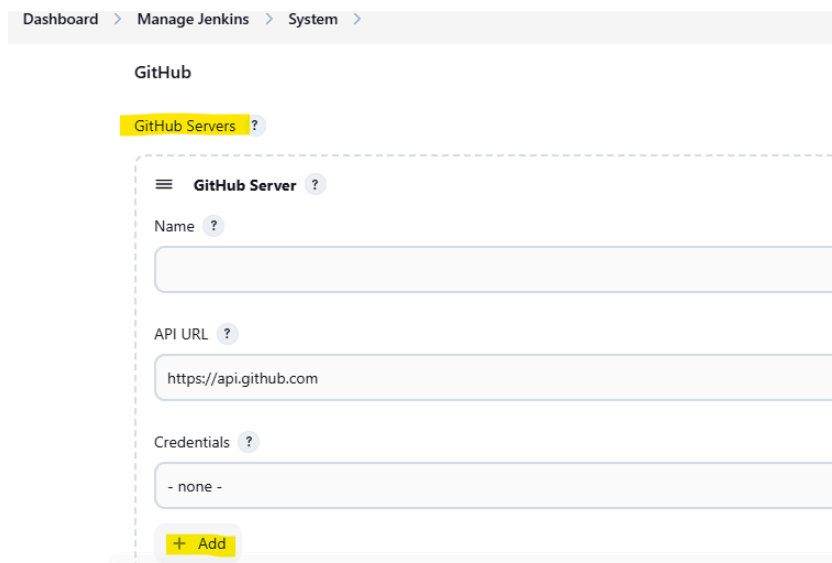
Go to Github => Settings => Developer Settings => Personal Access token => Tokens (Classic) => select all the scopes => generate token => Copy the token



The screenshot shows the GitHub 'New personal access token (classic)' page. On the left sidebar, 'Personal access tokens' is selected, with 'Tokens (classic)' highlighted. The main content area has a title 'New personal access token (classic)' and a note explaining that these tokens function like ordinary OAuth access tokens. Below the note, there is a text input field for the token name, which contains 'jenkinsintegration'. Underneath, there is a section for 'Expiration' with a dropdown menu set to '30 days' and a note that the token will expire on Sun, Feb 11 2024. At the bottom, there is a 'Select scopes' section.

Steps to follow in Jenkins:

Go to Jenkins => Dashboard => Manage Jenkins => System



The screenshot shows the Jenkins 'GitHub Servers' configuration page. The breadcrumb trail at the top reads 'Dashboard > Manage Jenkins > System > GitHub Servers'. The main content area is titled 'GitHub Servers' and contains a form for adding a new server. The form has three input fields: 'Name' (empty), 'API URL' (pre-filled with 'https://api.github.com'), and 'Credentials' (pre-filled with '- none -'). At the bottom of the form, there is a '+ Add' button.

Mention the github token while creating secret text.

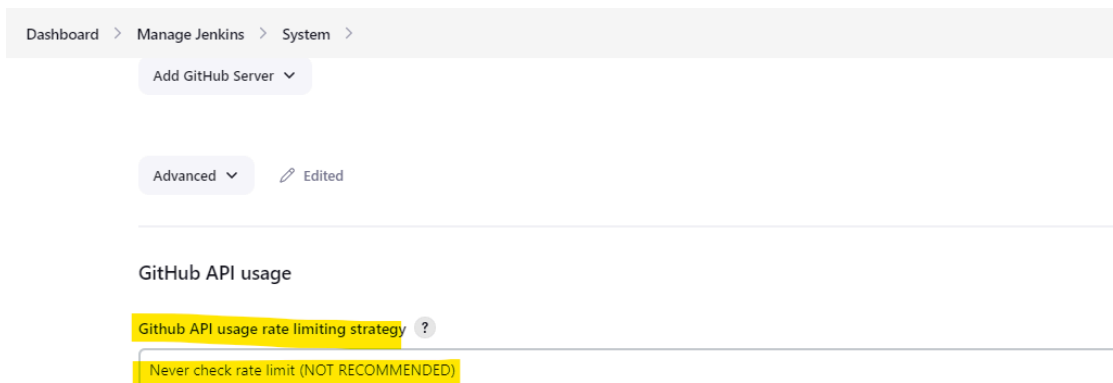


This screenshot shows the 'Add Secret' form in Jenkins. The 'Domain' dropdown is set to 'Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Secret text'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Secret' field contains a masked value represented by dots. The 'ID' field is empty.



