Modern DevOps Practices on IBMi

Practical Guide

Table of Contents

Pre-requisites

1. A GitHub Account
2. VS Code Editor
3. An IBM i server running 7.2 or above
4. Some patience and desire to learn something new.

Connect to IBMI from VS Code

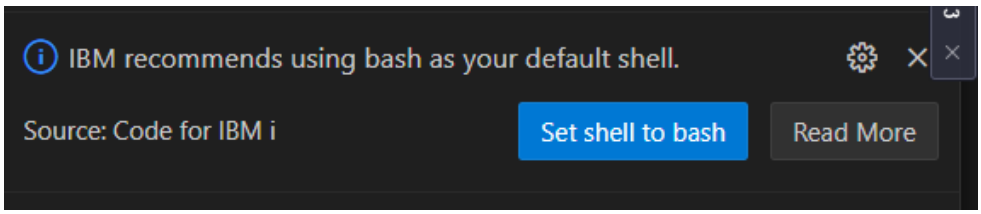
> *because, it is easy to execute shell commands and edit IFS files in VS Code!*

- Connect to IBMI via VS Code. You should knew that now already. If not, check this link and come back here once you have connected your IBMi.
- Enter `Ctrl + Shift + J` (once connected to the IBM I via VS Code) and select **PASE** terminal.

Set Shell to BASH

> *because, the default shell is very limiting and irritating!*

Either set it via VS Code.



OR

Enter the below command in the PASE terminal. *Don't forget to replace cecuser with your username*

```
/QOpenSys/pkgs/bin/chsh -s /QOpenSys/pkgs/bin/bash cecuser
```

Set Open Source path/env variables

> *because, we need to tell the IBMI where to locate the open source linux commands.*

In order to be able to run the linux commands without specifying the location of the command, we have to setup the Open Source Path/Environment Variables. Do the below steps to do so.

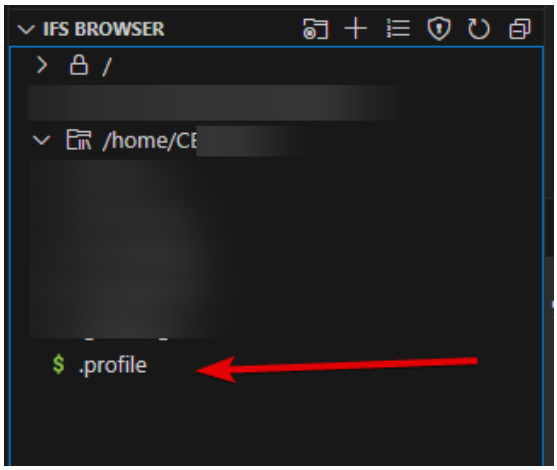
Setup for PASE/SSH terminal

Follow the below steps if you decide to run the applications from the PASE/SSH Terminal

- Navigate to your home folder by entering `cd ~`

- Enter command `touch .profile` in order to create a new file called *.profile*

- Open the file `.profile` using VS Code's IFS Browser. If you don't see the file, then click on the refresh icon on the top right side of the IFS browser panel.



- Copy paste the below content on the `.profile` file.

```
export PATH=/QOpenSys/pkgs/bin:$PATH
export JAVA_HOME=/QOpenSys/QIBM/ProdData/JavaVM/jdk17/64bit
export JENKINS_HOME=/home/CECUSER/jenkins
export GITBUCKET_HOME=/home/CECUSER/gitbucket
source ~/.git-prompt.sh
PROMPT_COMMAND='__posh_git_ps1 "${VIRTUAL_ENV:+(`basename $VIRTUAL_ENV`)}\[\e[32m\]\u\[\e[0m\]@\h:\[\e[33m\]\w\
```

> **Explanation:**
>
> #1: The open source linux commands are available in the path `/QOpenSys/pkgs/bin` , so we are appending that location to the already available `$PATH` variable.
>
> #2: The default JAVA version in IBMi sometimes would be 8. But Jenkins require version 11 or above. So we will tell IBMi to use the latest version of JAVA (17 in our case) for running Jenkins.
>
> #3: We are setting up the Jenkins' application on a folder called 'jenkins'. It provides better management of application, such as the whole application can be uprooted and planted in another location if required.
>
> #4: Similarly, we are setting the Gitbucket's application on a folder called 'gitbucket'
>
> #5 & #6: This is required for changing the command line prompt to display your "python virtual environment(if activated)" "username", "servername", "present working directory" and show the current git branch and git status at all the times.

- Create another file called `git-prompt.sh` by entering the below command.

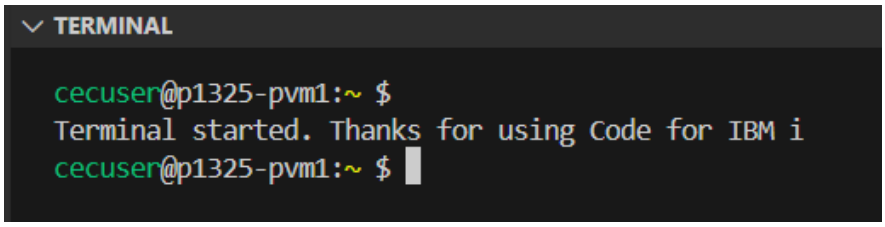
```
touch git-prompt.sh
```

- Once an empty file is created, open the file in VS Code and copy the entire code from this link and paste it on to the empty file and save it.

Verify the setup

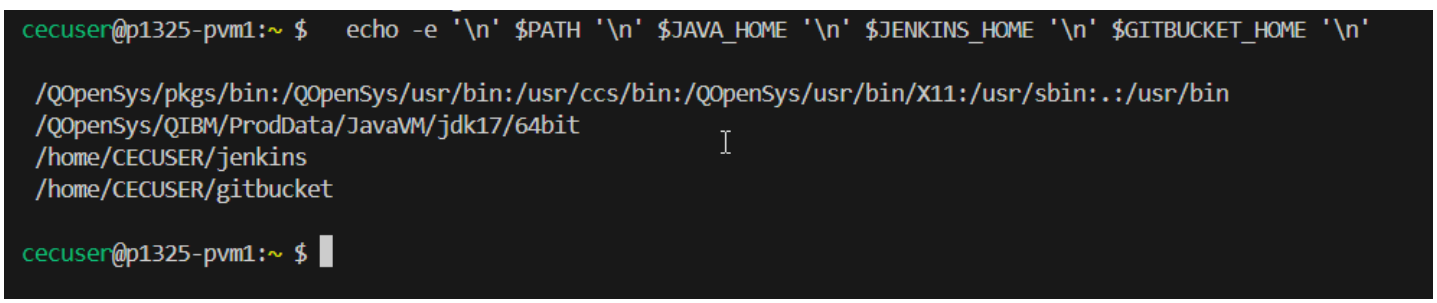
Once the initial setup is complete,

- Open up a new PASE terminal by entering `Ctrl+Shift+J` . If the shell is set to bash successfully, you should see the below screen



- Run the below command to check whether the path variable has been setup correctly.

```
echo -e '\n' $PATH '\n' $JAVA_HOME '\n' $JENKINS_HOME '\n' $GITBUCKET_HOME '\n'
```



Update yum packages

In the PASE Terminal, enter the below command,

```
yum update -y && yum upgrade -y
```

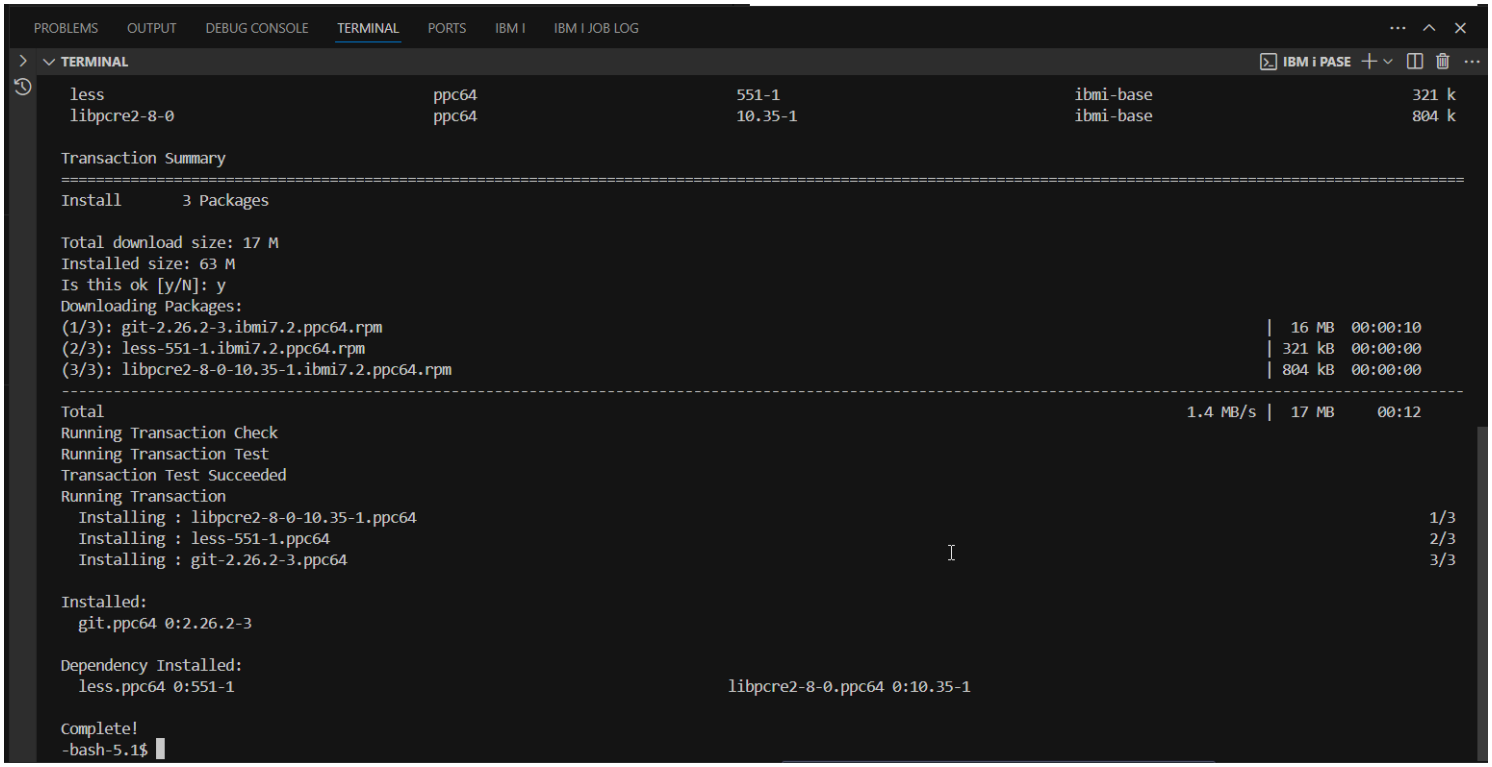
> **Warning!:** You want to run the update with caution as it might break the existing OSS applications. For that reason, it is best to use `chroot` to create separate container for this purpose.

Install

Install GIT

Enter the command below in your PASE terminal.

```
yum install git -y
```



> *GIT has been installed successfully*

Setup GITHUB

Let's connect our IBMi with the GitHub and try pushing (a.k.a. updating our sources) directly to the GitHub Repository.

Setup the user name and email for your local git

- Enter the commands below

```
git config --global user.name 'Ravisankar Pandian'
git config --global user.email ravisankar.pandian@programmers.io
```

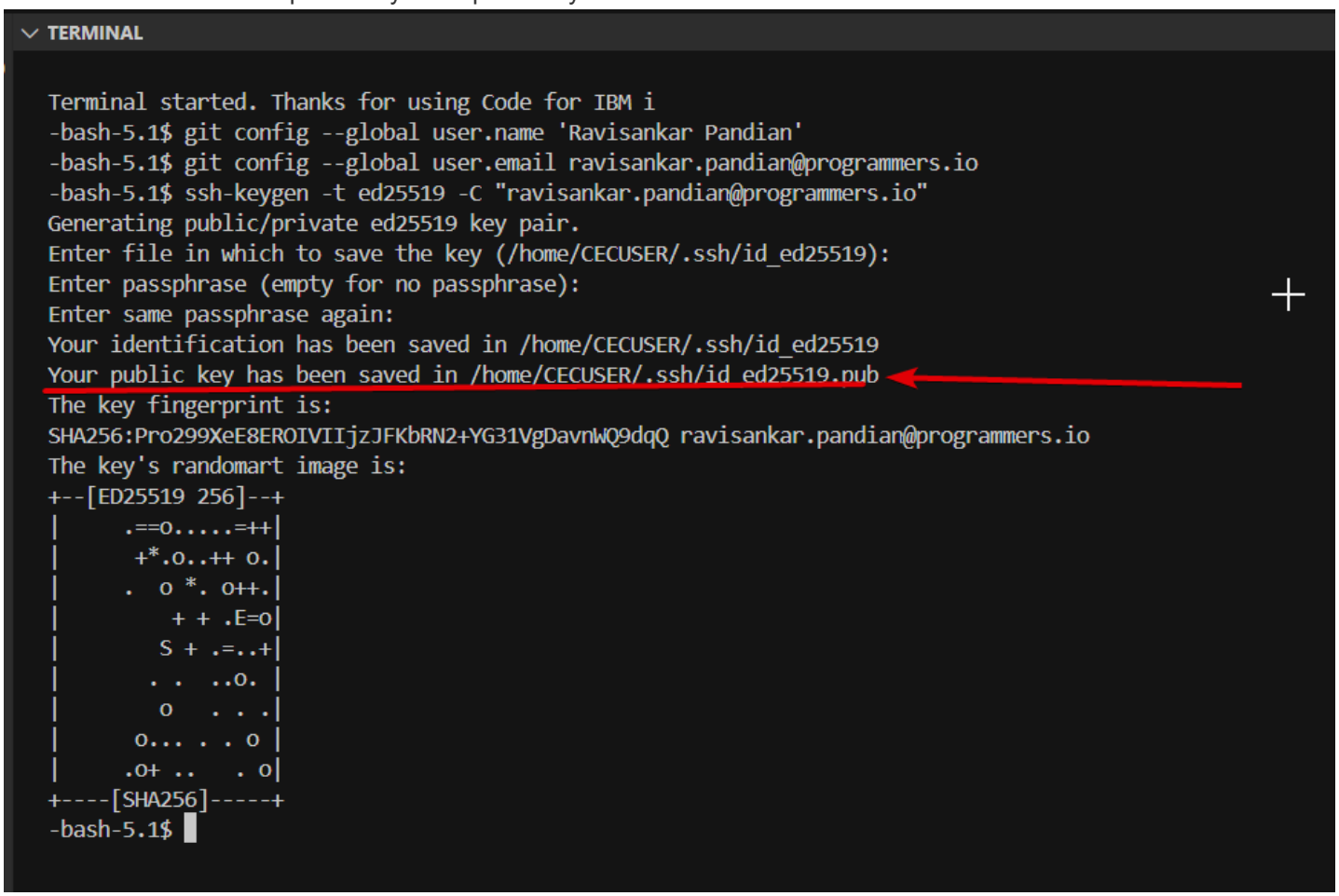
> *Make sure to enter the email ID that you use to login to your GitHub account*

Generate a public/private keypair.

- Enter the below command in your PASE terminal. (make sure to enter the email id that you use in your github account)

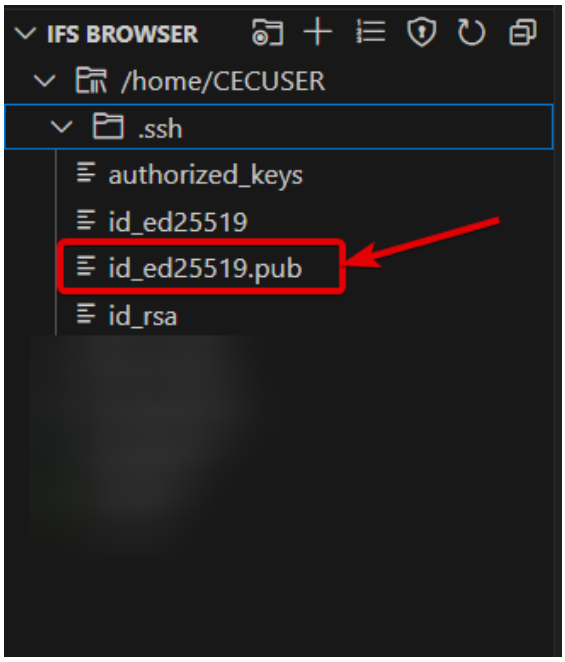
```
ssh-keygen -t ed25519 -C "ravisankar.pandian@programmers.io"
```

- Hit enter again to save the key pair at the default location itself.
- Hit enter again (no passphrase is required)
- Notice the location of the public key and open it in your VS Code.