This screenshot shows the 'Add GitHub Server' form in Jenkins. The 'Name' field is empty. The 'API URL' field is set to 'https://api.github.com'. The 'Credentials' dropdown is set to 'Secret text'. Below the form, a message states 'Credentials verified for user aratishatti15, rate limit: 4999'. There are 'Save' and 'Apply' buttons at the bottom.

Also set Github Api usage limit strategy as shown below and then save it.



This screenshot shows the 'Add GitHub Server' form in Jenkins, specifically the 'Advanced' settings. The 'GitHub API usage rate limiting strategy' dropdown is set to 'Never check rate limit (NOT RECOMMENDED)'. There are 'Save' and 'Apply' buttons at the bottom.

In Github:

Go to the Repository for Retail Analysis that you created in github => Setting => Webhooks

Now create a new webhook in which you have to mention jenkins URL and “/github-webhook/” as shown below.

<jenkinsurl>/github-webhook/

The screenshot shows the GitHub 'Webhooks' settings page. The left sidebar has a 'Webhooks' link highlighted. The main area is titled 'Webhooks' and contains an 'Add webhook' button. Below this, there is a section for adding a new webhook. It includes a 'Payload URL' field with the value 'http://35.201.205.191//github-webhook/'. The 'Content type' is set to 'application/x-www-form-urlencoded'. There is a 'Secret' field. At the bottom, there is a section titled 'Which events would you like to trigger this webhook?' with a radio button selected for 'Just the push event.'

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://35.201.205.191//github-webhook/

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

☐ Just the push event.

Which events would you like to trigger this webhook? => Let me select individual events.

=> Branch or tag creation, Pull requests, Pushes

Refer attached screenshot.

Which events would you like to trigger this webhook?

☐ Just the push event.

☐ Send me everything.

☒ Let me select individual events.

☒ Branch or tag creation

Branch or tag created.

☐ Branch or tag deletion

Branch or tag deleted.

☐ Branch protection configurations

All branch protections disabled or enabled for a repository.

☐ Branch protection rules

Branch protection rule created, deleted or edited.

☐ Check runs

Check run is created, requested, rerequested, or

☐ Check suites

Check suite is created, requested, rerequested, or

unresolved.

Pull request review submitted, edited, or dismissed.

☒ Pull requests

Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestoned, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked.

☒ Pushes

Git push to a repository.

☐ Registry packages

Registry package published or updated in a repository.

☐ Releases

Release created, edited, published, unpublished, or deleted.

And click on “Add Webhook”. The webhook will be listed as shown below.

The screenshot shows the GitHub repository settings page. On the left, there is a sidebar with navigation links: General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, and Actions. The 'Webhooks' tab is selected and highlighted. In the top right corner, there is a button labeled 'Add webhook'. Below the tab, there is a description of webhooks and a link to the 'Webhooks Guide'. A list of webhooks is displayed, with one entry highlighted: 'http://35.201.205.191/github-webhook... (create, pull_request, and push)'. To the right of this entry are 'Edit' and 'Delete' buttons.

With these steps Both ways connectivity has been done.

Also set the credentials for your multi lab cluster in Jenkins.

Go to jenkins => Manage jenkins => Credentials => System => Global Credentials => Add Credentials

Dashboard > Manage Jenkins

New Item

People

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

System Configuration

System

Tools

Nodes

Clouds

Security

Credentials

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	5e2c236d-1e3a-442b-b4de-0eb524ee26fc	Secret text

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Icon: S M L

Dashboard > Manage Jenkins > Credentials > System

System

Add domain

Domain ↓	Description
Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

Icon: S M L

Provide the username, password and id for your "Multi Lab Cluster," and the system will generate the corresponding credentials as depicted below.

The image shows two screenshots from the Jenkins web interface. The top screenshot is the 'Create' form for 'Global credentials (unrestricted)'. It has a breadcrumb trail: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted). The form includes a dropdown for 'Username with password', a 'Scope' dropdown set to 'Global (Jenkins, nodes, items, all child items, etc)', a 'Username' field with the value 'itv006753', a checkbox for 'Treat username as secret' which is unchecked, a 'Password' field with masked characters, and an 'ID' field with the value 'labcreds'. A blue 'Create' button is at the bottom. The bottom screenshot shows the 'Global credentials (unrestricted)' list page. It has a breadcrumb trail: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted). There is a '+ Add Credentials' button. Below the title, it says 'Credentials that should be available irrespective of domain specification to requirements matching.' There is a table with columns: ID, Name, Kind, and Description. The table contains two entries. The second entry is highlighted with a yellow background. Below the table, there is a legend for 'Icon: S M L'.