Copy the public key

- Navigate to the same folder in your VS Code as below and open the public key

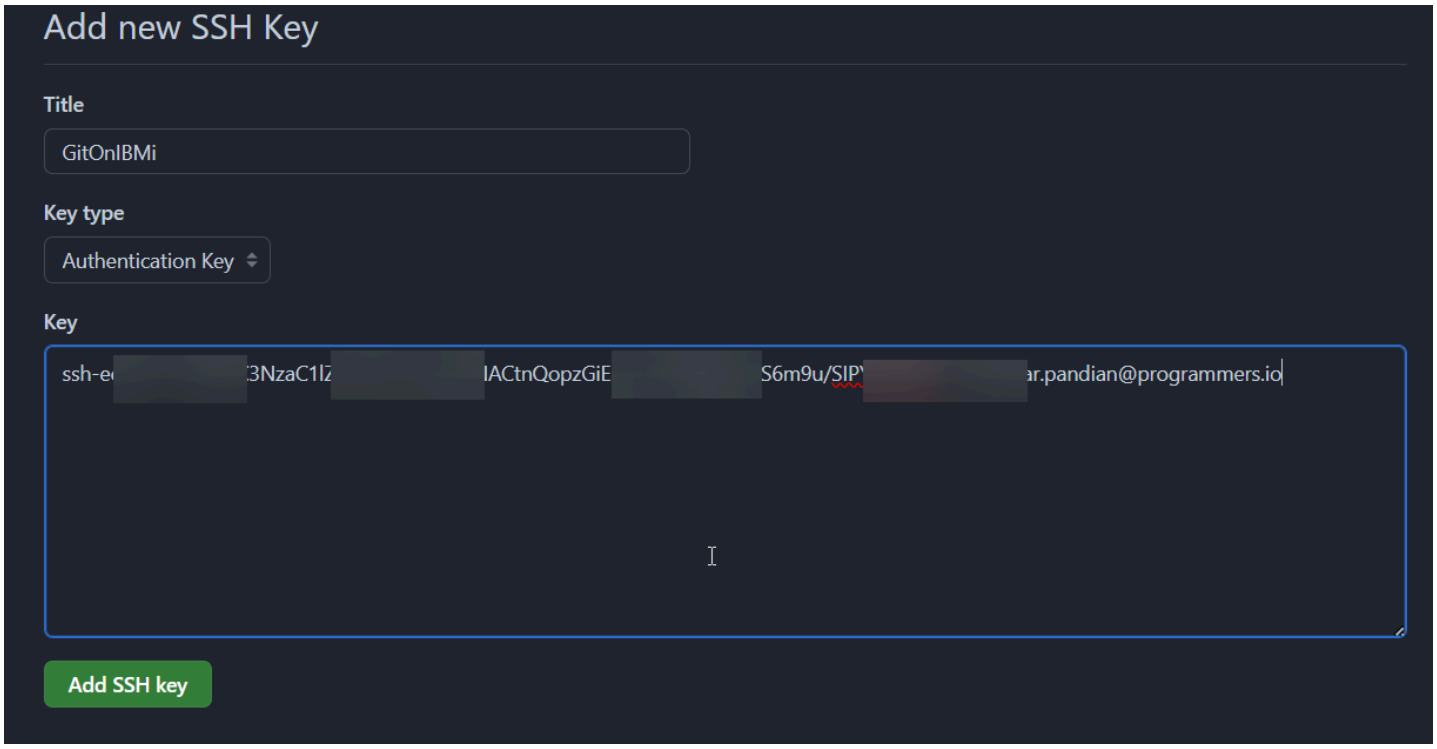


- Copy all the contents of the file. We need to put that into our GitHub account.

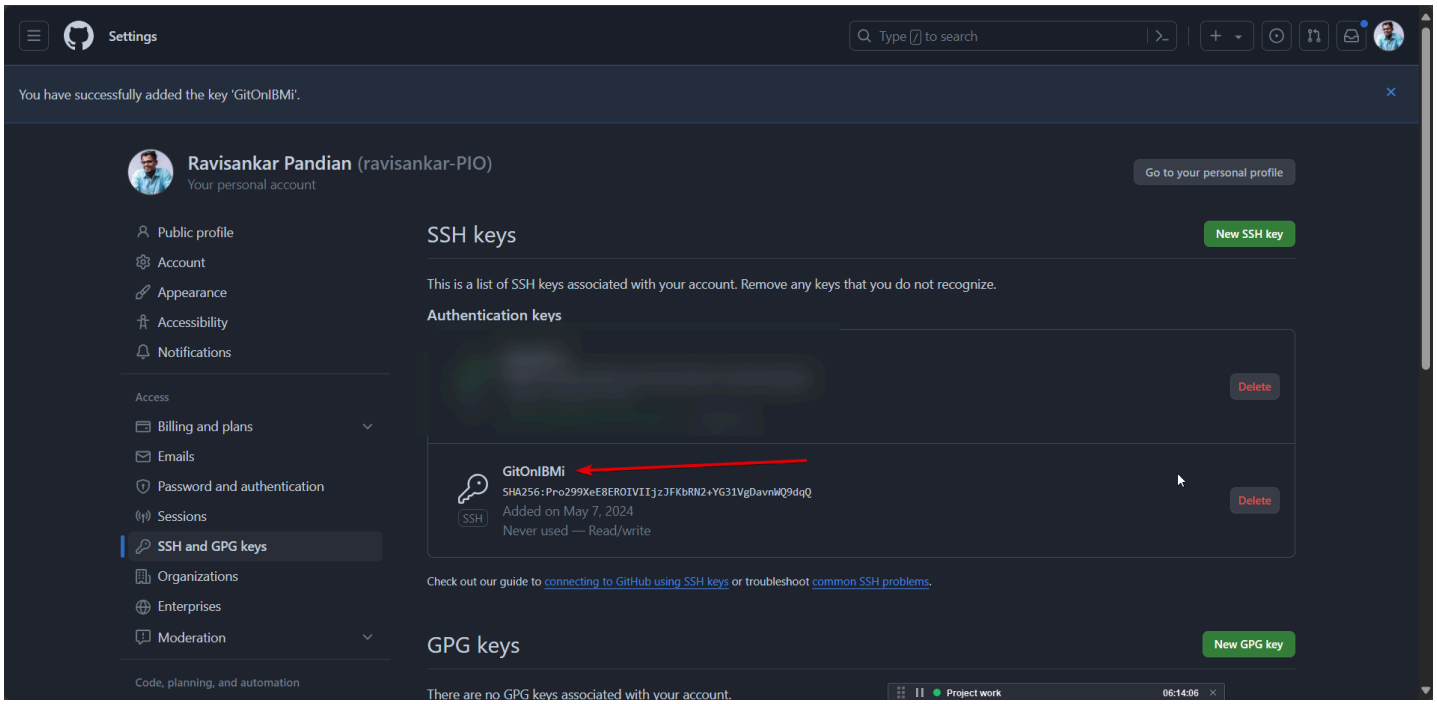
Create New SSH Key in your GitHub account

- Open https://github.com/settings/keys and click `New SSH Key` .

- Enter some title, Select the key type as "authentication key", paste the previously copied public key, and finally select 'Add SSH Key"



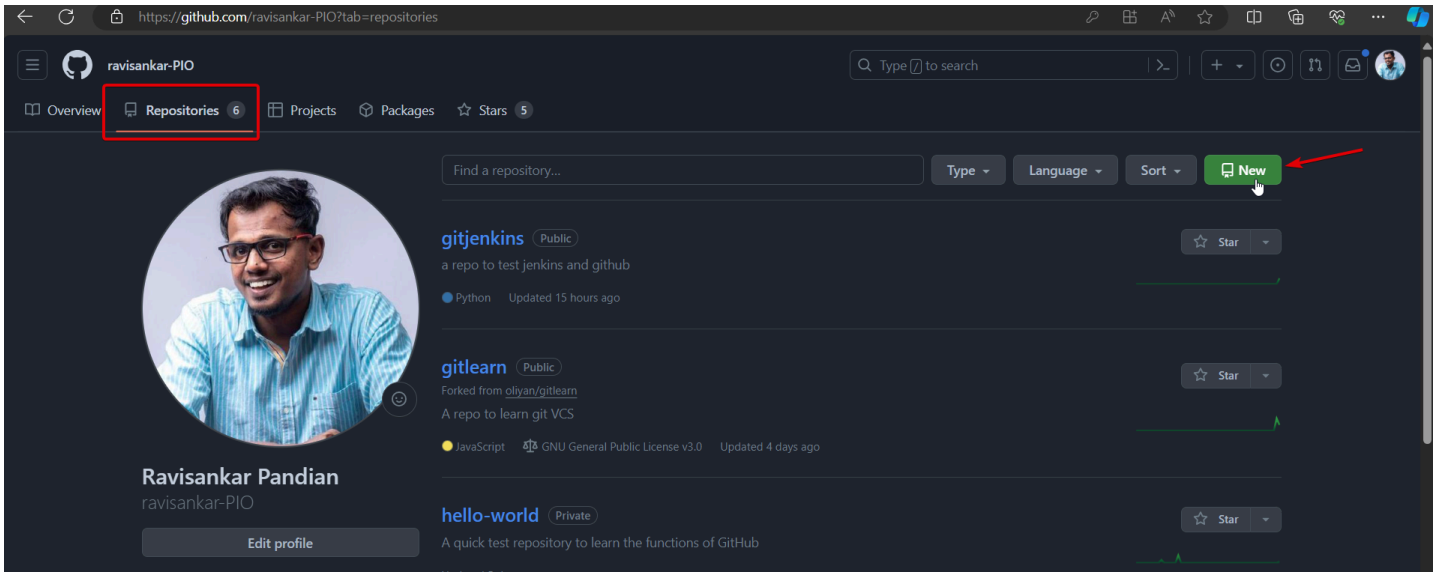
- Once added, you should see the below screen



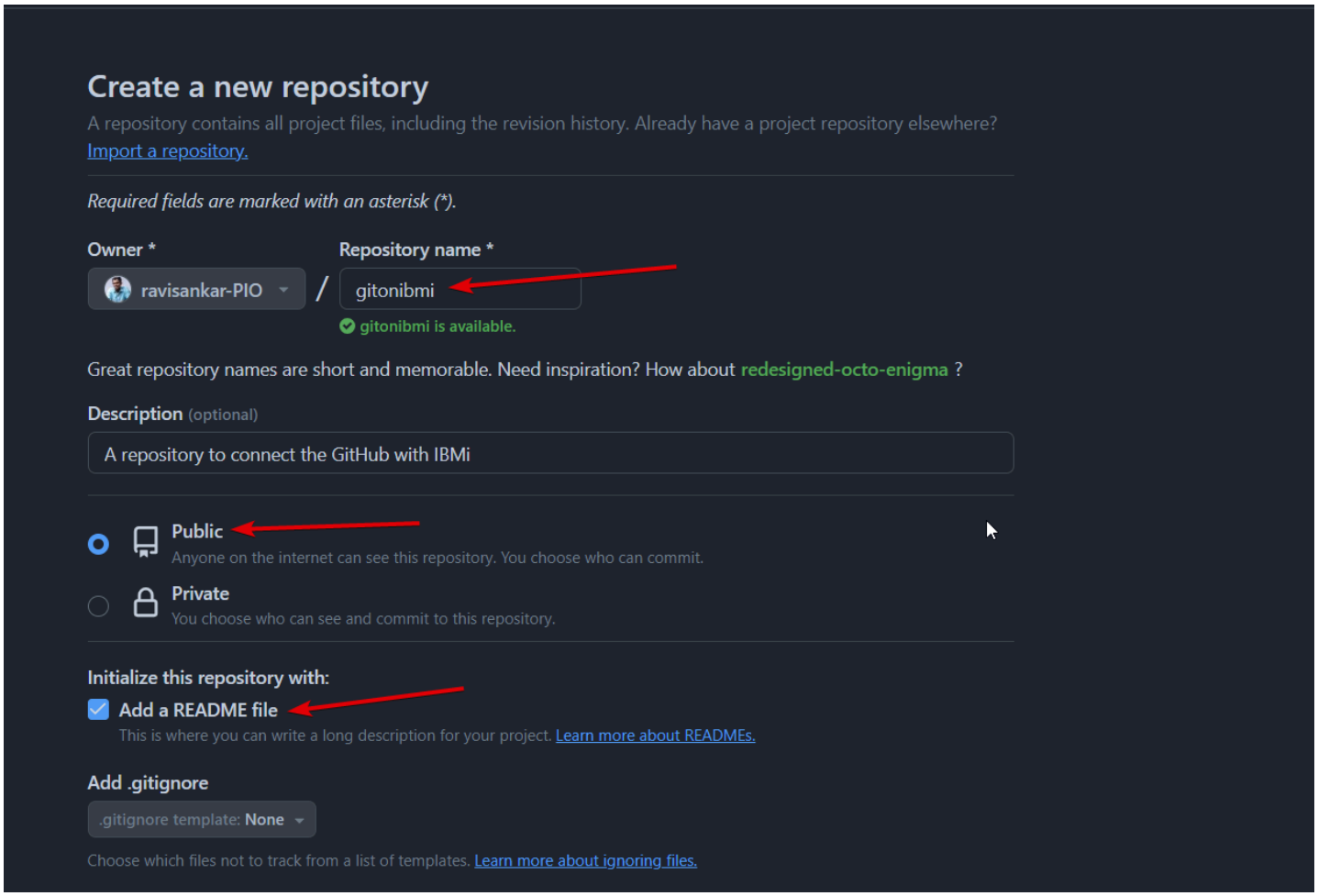
Create a GitHub repository

- Let's create a new empty repository in our GitHub account.

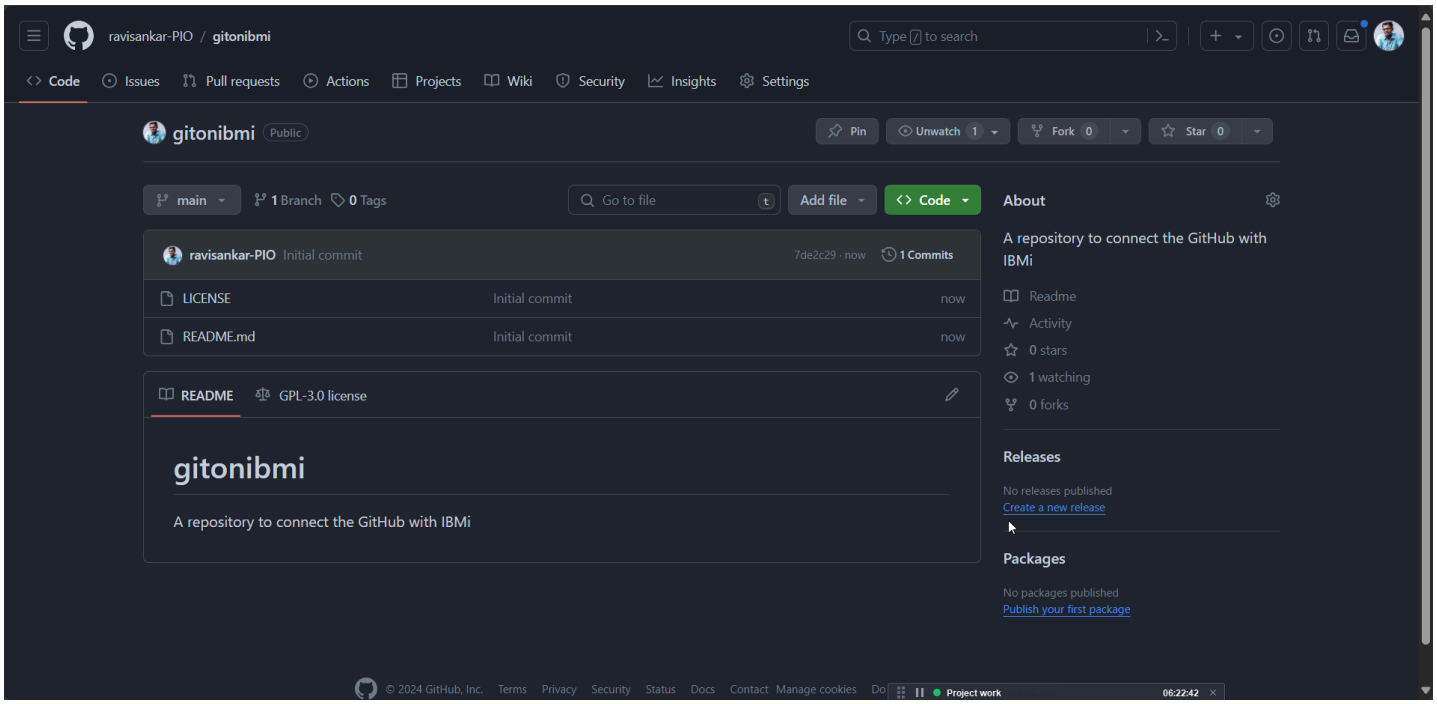
- Click on Repository >> Click New



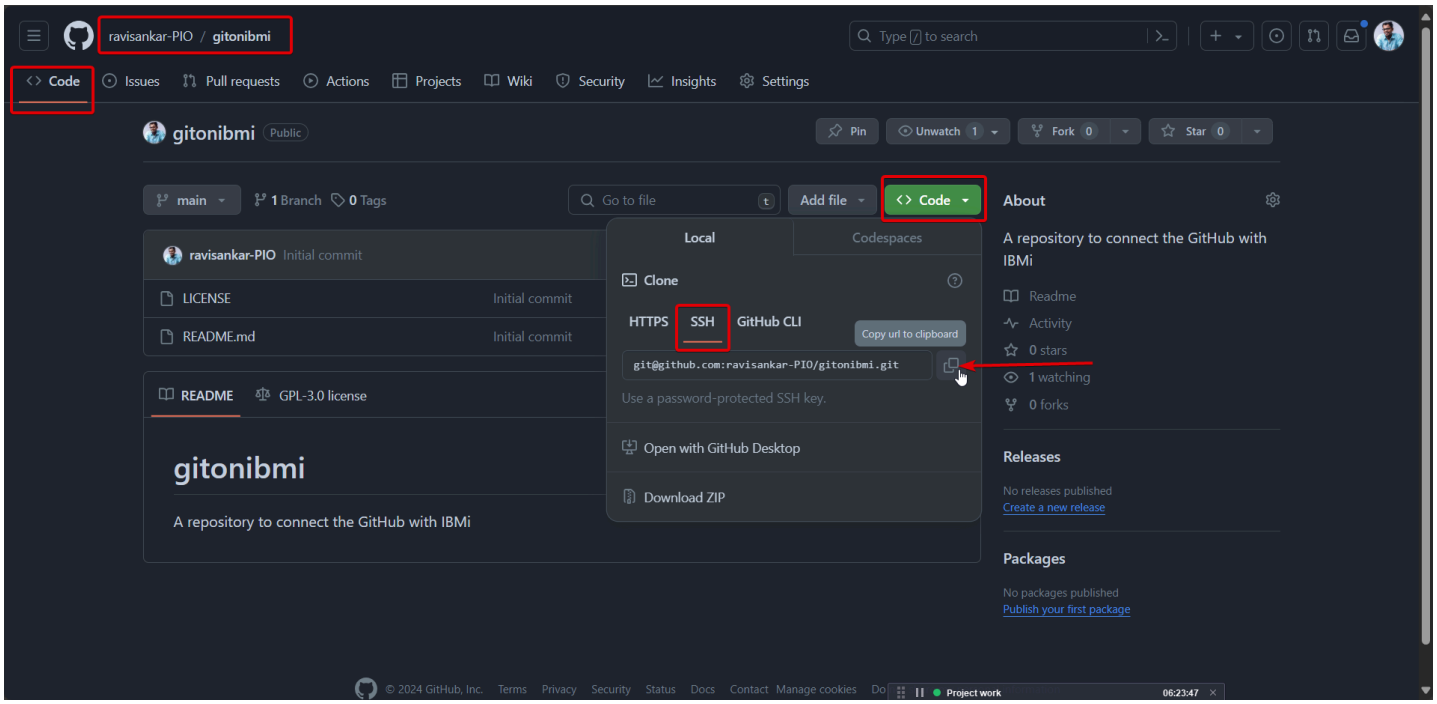
- Enter the repository name, some meaningful description, set it public, add a README.md file and click create repository.