ID	Name	Kind	Description
5e2c236d-1e3a-442b-b4de-0eb524ee26fc	Secret text	Secret text	
ff534522-6cee-4e28-aca8-a6eb74be6b0c	itv006753/*****	Username with password	

Note: Modify the Jenkinsfile with the provided code, and proceed to add, commit and push the changes.

```
C:\Users\HP\Desktop\Retail Analysis 2>git add .

C:\Users\HP\Desktop\Retail Analysis 2>git commit -m "Edited jenkins file"
[feature-rp-50001 ba153a1] Edited jenkins file
1 file changed, 27 insertions(+)

C:\Users\HP\Desktop\Retail Analysis 2>git push origin feature-rp-50001
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 395 bytes | 395.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/aratishatti15/Retail_analysis2.git
b499b8b..ba153a1 feature-rp-50001 -> feature-rp-50001
```

Steps for creation of Multibranch pipeline in Jenkins:

Go to the Jenkins UI (Dashboard) => New items => Create the multibranch pipeline

Dashboard > All >

Enter an item name

Retail_analysis_pipeline

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even use for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Dashboard > All >

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Dashboard > Retail_analysis_pipeline > Configuration

Configuration

General

Branch Sources

Build Configuration

Scan Multibranch Pipeline Triggers

Orphaned Item Strategy

Appearance

Health metrics

Properties

General Enabled

Display Name ?
Retail pipeline

Description

Plain text [Preview](#)

Branch Sources

Save Apply

Add the URL of Retail Analysis repository and in discover branches option mention “All branches”

The screenshot shows the 'GitHub' configuration panel in Jenkins. At the top, there's a 'Credentials' dropdown menu currently set to '- none -', with a '+ Add' button below it. A warning message states 'Credentials are recommended'. The 'Repository HTTPS URL' section is selected with a radio button and contains the URL 'https://github.com/aratishatti15/Retail_Analysis_2.git'. Below this, a message says 'Credentials ok. Connected to https://github.com/aratishatti15/Retail_Analysis_2.' and a 'Validate' button is present. At the bottom, the 'Repository Scan - Depreciated Visualization' option is unselected.

The screenshot shows the 'Behaviors' section in Jenkins. Under the 'Discover branches' heading, the 'Strategy' dropdown is set to 'All branches'. Below this is an 'Add' button. The 'Property strategy' dropdown is set to 'All branches get the same properties'. At the bottom, there is an 'Add property' button.

Note: Mention the name of jenkins file as you have created as it is case sensitive

Dashboard > Retail_analysis_pipeline > Configuration

Configuration

- General
- Branch Sources
- Build Configuration**
- Scan Multibranch Pipeline Triggers
- Orphaned Item Strategy
- Appearance
- Health metrics
- Properties

Build Configuration

Mode
by Jenkinsfile

Script Path ?
jenkinsfile

Scan Multibranch Pipeline Triggers
☐ Periodically if not otherwise run ?

Orphaned Item Strategy

Save Apply

After the pipeline is triggered and successfully executed, you'll observe the following output in the Jenkins UI.

Scan Repository Log

Multibranch Pipeline Events

Delete Multibranch Pipeline

People

Build History

GitHub

Rename

Pipeline Syntax

Credentials

Branches (5) Pull Requests (0)

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	dev	18 min #1	N/A	2.3 sec ▶
✓	☁	feature-rp-50001	5 min 31 sec #4	9 min 21 sec #3	10 sec ▶
✓	☀	main	18 min #1	N/A	2.3 sec ▶
✓	☀	test	18 min #1	N/A	2.3 sec ▶
✓	☀	uat	18 min #1	N/A	2.3 sec ▶

This indicates that the pipeline is successfully triggered upon the git push.

Now create the new branch “**git checkout -b feature-rp-50002**”

```
PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project-2> git checkout -b feature-rp-50002
Switched to a new branch 'feature-rp-50002'
```

And push changes to remote using the command “**git push origin feature-rp-50002**”.


```

PS C:\Users\HP\OneDrive\Desktop\Retail_analysis_project-2> git push origin feature-rp-50002
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 945 bytes | 945.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature-rp-50002' on GitHub by visiting:
remote:   https://github.com/aratishatti15/Retail_Analysis_2/pull/new/feature-rp-50002
remote:
To https://github.com/aratishatti15/Retail_Analysis_2.git
 * [new branch]      feature-rp-50002 -> feature-rp-50002

```

After the pipeline is triggered and successfully executed, you'll observe the following output in the Jenkins UI.

Dashboard > Retail pipeline >

Scan Repository Log

Multibranch Pipeline Events

Delete Multibranch Pipeline

People

Build History

GitHub

Rename

Pipeline Syntax

Credentials

Branches (6) Pull Requests (0)

S	W	Name ↓	Last Success	Last Failure
✓	☀	dev	23 min #1	N/A
✓	☁	feature-rp-50001	11 min #4	14 min
✓	☀	feature-rp-50002	2 min 9 sec #1	N/A
✓	☀	main	23 min #1	N/A
✓	☀	test	23 min #1	N/A
✓	☀	uat	23 min #1	N/A

Stage Logs (Deploy)

Print Message -- deploy completed successful (self time 9ms)

deploy completed successful

Changes

Full project name: Retail_analysis_pipeline/feature-rp-50001

Build Now

View Configuration

Full Stage View

GitHub

Pipeline Syntax

Build History trend

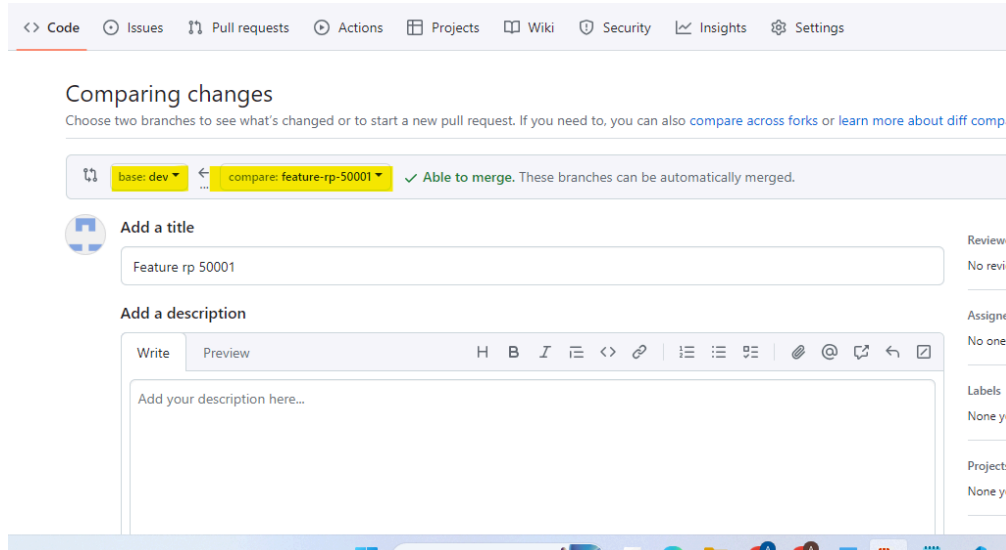
Stage View

Average stage times:
(Average full run time: ~10s)