- Nice, we have our own GitHub repository now.



- Click on the green `<> Code` button, click on `SSH` and copy the URL



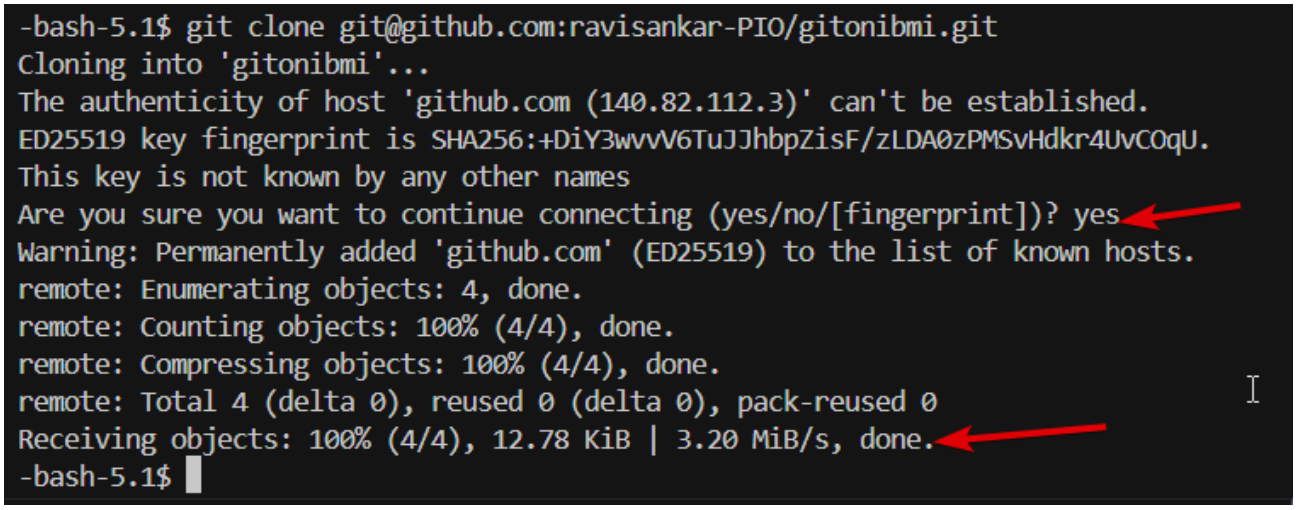
- FYI: This is the command that I just copied
  `git@github.com:ravisankar-PIO/gitonibmi.git`

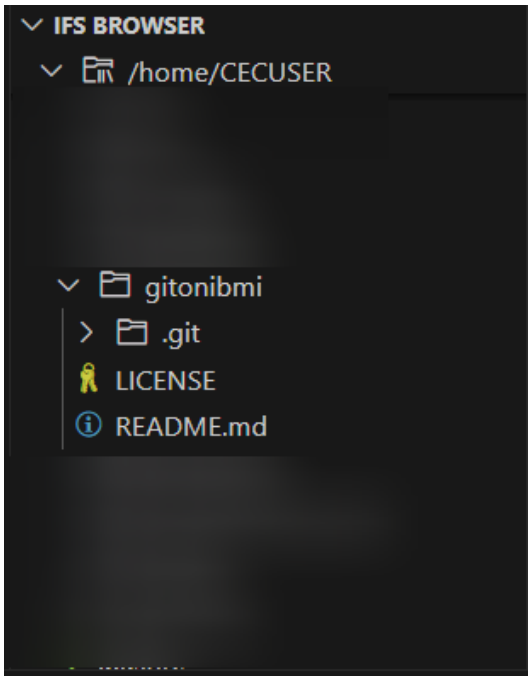
Clone the GitHub Repository to your IBMi

- Go to the PASE Terminal in VS Code and enter

  ```
  git clone git@github.com:ravisankar-PIO/gitonibmi.git
  ```

  Enter `yes` if it asks for anything about Fingerprint and Keys. Now we have successfully cloned the GitHub Repository to our IBMi's IFS folder.





Create a simple sqlrpgle program

- Let's create an SQLRPGLE program which inserts a record into some file for every time it is called.
- Go to the PASE Terminal in VS Code and enter below command to *navigate to our repository folder*

```
cd gitonibmi
```

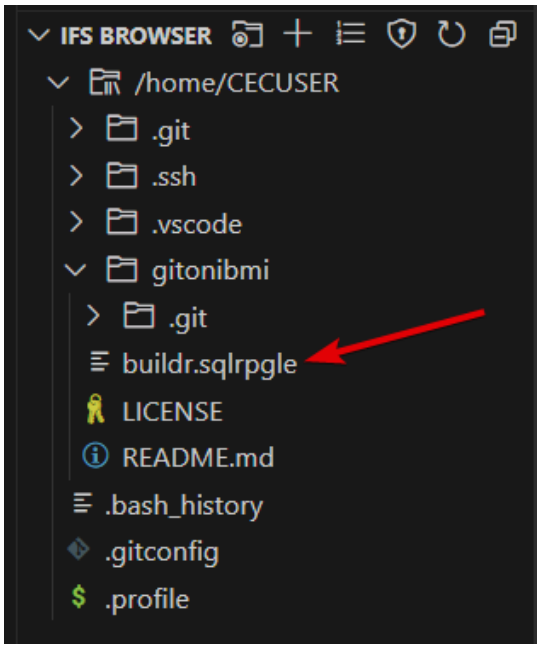
- *Now initialize the git repository*

```
git init
```

- *And create a new SQLRPGLE program*

```
touch buildr.sqlrpgle
```

- Once created, open the same file in your VS Code editor via the IFS Browser.



- Copy paste the below code into the `buildr.sqlrpgle` file and save it.

```
**free
dcl-s count int(10);
dcl-s note varchar(50);

exec sql SELECT COUNT(*) INTO :count FROM ravi.buildpf;
count += 1;
note = 'Build# ' + %char(count);
exec sql INSERT INTO ravi.buildpf (note) VALUES (:note);

*inlr = *on;
```

Commit the program

- Once you saved the sqlrpgle program, head over to the PASE Terminal and enter the below commands one by one. Read below for explanation

```
git add buildr.sqlrpgle
git commit -m "added build sqlrpgle program"
git push
```

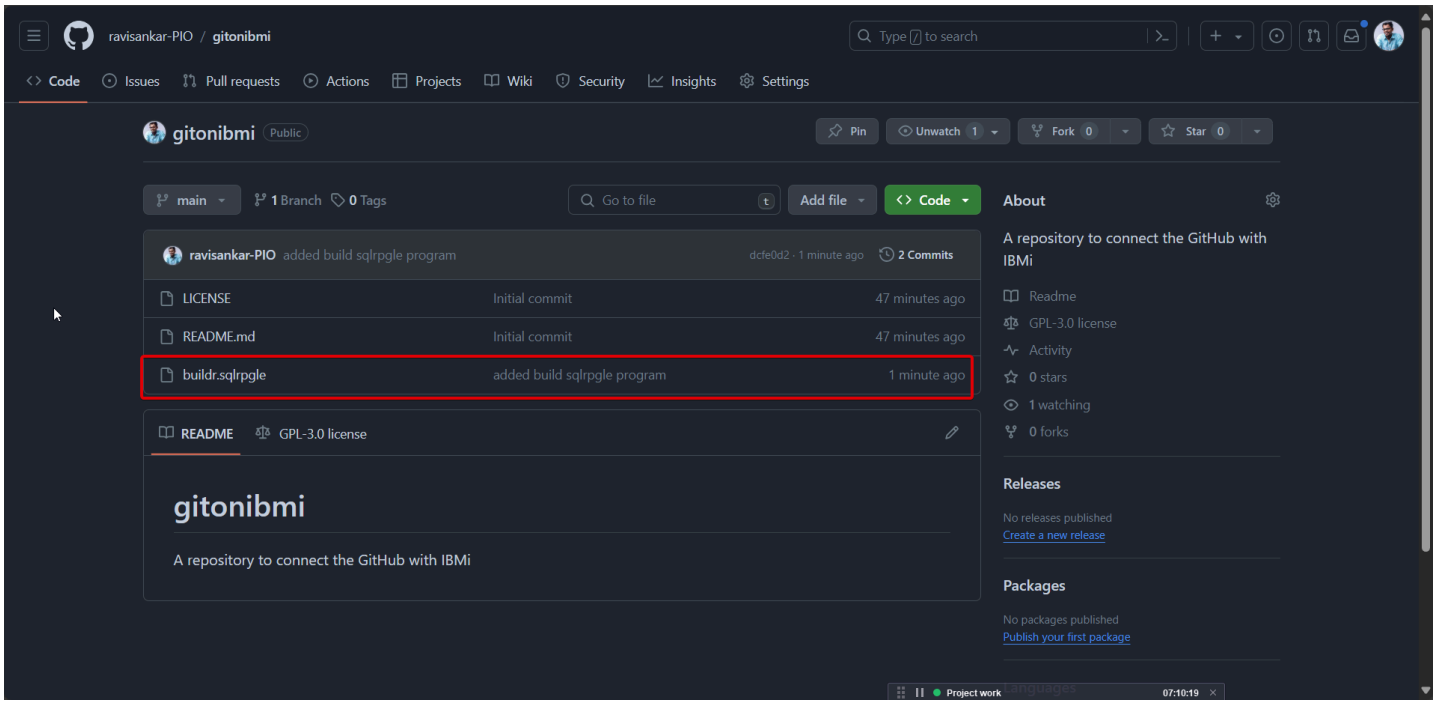
> *Explanation*
>
> - `git add` => we're telling the GIT that we're adding a new file in our repository. So that it will watch for that file for any changes.
> - `git commit` => we're telling the GIT that we want to commit our changes for the previously added/edited files. The commit message describes the purpose of this change. Imagine you're doing a code change, so doing a `git commit` means you're 100% sure that the code change is right and it can be pushed to the QA/Prod. Once the commit is done, git will take a snapshot of the entire repository and you can access this snapshot anytime (like a time machine).

> - `git push` => we're telling the GIT to push the changes to the remote repository (GitHub). So far the changes are done only to the local git (i.e. dev library). Once this is done, the GitHub should reflect the updated sources. This is like promoting the sources to the QA/Prod.
> - *Tip: between every command, you can check the status by issuing 'git status' command*

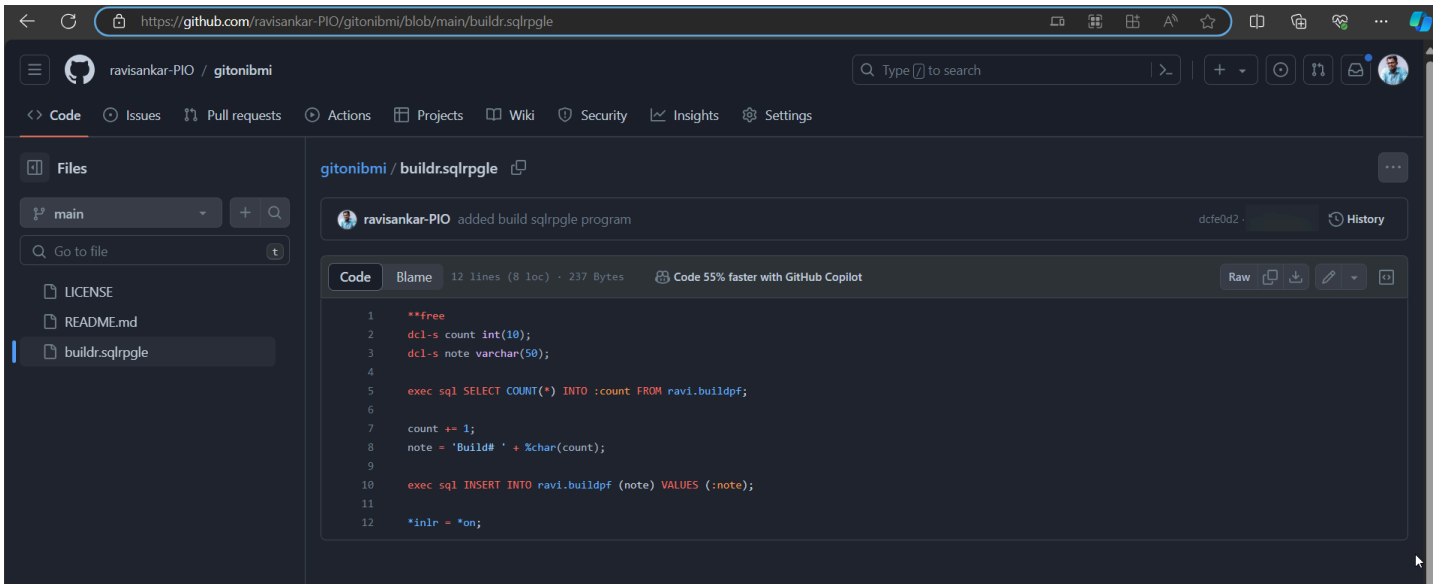
- Once the changes are pushed, you should see a message like something below

```
-bash-5.1$ git commit -m "added build sqlrpgle program"
[main dcfe0d2] added build sqlrpgle program
 1 file changed, 12 insertions(+)
 create mode 100644 buildr.sqlrpgle
-bash-5.1$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 32 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 501 bytes | 501.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:ravisankar-PIO/gitonibmi.git
   7de2c29..dcfe0d2  main -> main
-bash-5.1$
```

- Head over to the GitHub repository to check if the changes are updated there.



Congratulations! You have successfully modernized the IBM i development to GitHub.



Jenkins

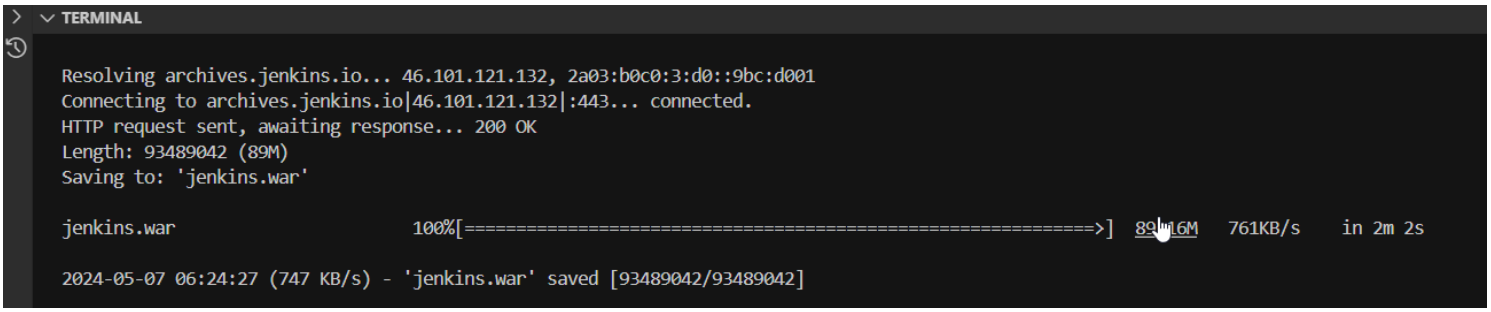
Setup Jenkins

Jenkins is a java based application for setting up CI/CD pipeline. We will setup Jenkins in our IBMi and serve it from the IBMi's IP address itself.

Download the Jenkins install file.

Run the below commands in your PASE terminal to download the `jenkins.war` via `wget` command.

```
cd ~
wget http://mirrors.jenkins.io/war-stable/latest/jenkins.war
```



> *Jenkins is downloaded successfully!*

Start Jenkins

Launching the Jenkins app is nothing but launching the `jenkins.war` file via a JAVA command with correct parameters. It can be started via multiple methods.
*(Notice that I am using the **port# 9095**)*

- **Method-1:** Use Process Management tool like Service Commander. **(preferred)**
  Jump to this section to view how to start the application via Service-Commander.

- **Method-2:** Start directly in an interactive SSH sesion
  Enter the below command in your PASE terminal.

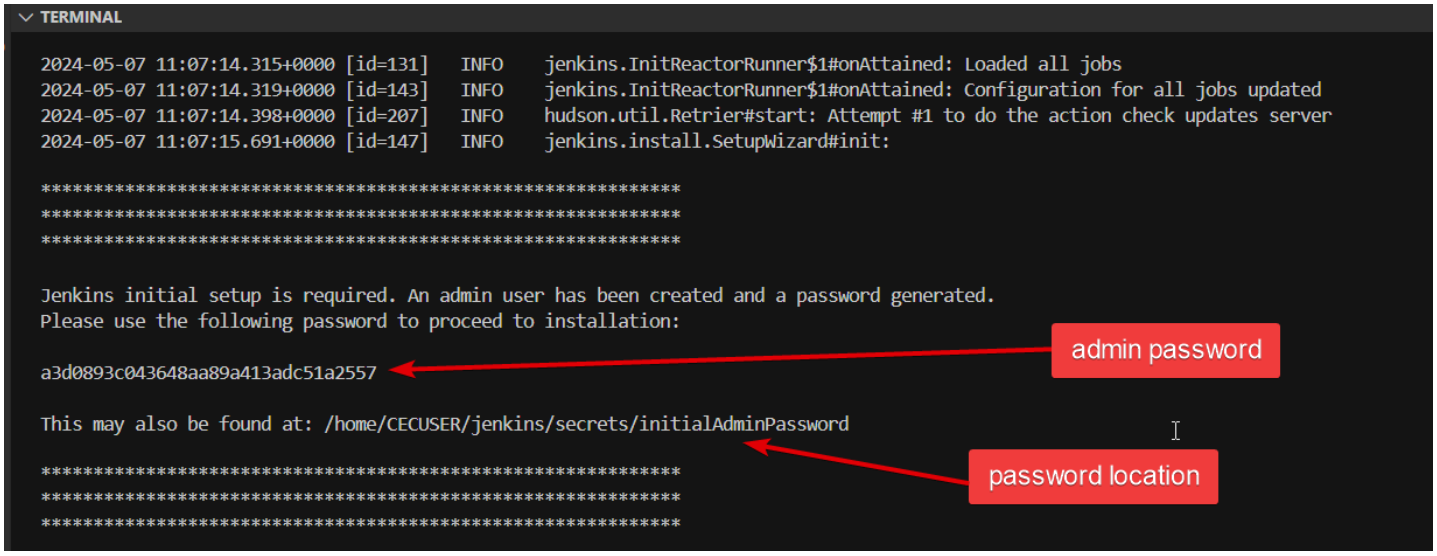
  ```
  java -jar /home/CECUSER/jenkins.war --httpPort=9095
  ```

  Note: You need to keep the PASE Terminal session alive for Method-2

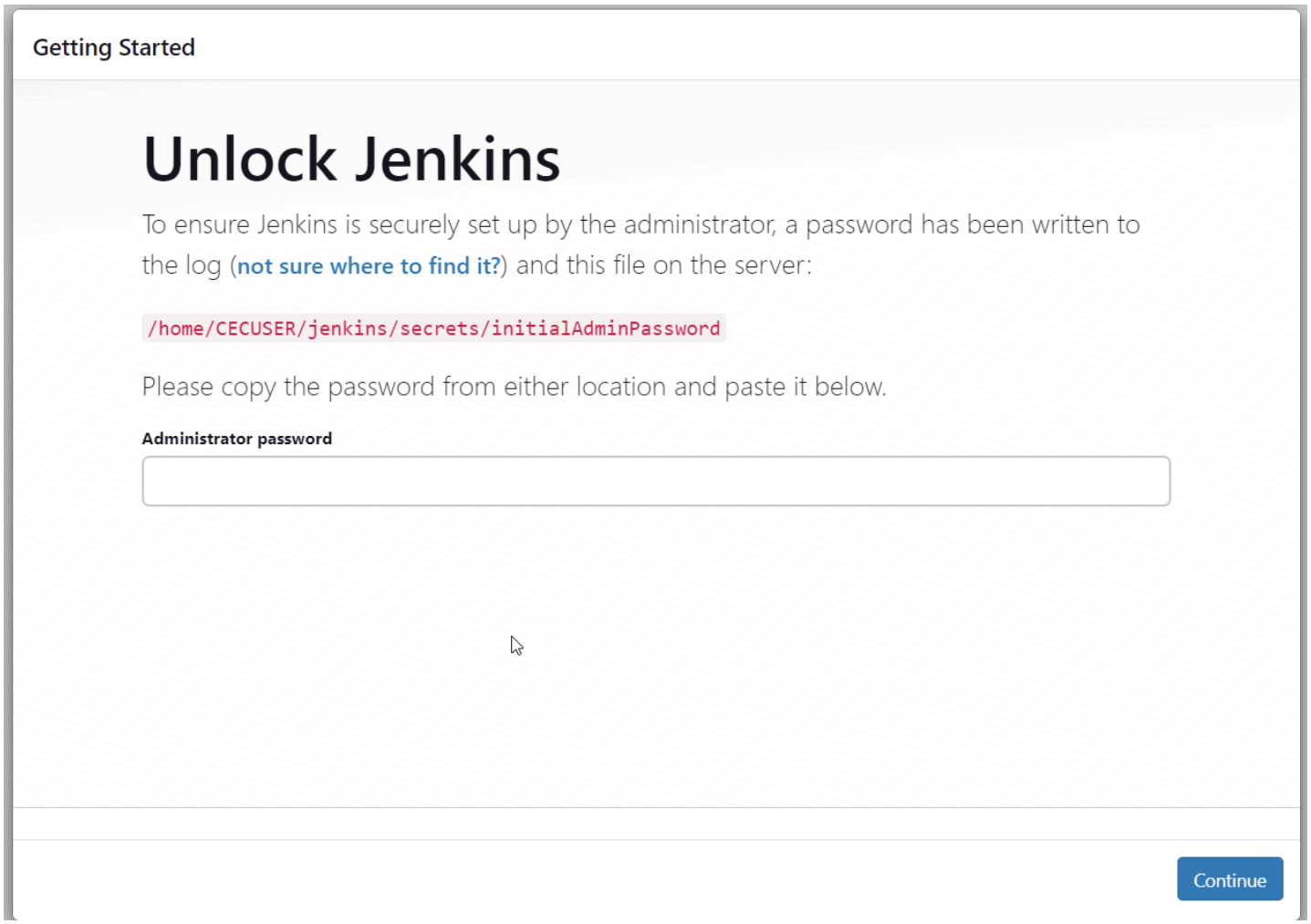
Initial Configuration

- If all worked correctly, then a default admin password will be stored on the location
  `/jenkins/secrets/InitialAdminPassword`

- Open the file `initialAdminPassword` and copy the contents of that file to your clipboard.



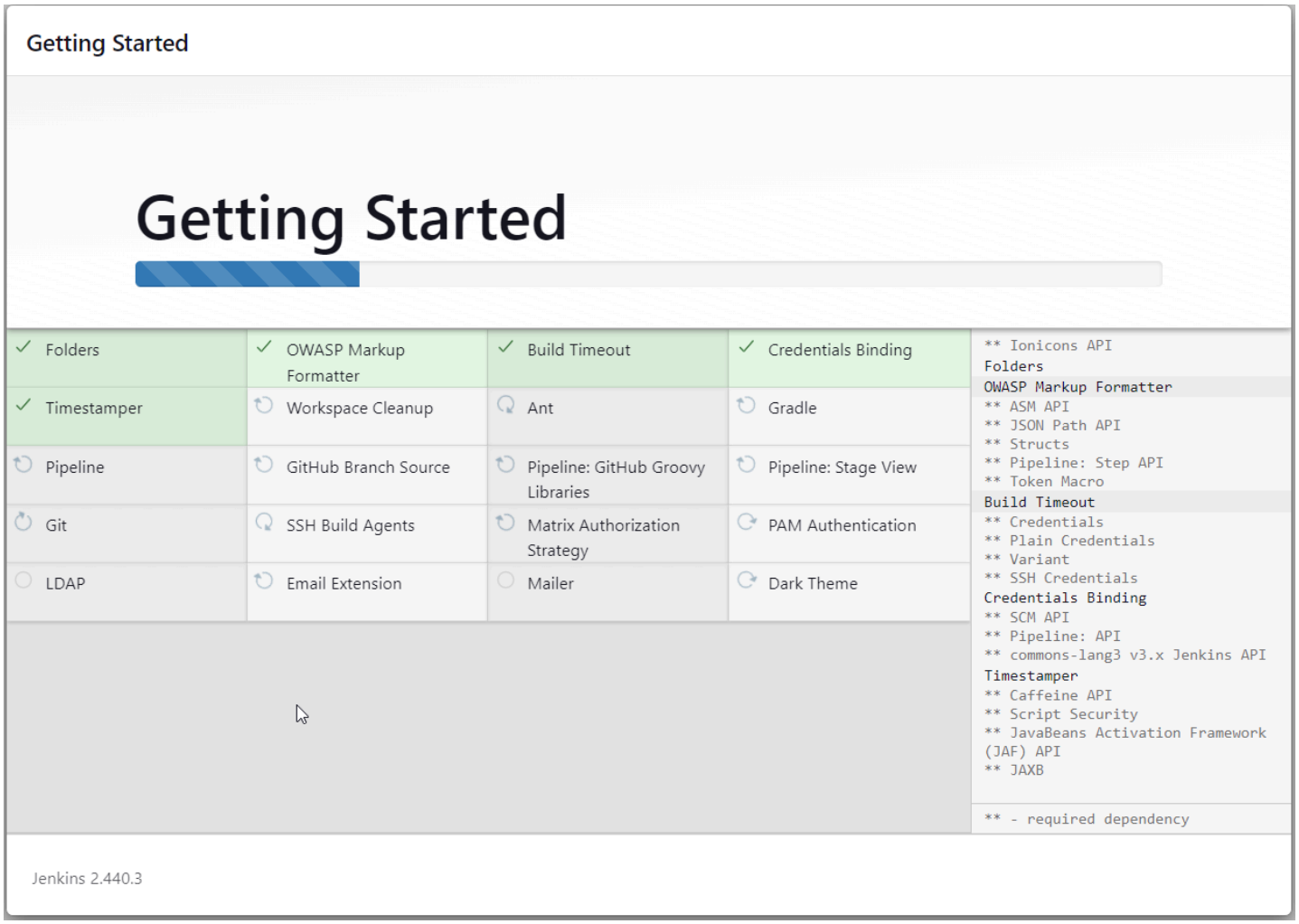
- **Jenkins initial setup in browser**
  Head over to the browser and type in the IP address of the IBMi followed by the port# that we defined earlier. In my case, it is `http://129.40.94.17:9095/` . Paste the admin password that we just copied a while ago to unlock Jenkins.



Remember to select "Install suggested plugins"

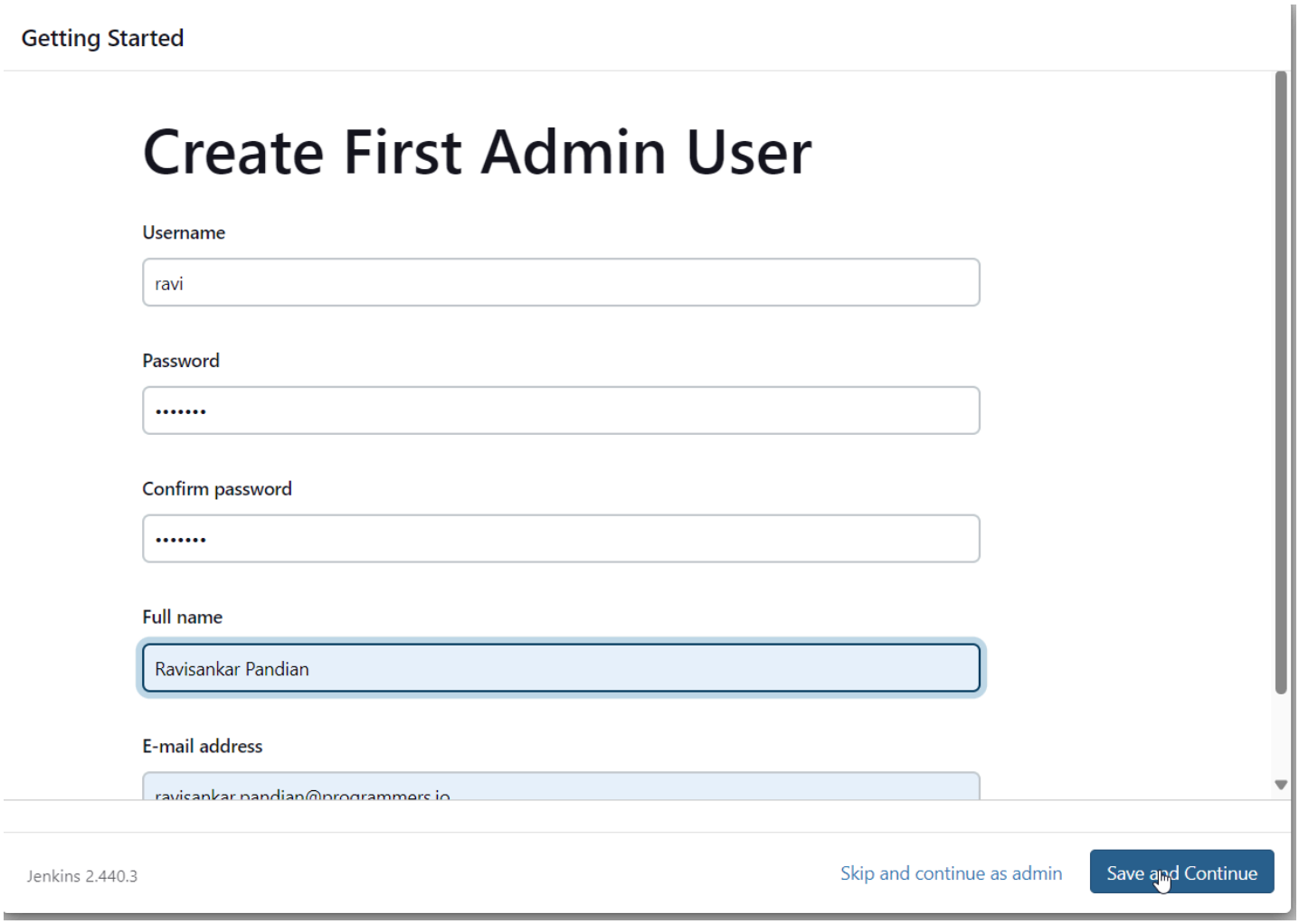
(***Note***: *It will take some time to load the next screen. Don't click more than once, as it might end up in error)*

plugins are currently loaded

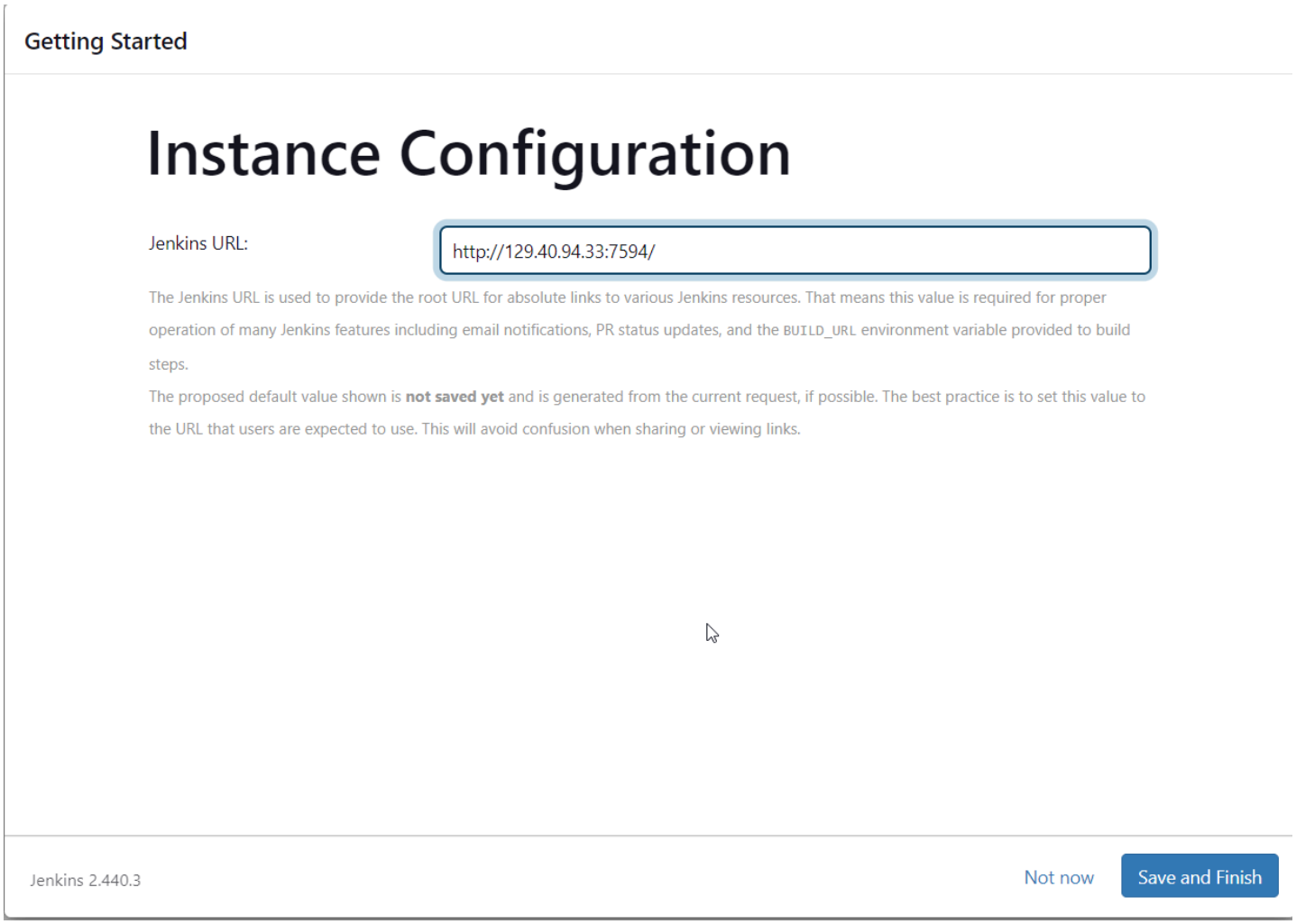


Let's create an Admin user which will be used to login to the Jenkins app from now on.