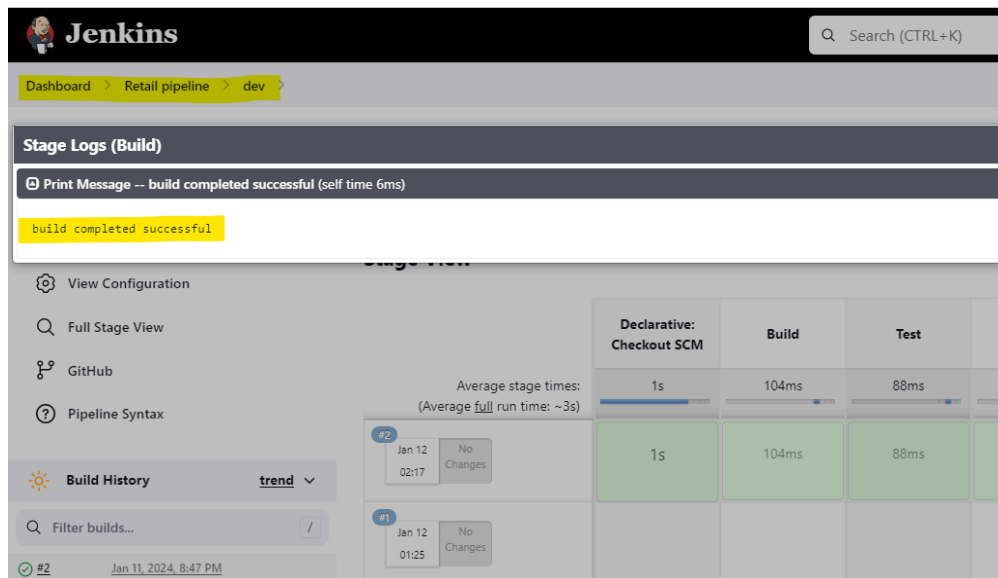
	Declarative: Checkout SCM	Build	Test	Package	Deploy
Average	5s	194ms	130ms	122ms	119ms
#4 Jan 12 01:38 No Changes	5s	194ms	130ms	122ms	119ms

This indicates that the pipeline is successfully triggered upon the creation of a new branch.

Now create and approve the pull request for merging changes of branch “feature-rp-50001” into “dev” branch.



The screenshot shows the GitHub Pull Request interface. At the top, there's a navigation bar with links to Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, the 'Comparing changes' section is visible, showing a comparison between the 'dev' branch and the 'feature-rp-50001' branch. A message states: 'Able to merge. These branches can be automatically merged.' Below this, there's a section for 'Add a title' with a text input field containing 'Feature rp 50001'. To the right of this field are labels for 'Reviewer' (No review), 'Assignee' (No one), 'Labels' (None yet), and 'Projects' (None yet). Below the title field is a section for 'Add a description' with a rich text editor containing the placeholder text 'Add your description here...'. The editor has tabs for 'Write' and 'Preview' and various formatting tools.



The screenshot shows the Jenkins pipeline dashboard. At the top, there's a search bar and a navigation breadcrumb: Dashboard > Retail pipeline > dev. Below this, the 'Stage Logs (Build)' section is visible, showing a message: 'Print Message -- build completed successful (self time 6ms)'. Below this, there's a yellow box with the text 'build completed successful'. Below this, there's a 'View Configuration' section with links to 'Full Stage View', 'GitHub', and 'Pipeline Syntax'. Below this, there's a 'Build History' section with a 'trend' dropdown and a search bar. Below this, there's a table showing the pipeline stages and their execution times.

	Declarative: Checkout SCM	Build	Test
Average stage times: (Average full run time: ~3s)	1s	104ms	88ms
#2 Jan 12 02:17 No Changes	1s	104ms	88ms
#1 Jan 12 01:25 No Changes			

This indicates that the pipeline is successfully triggered upon the creation of a new branch.

To deploy the Retail Project code from our local system to the edge node (In Multi node lab), We will create a Jenkins file for this use case.

Code:

```
pipeline {
  agent any

  environment {
    LABS = credentials('labcreds')
    JAVA_HOME = '/opt/bitnami/java' // Set your JAVA_HOME path here.
    PATH = "${env.JAVA_HOME}/bin:${env.PATH}" // Add Java binaries to PATH
  }

  stages {
    stage('Setup Virtual Environment') {
      steps {
        script {
          // Create a virtual environment with the project name (Retail pipeline)
          sh 'python3 -m venv retail_pipeline_venv'
          // Upgrade pip and install pipenv in the virtual environment
          sh './retail_pipeline_venv/bin/pip install --upgrade pip'
          sh './retail_pipeline_venv/bin/pip install pipenv'
        }
      }
    }
    stage('Install Dependencies') {
      steps {
        script {
          // Install your project dependencies (e.g., requirements.txt or Pipfile)
          sh './retail_pipeline_venv/bin/pipenv install'
        }
      }
    }
    stage('Test') {
      steps {
        script {
          // Ensure JAVA_HOME is set for PySpark to work
          sh 'echo $JAVA_HOME'
          sh 'echo $PATH'

          // Run tests (assuming you are using pytest for tests)
          sh './retail_pipeline_venv/bin/pipenv run pytest'
        }
      }
    }
    stage('Package') {
      steps {
        // Create the zip file but exclude the venv directory
        sh 'zip -r retailproject.zip . -x "retail_pipeline_venv/*"'
      }
    }
  }
}
```

```

    }
  }
  stage('Deploy') {
    steps {
      // Add deployment steps here (e.g., deploy to a server or cloud)
      sh 'sshpass -p $LABS_PSW scp -o StrictHostKeyChecking=no -r
retailproject.zip $LABS_USR@g02.itversity.com:/home/itv012419/retailproject'
    }
  }
}

```

Note: We have updated this code as compared to “Sumit” Sir’s Code as there were some issues with Jenkins cluster creation so based on requirement we have created this Jenkins file.