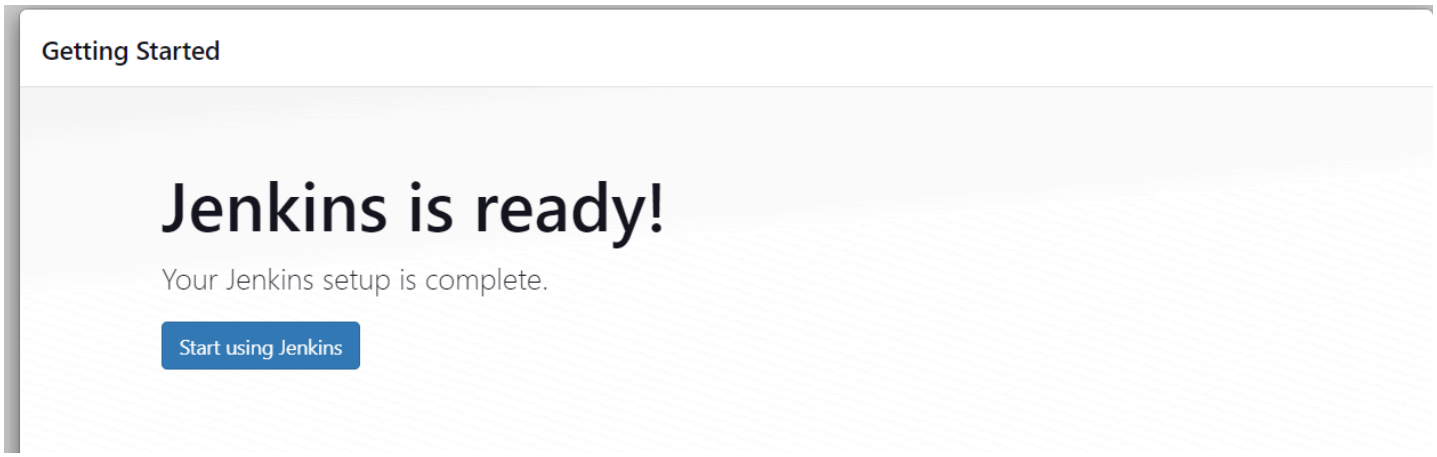
```
UserName: ravi
Password: welcome
Email: ravisankar.pandian@programmers.io
```



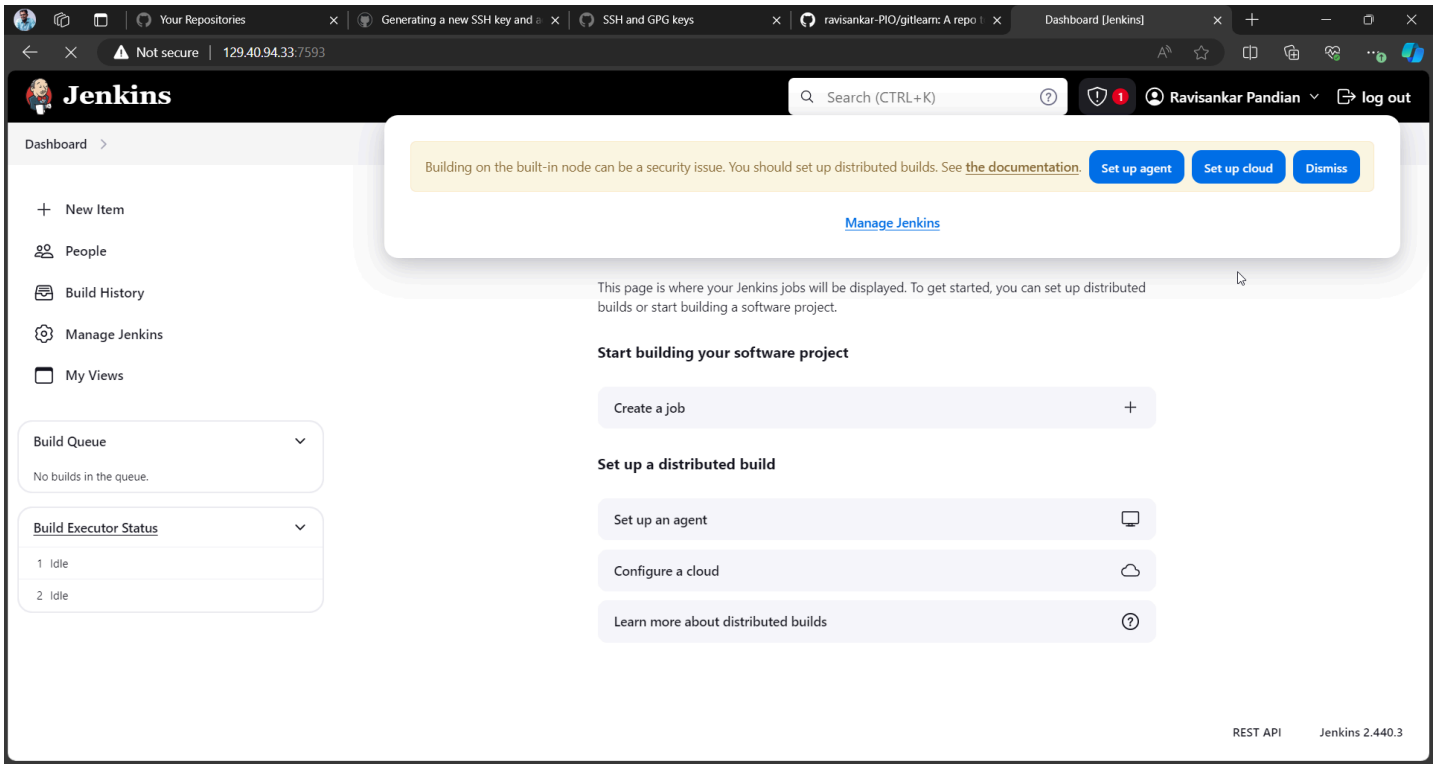
click on save and finish to complete the setup



Nice! we can start using the Jenkins now

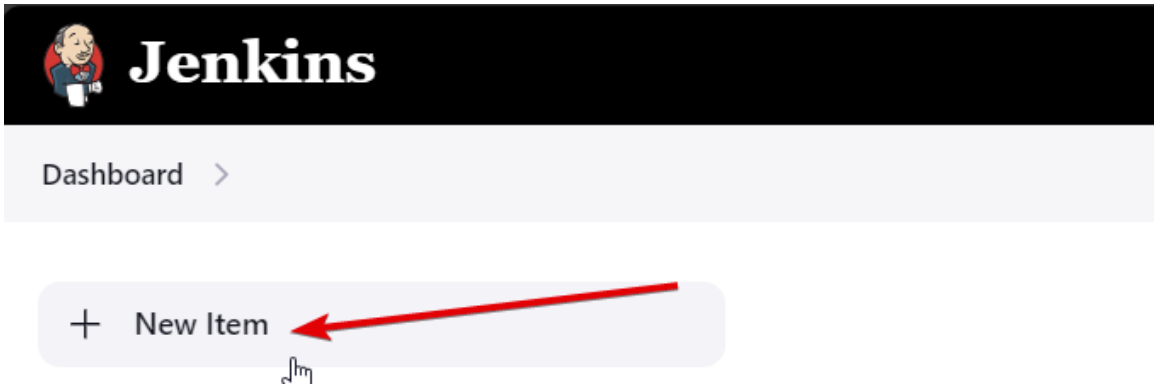


If you see a notification at the top as given below, It is advised to have a separate node for building the code. But we will click dismiss for now and continue with our work

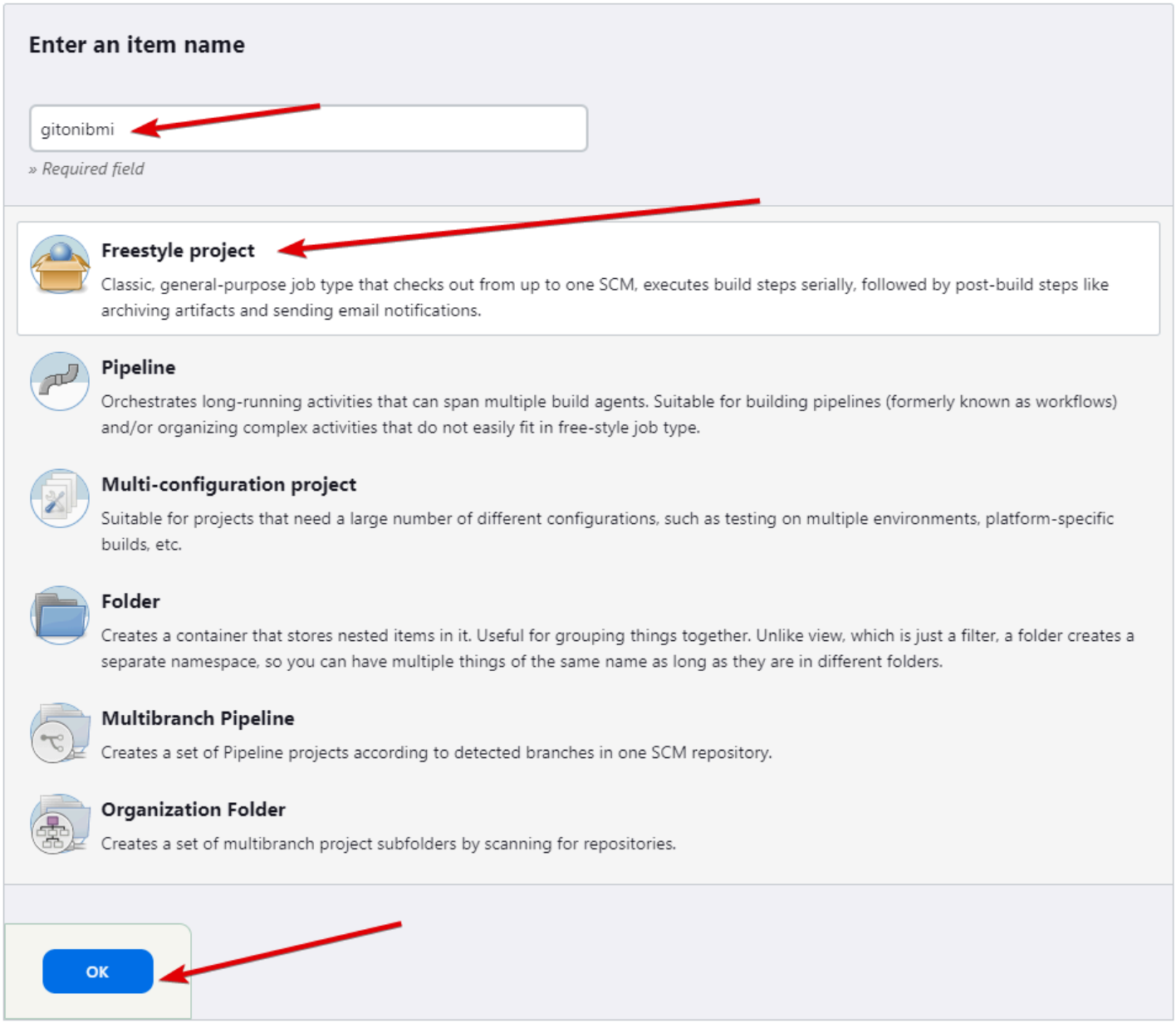


Create a New Job in Jenkins

- On the Dashboard, click on New Item

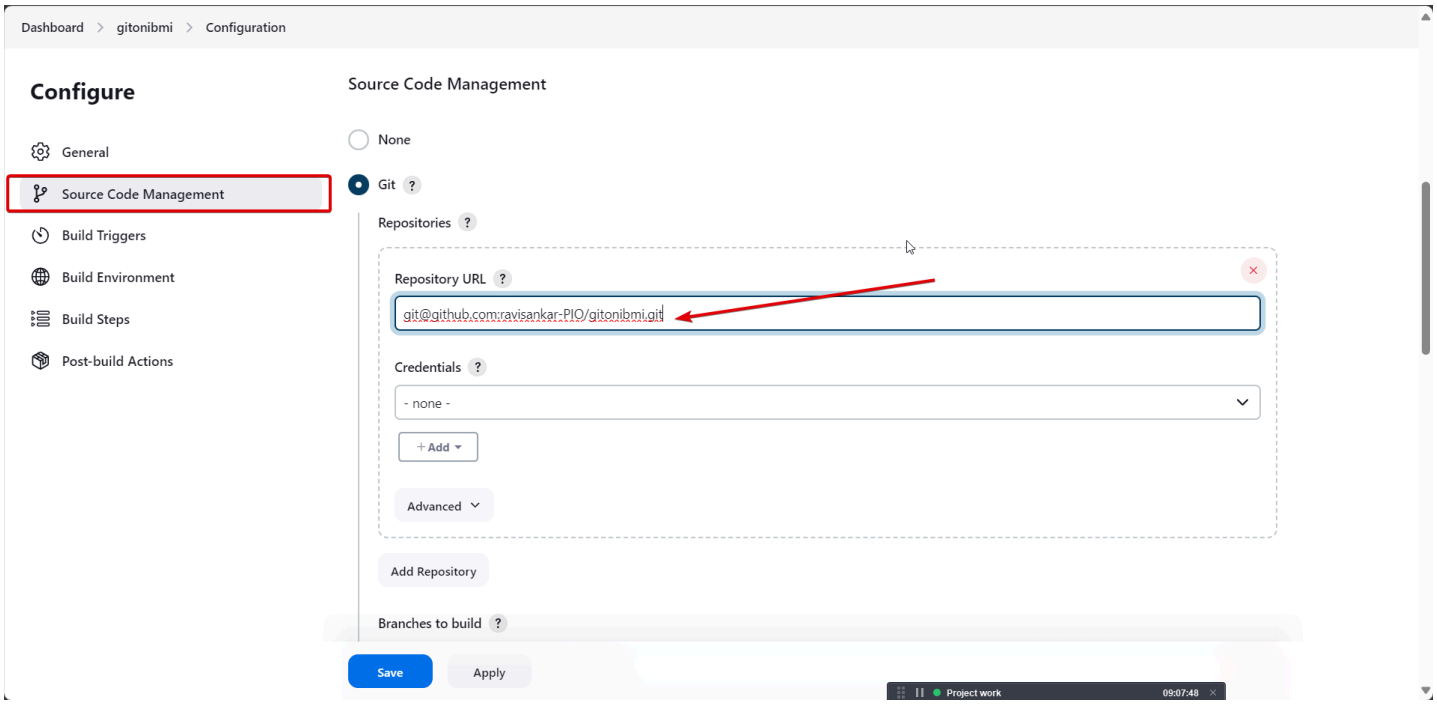


- Enter the Job Name as `gitonibmi` , select `freestyle project` and click OK.

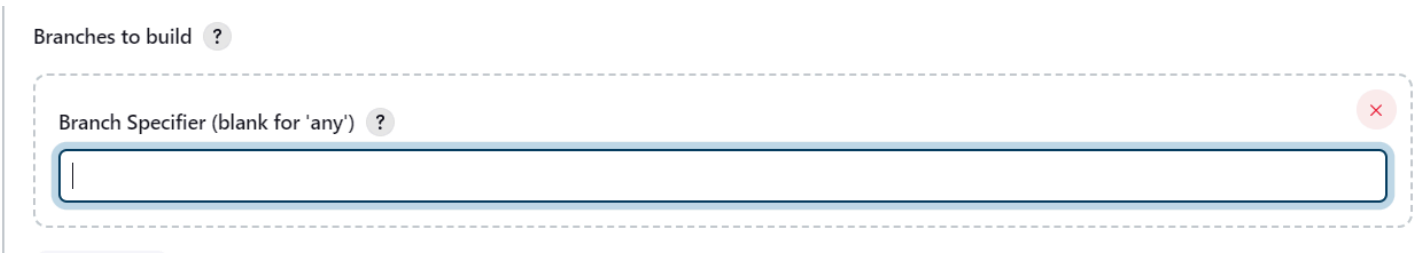


Configure the Job to connect to GitHub

- In the Job Configuration Page, Click on the `Source Code Management` on the left, and select `GIT` .
- Remember the Repository URL that we copied a while ago from our GitHub? We need to paste that over here.

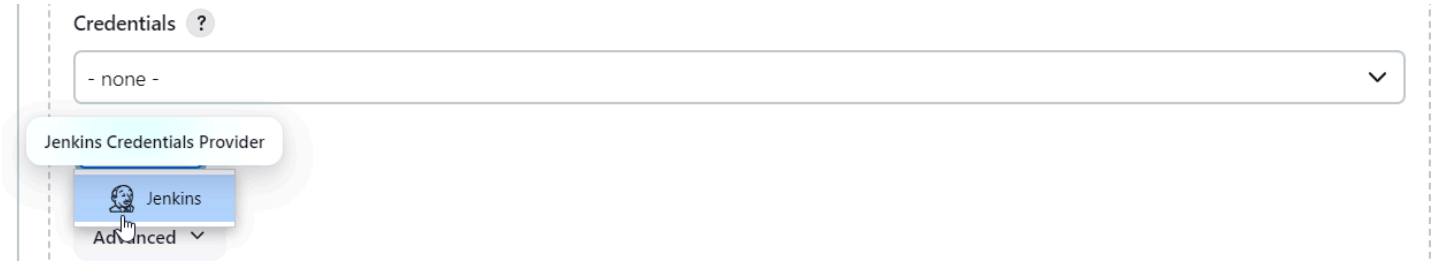


- Make sure to clear out the branches to build field

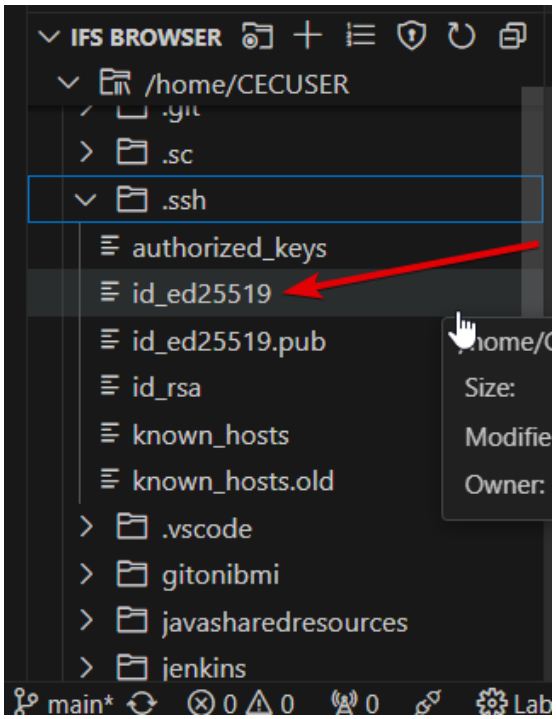


Add a credential

- Notice the `+Add` button under credential. Click on it to add a credential to connect to the GitHub securely.
  Note: Sometimes, the button will load slowly. So don't press multiple times



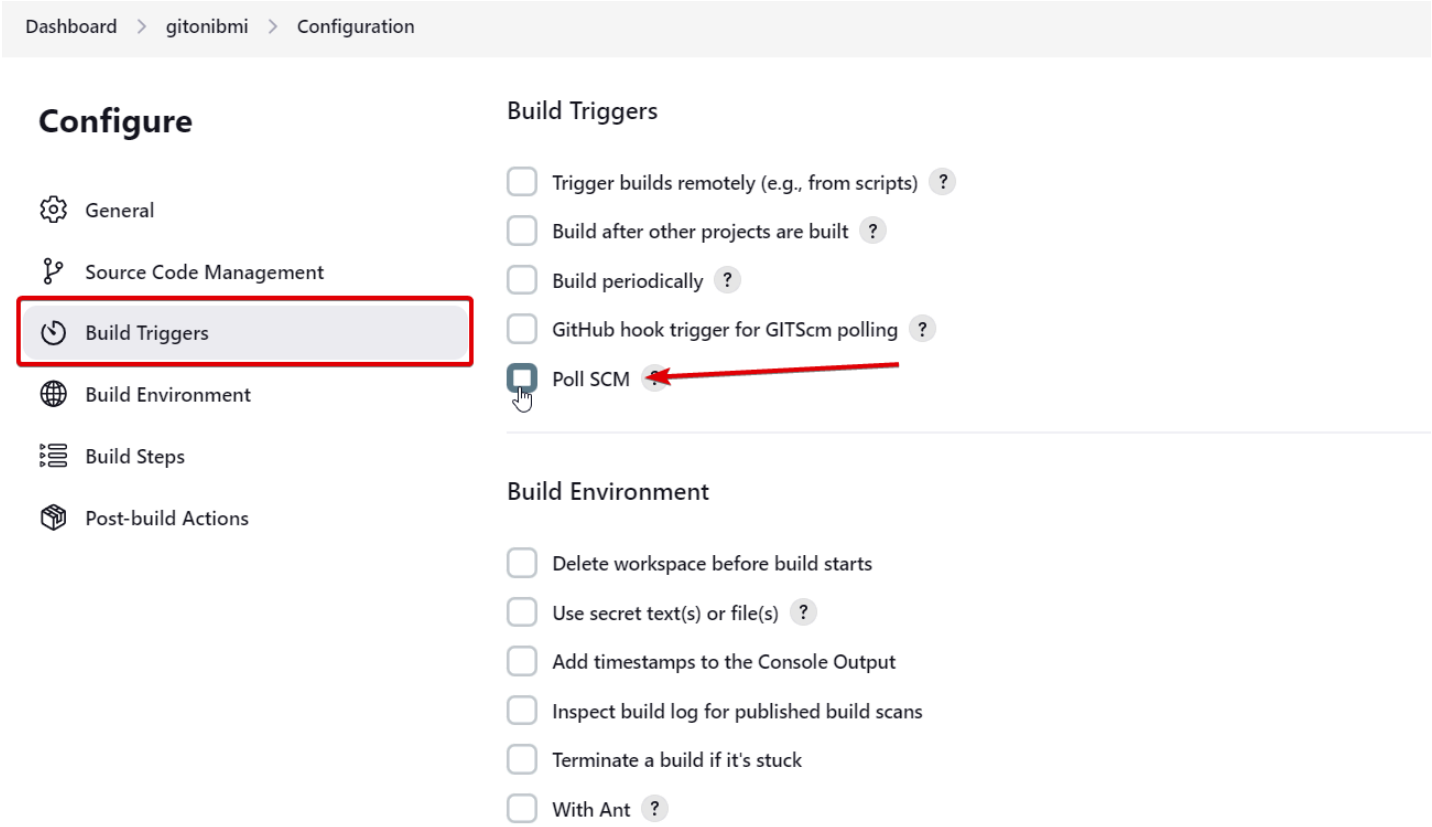
- Enter the details as below
  - Domain - Global (unrestricted)
  - Kind - SSH Username with Private key
  - Scope - Global
  - ID - anything you like
  - Private key - Select `enter directly` => click add => Open the private key from your ssh folder as below => copy the entire content => finally paste it on Jenkins window.



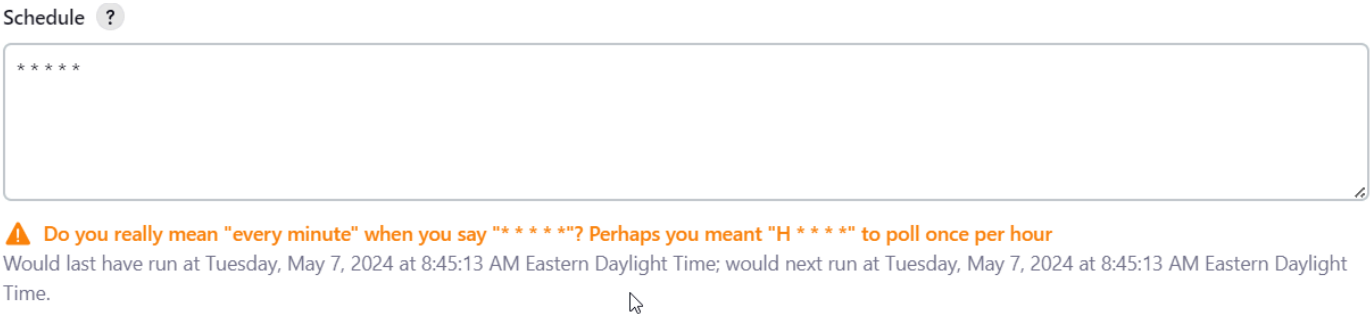
- Then click add
- Then, click on the credentials drop down, and select the one that has your user name.

Add a Build Trigger

- Click on `Build Triggers` on the left menu and check `Poll SCM`

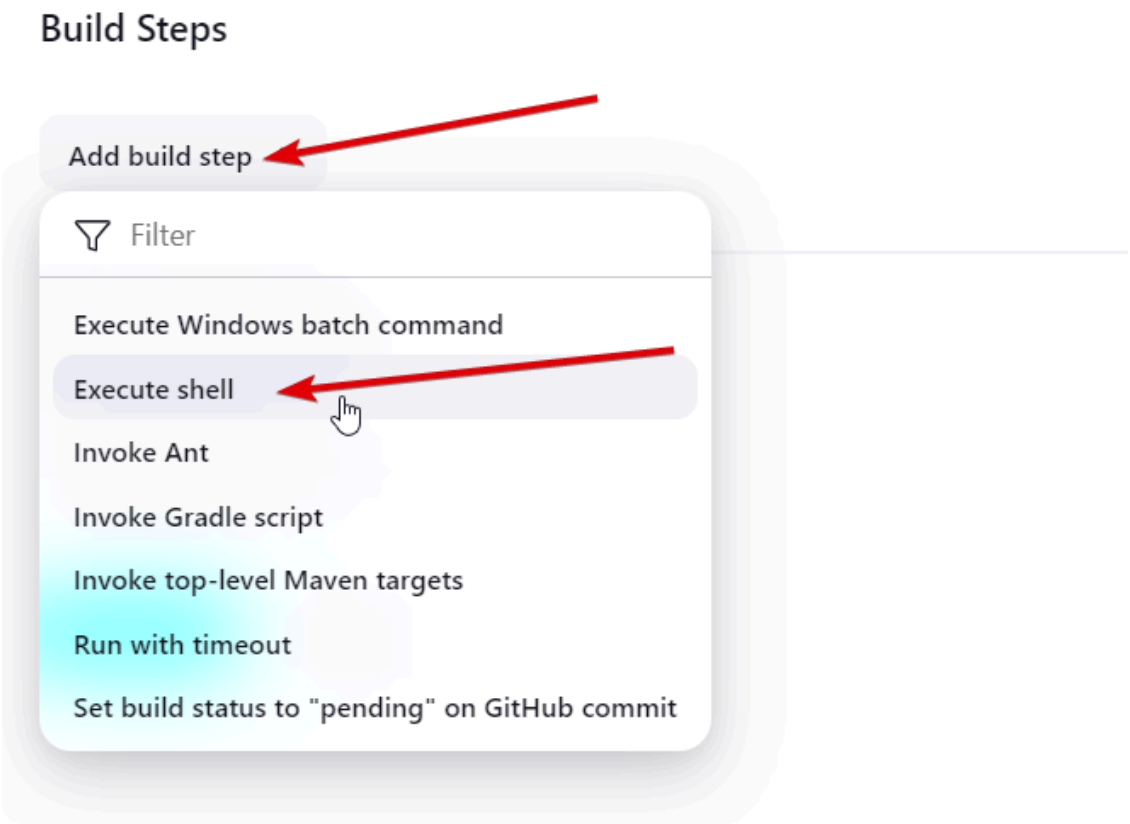


- Enter the value as `* * * * *` (5 asteriks with spaces inbetween). This is a cron scheduler which will look for any changes made in our repository for every minute and then execute the build steps.



Add Build Steps

- Scroll down to find `Build Steps` and click on it, then select `Execute Shell`

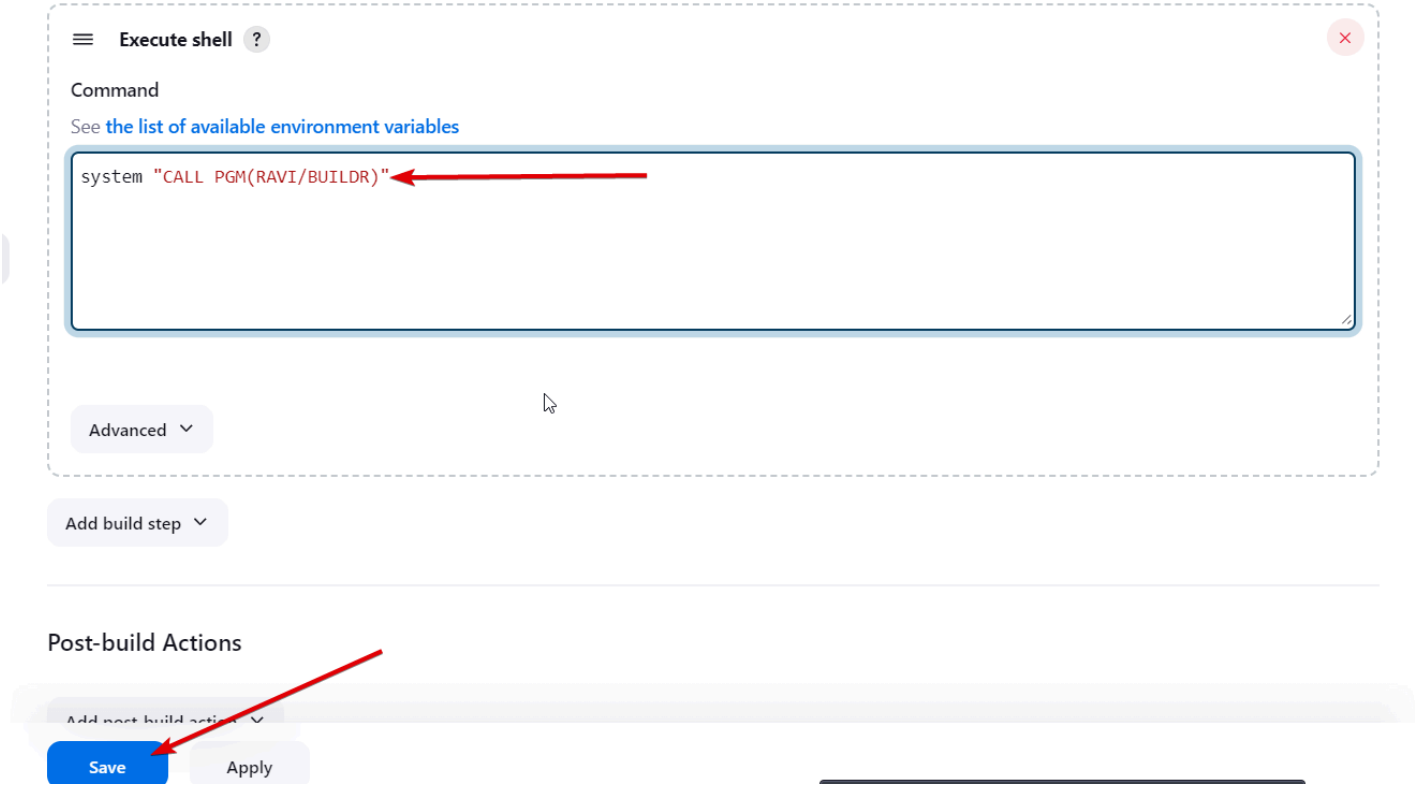


- Enter the below command to call our newly created BUILDR program.

```
system "CALL PGM(RAVI/BUILDR)"
```

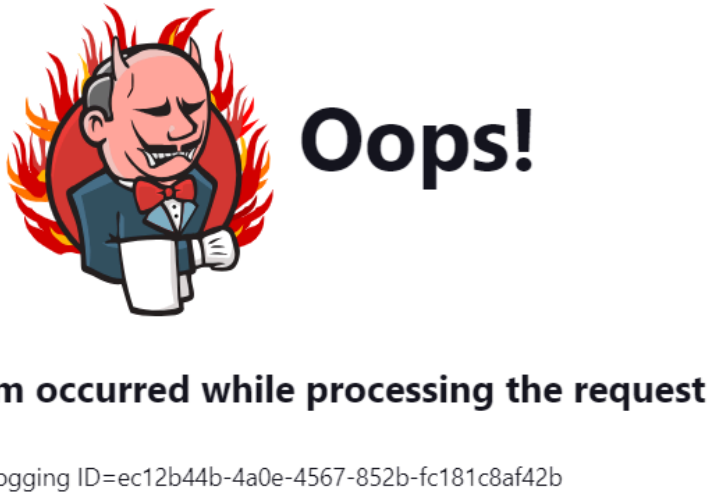
This means whenever the GitHub repository is committed (i.e. updated), we will call a program called `buildr`

- That's it! We will save the Job Configuration now.



Error!

Are you seeing this error below and wondering what you did wrong? Don't worry as I am on your side too! The same error happened to me as well. This could be a bug in Jenkins, but let's just proceed forward.

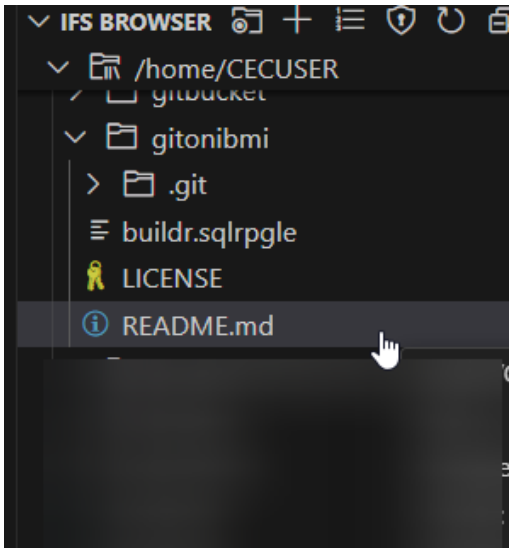


What we have done so far?

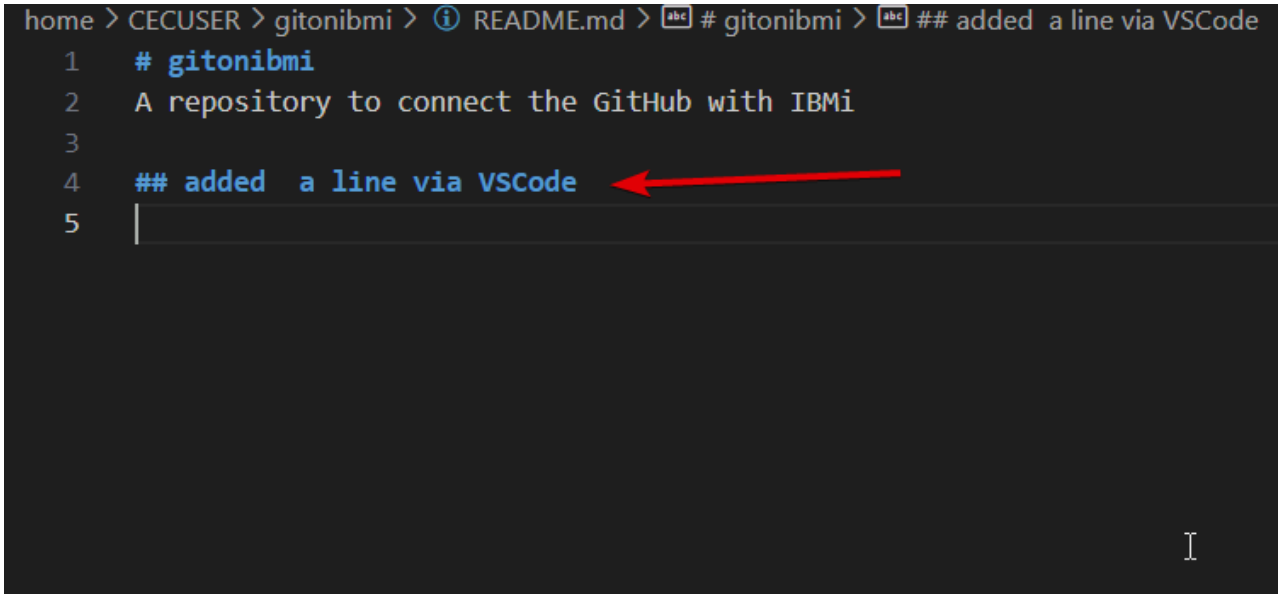
- We have added a Jenkins job to listen on the GitHub repository for any change.
- If any change(commit) occurs, then the Jenkins will execute the shell commands that we have entered in the 'build steps'.
- The shell command is nothing but to call an SQLRPGLE program which will insert a record to the buildpf with the build#.