Modify the Jenkinsfile with the provided code in feature-rp-50002, and proceed to add, commit and push the changes.

```

● PS C:\Users\HP\Desktop\Retail Analysis 2> git add .
● PS C:\Users\HP\Desktop\Retail Analysis 2> git commit -m "Modified jenkins file"
[feature-rp-50002 9934cfd] Modified jenkins file
 1 file changed, 14 insertions(+), 9 deletions(-)
● PS C:\Users\HP\Desktop\Retail Analysis 2> git push origin feature-rp-50002
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 579 bytes | 13.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/aratishatti15/Retail_analysis2.git
 e462ae2..9934cfd feature-rp-50002 -> feature-rp-50002
PS C:\Users\HP\Desktop\Retail Analysis 2> 

```

Now your pipeline will run and it will zip all the files in the Retail project folder and using scp command it will copy those files to the mentioned path in edge node and you can see the retail project folder in edge node of our multi node lab.

Note: If you face below error after updating code of Jenkins file then run the command “sudo apt install python3.11-venv” and again scan your pipeline:

S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓	☀	dev	1 hr 3 min #1	N/A	3.1 sec	▶
✓	☀	feature-rp-50001	1 hr 6 min #1	N/A	9.9 sec	▶
✗	☁	feature-rp-50002	N/A	5 min 30 sec #1	4 sec	▶
✓	☁	main	1 hr 4 min #2	1 hr 6 min #1	6.1 sec	▶
✓	☀	test	1 hr 2 min #1	N/A	3 sec	▶
✓	☀	uat	14 min #2	N/A	8 min 53 sec	▶

```

Stage Logs (Setup Virtual Environment)

Shell Script -- python3 -m venv retail_pipeline_venv (self time 301ms)

+ python3 -m venv retail_pipeline_venv
The virtual environment was not created successfully because ensurepip is not
available. On Debian/Ubuntu systems, you need to install the python3-venv
package using the following command.

    apt install python3.11-venv

You may need to use sudo with that command. After installing the python3-venv
package, recreate your virtual environment.

Failing command: /bitnami/jenkins/home/workspace/Retail_pipeline_feature-rp-50002/retail_pipeline_venv/bin/python3

```

Follow below steps to resolve it:

On your Jenkins server, install the missing package:

`sudo apt update`

`sudo apt install python3.11-venv`

```

-1-vm:~$ sudo apt install python3.11-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpython3.11-minimal libpython3.11-stdlib python3-distutils python3-lib2to3 python3-pip-whl
  python3-setuptools-whl python3.11 python3.11-minimal
Suggested packages:
  python3.11-doc binfmt-support
The following NEW packages will be installed:
  python3-distutils python3-lib2to3 python3-pip-whl python3-setuptools-whl python3.11-venv
The following packages will be upgraded:
  libpython3.11-minimal libpython3.11-stdlib python3.11 python3.11-minimal
4 upgraded, 5 newly installed, 0 to remove and 45 not upgraded.

```

After successfully running the pipeline it will be reflected as shown below in Jenkins ui.

Dashboard > Retail Analysis Pipeline > feature-rp-50002 >

Status

</> Changes

Build Now

View Configuration

Full Stage View

GitHub

Pipeline Syntax

Builds

Filter

Today

#2 11:23 AM

#1 11:18 AM

feature-rp-50002

Full project name: Retail pipeline/feature-rp-50002

Stage View

Average stage times:
(Average full run time: ~2min 2s)

	Declarative: Checkout SCM	Setup Virtual Environment	Install Dependencies	Test	Package	Deploy
#2 Nov 22 16:53 No Changes	897ms	12s	1min 0s	27s	605ms	19s
#1 Nov 22 16:48 No Changes	1s	527ms	89ms	82ms	94ms	84ms

Permalinks

And the zipped file will be copied to your external lab.