Let's try it in action!

- For the sake of simplicity, let's update the `README.md` file via VS Code and push the changes to the GitHub repo. We will expect the Jenkins to pickup the change and process the job.
- Open the `README.md` file



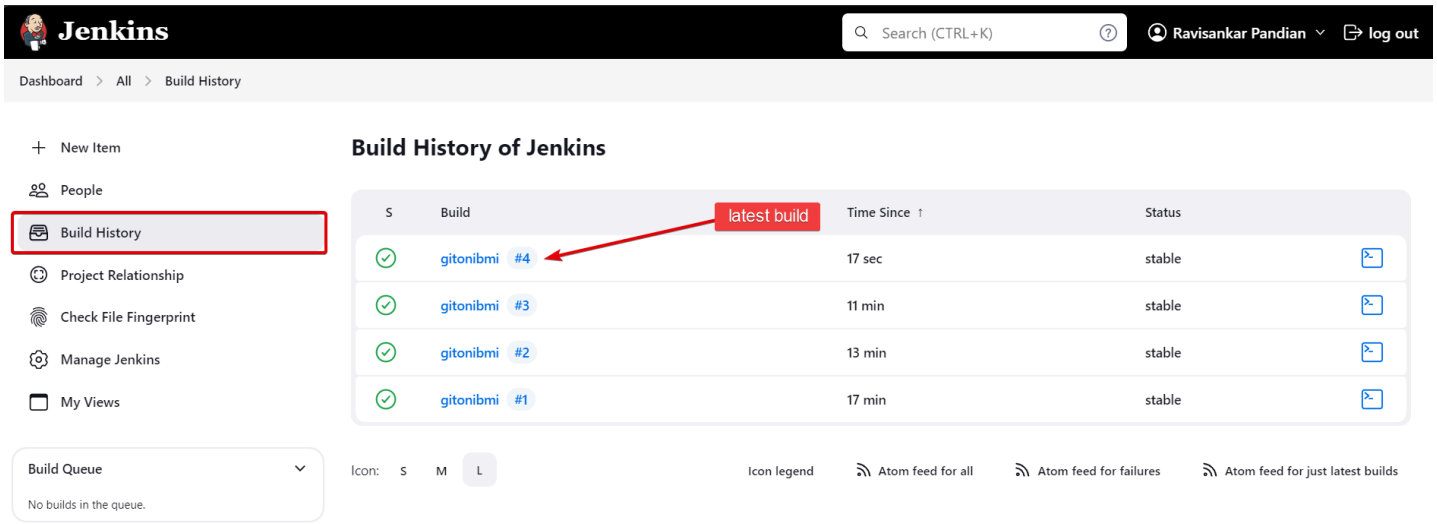
- Add another line and save it.



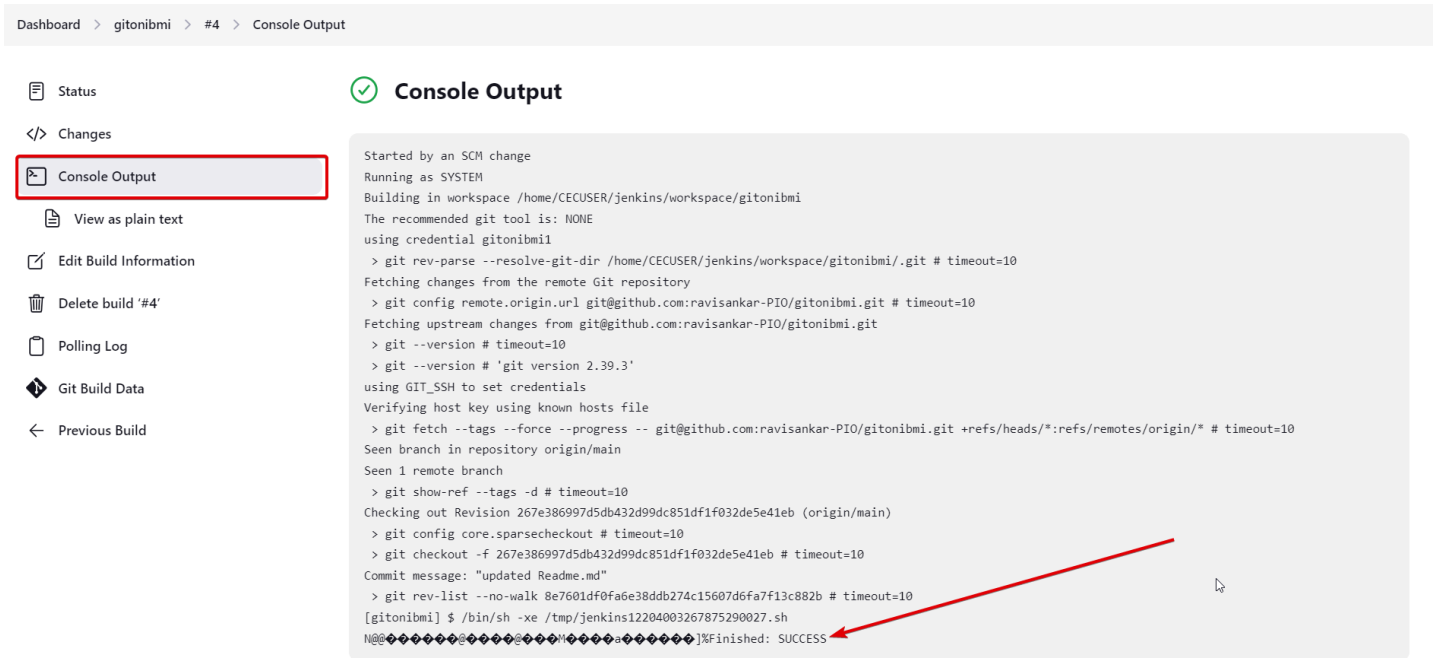
- Head over to the PASE terminal and enter commands below.

```
git add *
git commit -m "updated Readme.md"
git push
```

- Wait for a few seconds and check your build history in Jenkins. Click on the build#. In my case it is #4, but for you it should probably be #1.



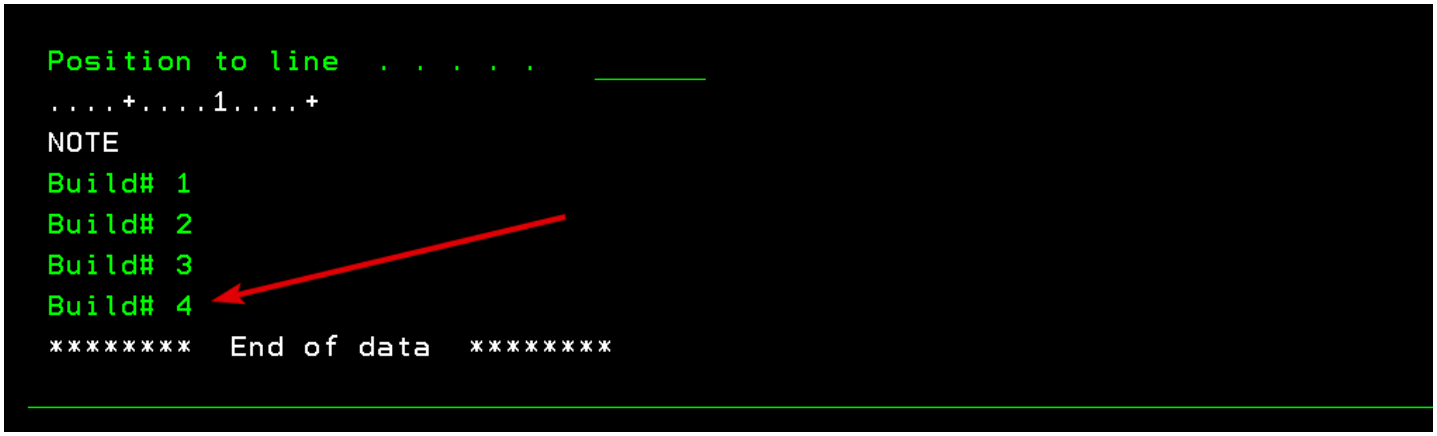
- Let's check the console ouput as well.



It is success!

- Let's check our physical file if the shell script got executed correctly.

```
select * from ravi/buildpf
```



> Nice! It is working

Install GitBucket on IBMi

Gitbucket is a JAVA based SCM tool which can be run on IBMi. This is an Open Source alternative to GitHub.

Download the GitBucket installation JAR file first
Run the below commands in your PASE terminal

```
cd ~
wget https://github.com/gitbucket/gitbucket/releases/download/4.41.0/gitbucket.war
```

Start GitBucket using the Java Command
Launching the GitBucket app is nothing but opening the `gitbucket.war` file via a JAVA command with correct parameters. It can be started via multiple methods.

*(Notice that I am using the **port# 8085**)*

- **Method-1:** Start directly in an interactive SSH sesion
  Head over to the green screen and issue the command below.

  ```
  java -jar /home/CECUSER/gitbucket.war --port=8085
  ```

- **Method-2: (Preferred)** Use Service Commander to start the tool.

Footnotes/References

- Jenkins
  - Getting started with Jenkins.
  - Create Jenkins pipeline
  - Setup Jenkins on IBMi
- GitBucket

    - A VCS that can be hosted within IBMi. Click here to learn more.
- Source Orbit ()
    - Defintion of Source Orbit - a dependency management system
    - A blog post by Liam Barry that tells about the SO - SourceOrbit.
- IBMi - CI
    - A built in CI tool within IBMi. See it in action here
    - Learn more about IBMi-CI
- PM2
    - A Node.JS package used to manage the applications.
    - Refer this, this and this link to automate Jenkins using PM2.
- Service Commander
    - A command line tool for managing various services and applications running on IBMi.
- Others
    - Click here to view about the third party Repos for IBMI
    - RPG-GIT-BOOK - This is an excellent starting point for moving to GIT
- Ansible
    - Ansible for i's github repo
    - Ansible's documentation for IBMi
    - Ansible's core documentation
    - A blog post about Automating your IBM i tasks with Ansible.

Further Research

GitLab

Since Gitlab is an open source alternative for GitHub, I wanted to check if it can be installed on the IBMi. But it didn't helped.

First, I **Installed the dependencies for GitLab**

```
yum install -y curl policycoreutils-python openssh-server perl
```

https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.rpm.sh

```
2024-05-03 08:02:53 (3.54 MB/s) - 'script.rpm.sh' saved [7153/7153]

-bash-5.1$ ls
script.rpm.sh
-bash-5.1$ ./script.rpm.sh
grep: can't open /etc/issue
Unfortunately, your operating system distribution and version are not supported by this script.

You can override the OS detection by setting os= and dist= prior to running this script.
You can find a list of supported OSes and distributions on our website: https://packages.gitlab.com/docs#os_distro_version

For example, to force CentOS 6: os=el dist=6 ./script.sh

Please email support@packagecloud.io and let us know if you run into any issues.
-bash-5.1$
```

- Read more about Gitlab
  - dependencies for GitLab
  - CICD Script to connect GitLab with IBMI
  - Gitlab installation methods
  - Gitlab installation steps

PM2

PM2 is a process management app (built on Node.js) which is like an enhanced Task Manager for IBMi. It will be used to autostart, keep the node.js & java based apps persistent.

Install PM2

- Kill the Jenkins app first if already launched by entering `Ctrl+c` two times on the PASE terminal where the Jenkins is currently launched.
- Install NodeJS => `yum install nodejs14`
- Then install PM2 => `npm install pm2@latest -g`
- Add the location of the nodejs's binary to the path
  - Open the .profile file and add a new location to include in path
    alt text
  - Save and close the `.profile` file. Disconnect from IBMi and connect again (for the changes to take effect).

Configure PM2

- Create a new file called `jen.json` in your home folder. This will be used to start the Jenkins app.
  `touch jen.json`
- Paste this content into that file as below.

```
{
"apps":[
{
    "name":"jenkins",
    "cwd":".",
    "script":"/QOpenSys/usr/bin/java",
    "args":[
        "-jar",
        "jenkins.war",
        "--httpPort=9095"
    ],
    "exec_interpreter":"",
    "exec_mode":"fork"
 },
 {
  "name":"gitbucket",
  "cwd":".",
  "script":"/QOpenSys/usr/bin/java",
  "args":[
      "-jar",
      "/home/CECUSER/gitbucket/gitbucket.war",
      "--port=8085"
  ],
  "exec_interpreter":"",
  "exec_mode":"fork"
}
]
}
```

- Run this command to start the Jenkins => `pm2 start jen.json`
- Once started, we can view the started apps by => `pm2 ls`

```
-bash-5.2$ pm2 ls
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
1	gitbucket	default	N/A	fork	85709	21s	0	online	0%	0b	cecuser	disabled
0	jenkins	default	N/A	fork	85708	21s	0	online	0%	0b	cecuser	disabled

- For some reason, I am **unable to end the application via PM2**. So I searched for a better alternative and found the **IBMi native service commander**

Service-Commander

- Install service commander using the below command

```
yum install service-commander -y
```

Startup Jenkins

- Now we will do the one time setup to include the Jenkins app in our service commander utility.
- Now the actual command to start the jenkins app from the PASE Terminal is

  `java -jar /home/CECUSER/jenkins.war --httpPort=9095`
- We will just add `scinit` at the front to start the setup. So run the below command

```
scinit java -jar /home/CECUSER/jenkins.war --httpPort=9095
```

- Service commander is an intelligent tool that will ask us series of questions in order to be able to do the intial setup.

Question	Answer
Would you like this service to be available to all users? [n] (we don't want other users to start this application)	n
Short Name (this will be the name to start the application. So choose wisely)	jenkins
Friendly Name (a short description about the app)	Jenkins for IBMi
Start app in the current directory (/home/CECUSER)? [y] (yes, we want to start the app in the current directory)	y
Which ports does your app run on? (Enter the port# that the app is using)	9095
App to be run under a unique Job Name? (No, so we will leave it to blanks)	
Submit to batch? (No, we will run the app from within PASE environment)	n
Environment Variables? (Since we have already setup the path variables, we will leave it as blanks)	
What Other Environment Variables? (Nothing here, just hit enter again)	
What Other groups would this app be a part of? (Nothing here, just hit enter again)	
What Other services would this app be a part of? (Nothing here, just hit enter again)	

- If all worked correctly, then you should see the below output

```
-------------------------------------------------------------------
jenkins (Jenkins on IBMi)


Defined in: /home/CECUSER/.sc/services/jenkins.yml

Working Directory: /home/CECUSER

Startup Command: java -jar '/home/CECUSER/jenkins.war' '--httpPort=9095'
Startup Wait Time (s): 60

Shutdown Wait Time (s): 45

Check-alive conditions: PORT:9095
Batch Mode: <not running in batch>

Inherits environment variables?: true
Custom environment variables:
    PATH=/QOpenSys/pkgs/bin:/QOpenSys/usr/bin:/usr/ccs/bin:/QOpenSys/usr/bin/X11:/usr/sbin:.:/usr/bin
    JAVA_HOME=/QOpenSys/QIBM/ProdData/JavaVM/jdk17/64bit
-------------------------------------------------------------------
```

- Now let us start the application by entering

```
sc start jenkins
```

```
-bash-5.2$ sc start jenkins
Performing operation 'START' on service 'jenkins'
Service 'Jenkins on IBMi' successfully started
For details, see log file at: /home/CECUSER/.sc/logs/2024-05-07-22.43.11.jenkins.log
```

Startup GitBucket

- Run the below command

```
scinit java -jar /home/CECUSER/GitBucket.war --port=8085
```

- and answer the questions below

Question	Answer
Would you like this service to be available to all users? [n] (we don't want other users to start this application)	n
Short Name (this will be the name to start the application. So choose wisely)	gitbucket
Friendly Name (a short description about the app)	GitBucket for IBMi
Start app in the current directory (/home/CECUSER)? [y] (yes, we want to start the app in the current directory)	y

Question	Answer
Which ports does your app run on? (Enter the port# that the app is using)	9095
App to be run under a unique Job Name? (No, so we will leave it to blanks)	
Submit to batch? (No, we will run the app from within PASE environment)	n
Environment Variables? (Since we have already setup the path variables, we will leave it as blanks)	
What Other Environment Variables? (Nothing here, just hit enter again)	
What Other groups would this app be a part of? (Nothing here, just hit enter again)	
What Other services would this app be a part of? (Nothing here, just hit enter again)	

- If all worked correctly, then you should see the below output

- Now let us start the application by entering
  `sc start GitBucket`

Gmake or BOB?

Gmake or BOB can be used to compile the objects directly from the IFS path, without the need to have source members as a middle man.

Following this article for BOB

1. Install IBMi Repos => `yum install ibmi-repos`
2. Install BOB => `yum install bob`
   faced an error
3. update yum packages => `yum update` and try again to install bob
4. Bob installed successfully
5. Let's test the build command by cloning a git repo.
   i. Create a library to save the build into
   `system "CRTLIB LIB(BOBTEST) TEXT('Better Object Builder test project')"`

ii. Clone the git repo (that already contains sample sources)

```
git clone https://github.com/ibm/bob-recursive-example
```

iii. Change directory

```
cd bob-recursive-example
```

iv. Set the environment variable to point the library that we just created

```
export lib1=BOBTEST
```

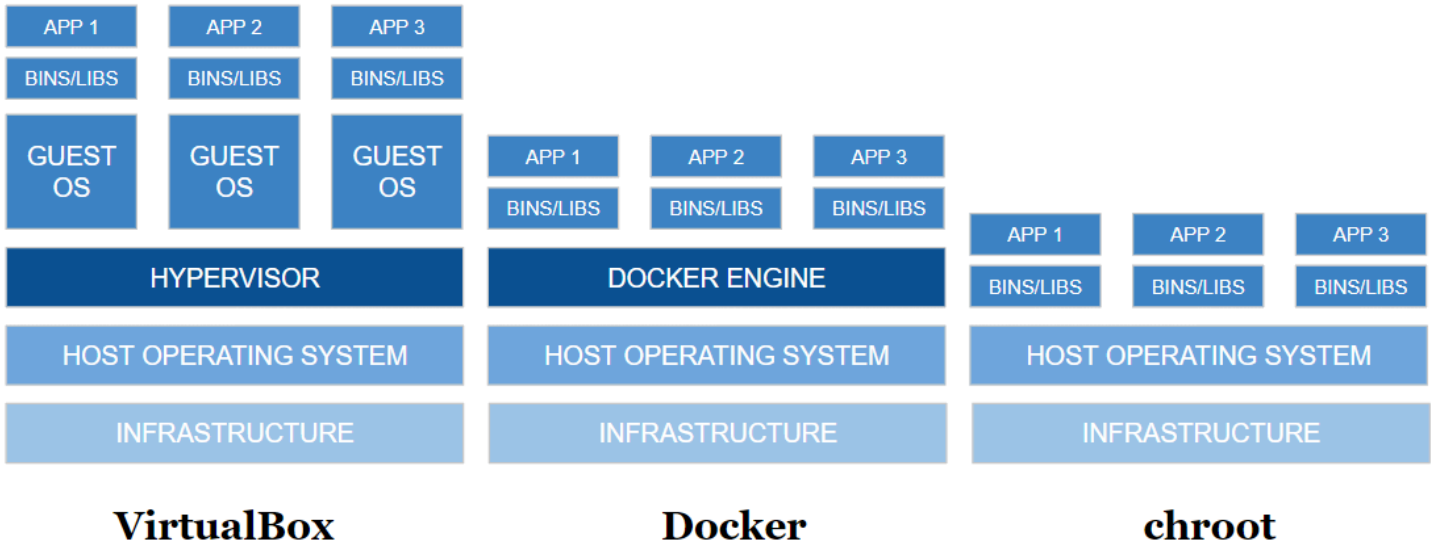
v. Run the build using,

```
makei build
```

```
TERMINAL

=== Creating program [CHS] from modules [CHS] and service programs []
crtpgm pgm(BOBTEST/CHS) module(CHS) bndsrvpgm(*NONE) ACTGRP() USRPRF(*USER) TGTRLS() AUT() DETAIL(*EXTENDED) OPTION(*EVENTF ) STGMDL(*SNGLVL) TEXT(' ')
✓ CHS.PGM was created successfully!

=== Creating RPG module [DEU.RPGLE]
crtrpgmod module(BOBTEST/DEU) srcstmf('/home/CECUSER/bob-recursive-example/globalization/DEU/DEU.RPGLE') AUT() DBGVIEW(*ALL) OPTIMIZE() OPTION(*EVENTF) OUTPUT(*PRINT) TEXT(' ') TGTC
CSID(273) TGTRLS() INCDIR('includes' 'QPROTOSRC') DEFINE()
✓ DEU.MODULE was created successfully!

=== Creating program [DEU] from modules [DEU] and service programs []
crtpgm pgm(BOBTEST/DEU) module(DEU) bndsrvpgm(*NONE) ACTGRP() USRPRF(*USER) TGTRLS() AUT() DETAIL(*EXTENDED) OPTION(*EVENTF ) STGMDL(*SNGLVL) TEXT(' ')
✓ DEU.PGM was created successfully!

=== Create Bound RPG Program [HEB] in BOBTEST
CRTBNDRPG srcstmf('/home/CECUSER/bob-recursive-example/globalization/HEB/heb.pgm.rpgle') PGM(BOBTEST/HEB) TGTCCSID(424) DBGVIEW(*ALL) OPTION(*EVENTF) TEXT(' ') INCDIR('includes' 'QP
ROTOSRC')
✓ HEB.PGM was created successfully!

Objects:          0 failed 124 succeed 124 total
Build Completed!
```

Chroot

- Chroot creates IFS containers within IBMi for `/QOpenSys`. It empowers users to have their own root folder.
- Let's say if we have an IBMi server that already runs some OSS software in it that requires certain version of Node.JS or Python to function. We don't want to break/update that version for our DevOps practices. So we can create containers where the entire IBMi OSS environment would be run independently.



> Further information about chroot can be found here
>
> A blog post about Chroot
>
> Another one

Test Cases

- Refer IBMiUnit by Liam
- Refer iUnit by Wim Jongman

Migrate to IFS

A tool to migrate the source members to IFS path

Todos

- Clone the IBMI Company System, and bob recursive repositories for testing the DevOps scenarios - **Done**
- Create a script to install the pre-requisites and installations of DevOps tools - **Done**

Local vs Cloud GIT

Local/On Prem GIT Tools

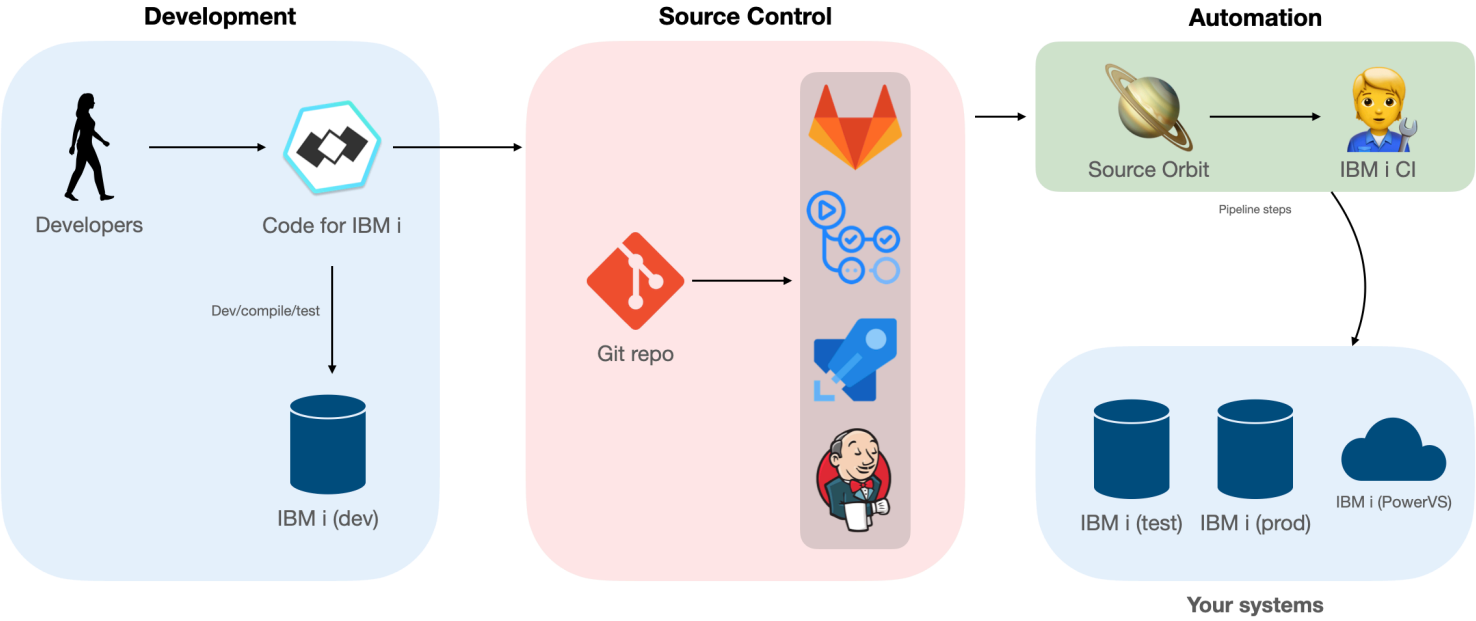
- GitBucket on IBMi
- GitLab on Linux
- Klaus Git Viewer on IBMi

Final Thoughts

- Pick the right tools required for the DevOps Practice.
  - GitHub **(Prop & Cloud)**, GitBucket **(OSS & On-Prem)**, GitLab or BitBucket
  - Jenkins, IBMi-CI ***(new)***, GitLab CICID, GitHub Actions etc., for CI-CD
  - Gmake or BOB for building the code
- Research on Source Orbit ***(new)*** to resolve object dependency conflicts
- Research on the right tool to setup unit test cases.
- Research on the migration of the Sources from Members to IFS.

For a Modernized IBM-i development:

May be a **self hosted GitBucket**, running along with **Jenkins**, that triggers **Source Orbit** for checking object dependencies, use **gmake** to compile the sources, and **IBMi-CI** to deploy the sources to production would be an ideal setup.



Usage of Self-Hosted GitBucket instead of the GitHub

Jenkins

Search (CTRL+K) | Ravisankar | log out

Dashboard > gitbucketjobonssh > #1 > Console Output

- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#1'

Console Output

```
Started by user Ravisankar
Running as SYSTEM
Building in workspace /home/CECUSER/jenkins/workspace/gitbucketjobonssh
The recommended git tool is: NONE
using credential bucketid
Cloning the remote Git repository
Cloning repository http://129.40.94.33:8085/git/root/gitlearn1.git
 > git init /home/CECUSER/jenkins/workspace/gitbucketjobonssh # timeout=10
Fetching upstream changes from http://129.40.94.33:8085/git/root/gitlearn1.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.3'
using GIT_ASKPASS to set credentials
 > git fetch --tags --force --progress -- http://129.40.94.33:8085/git/root/gitlearn1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url http://129.40.94.33:8085/git/root/gitlearn1.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse origin/master^{commit} # timeout=10
ERROR: Couldn't find any revision to build. Verify the repository and branch configuration for this job.
Finished: FAILURE
```

REST API | Jenkins 2.440.3

Jenkins

Search (CTRL+K) | Ravisankar | log out

Dashboard >

- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views

Build Queue

Build Executor Status

1 Idle

2 Idle

Add description

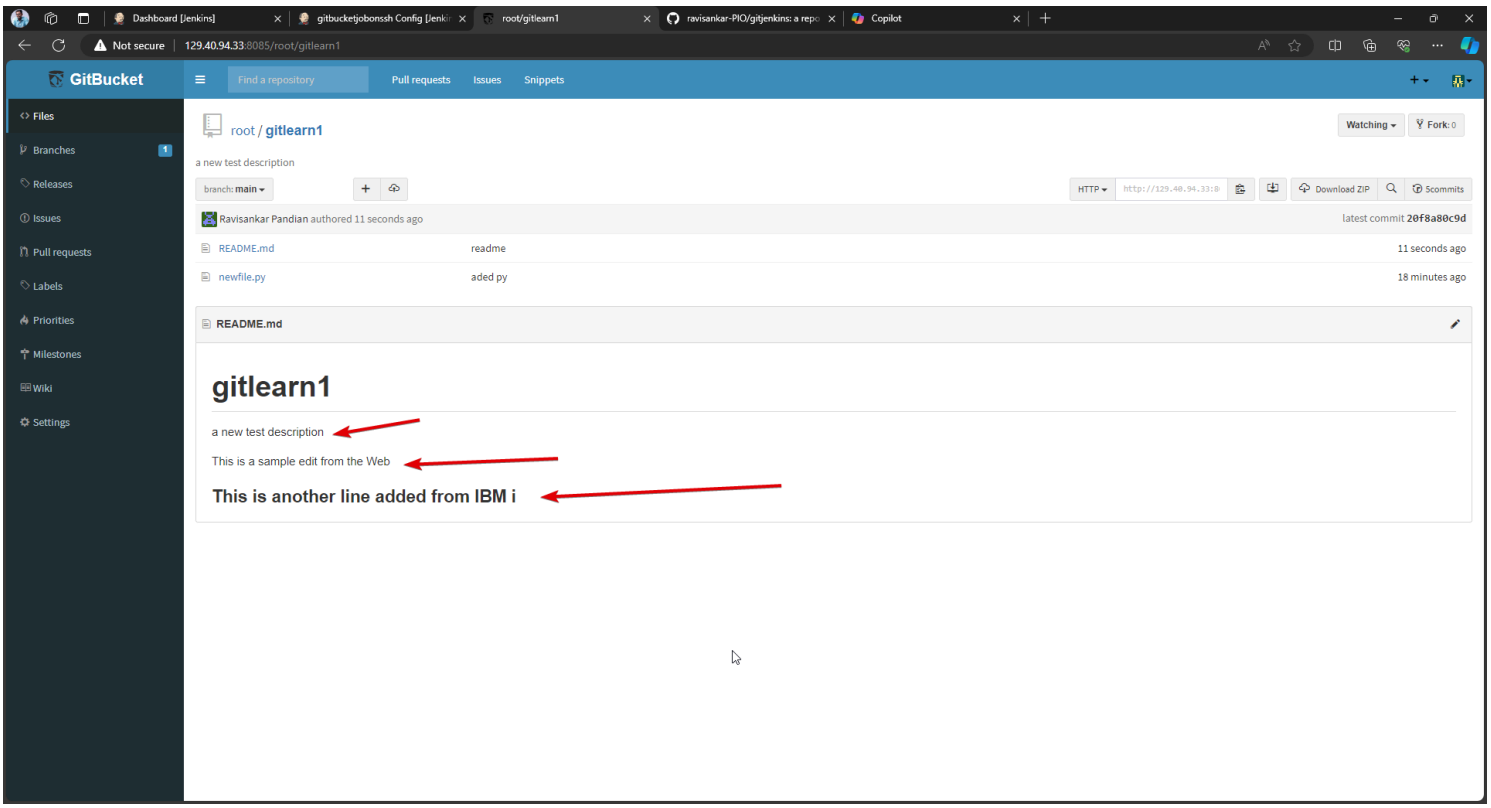
All +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
✗		gitbucketjob	N/A	15 min #1	2.9 sec	▷
✗		gitbucketjobonssh	N/A	2 min 5 sec #1	4.1 sec	▷
✓		gitjenkins3	1 hr 23 min first build	N/A	5.4 sec	▷

Icon: S M L | Icon legend | Atom feed for all | Atom feed for failures | Atom feed for just latest builds

REST API | Jenkins 2.440.3

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 32 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 341 bytes | 341.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Updating references: 100% (1/1)
To http://129.40.94.33:8085/git/root/gitlearn1.git
   a088ae9..39215b2  main -> main
-bash-5.2$ rm * -R
-bash-5.2$ ls
-bash-5.2$ cd ..
-bash-5.2$ rm gitlearn1 -R
-bash-5.2$ git clone ssh://git@129.40.94.33:29418/root/gitlearn1.git
Cloning into 'gitlearn1'...
The authenticity of host '[129.40.94.33]:29418 ([129.40.94.33]:29418)' can't be established.
RSA key fingerprint is SHA256:O5msfNTgOfU4fCnjxHsP/XSp1T8jbTvPPFM43oqCAA0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[129.40.94.33]:29418' (RSA) to the list of known hosts.
remote: Counting objects: 12, done
remote: Finding sources: 100% (12/12)
remote: Getting sizes: 100% (8/8)
remote: Compressing objects: 100% (209/209)
remote: Total 12 (delta 2), reused 6 (delta 0)
Receiving objects: 100% (12/12), 1014 bytes | 338.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
-bash-5.2$
```



GitBucket

Find a repository | Pull requests | Issues | Snippets

- User management
- System settings
- Plugins
- Data export / import
- Database viewer

System settings | Integrations | Authentication

Property	Value
GITBUCKET_VERSION	4.40.0
GITBUCKET_HOME	/home/CECUSER/gitbucket
DATABASE_URL	jdbc:h2:/home/CECUSER/gitbucket/data;MVCC=true GitBucket is using the embedded H2 database. It's recommended that you configure GitBucket to use an external database if you're running GitBucket in a production environment.

Base URL (e.g. `http://example.com/gitbucket`)

http://129.40.94.33:8085

The base URL is used for redirects, notification emails, git repository URL boxes, and more. If the base URL is empty, GitBucket generates the URL from the request information. You can use this property to adjust to URL differences between a reverse proxy and GitBucket.

Site notification (Supports HTML)

Default branch

main

AdminLTE skin name

Skin name Blue

User-defined CSS

Account registration

- ○ **Allow** - Users can create accounts by themselves.
- ◉ **Deny** - Only administrators can create accounts.

Reset password

GitBucket

Find a repository | Pull requests | Issues | Snippets

- User management
- System settings
- Plugins
- Data export / import
- Database viewer

System settings | Integrations | Authentication

Services

☑ **Use Gravatar for profile images**

SSH access

☑ **Enable SSH access to git repository** (Both SSH bind host and Base URL are required if SSH access is enabled)

SSH bind host 129.40.94.33

SSH bind port 29418

SSH public host 129.40.94.33

SSH public port 29418

Communication

☐ **SMTP** (Enable notification as well as SMTP configuration if you want to send notification email too)

SMTP host

SMTP port

SMTP user

SMTP password

Enable SSL ☐

Enable STARTTLS ☐

FROM address

FROM name

Send test mail to: Send

Notifications

☐ **Send notifications**

gitjenkins3 fi | root/newpyre | gitjenkins/ne | Copilot

Not secure | 129.40.94.33:9095/job/gitjenkins3/2/console

Dashboard > gitjenkins3 > first build > Console Output

Polling Log

Git Build Data

Previous Build

Console Output

```
Started by an SCM change
Running as SYSTEM
Building in workspace /home/CECUSER/jenkins/workspace/gitjenkins3
The recommended git tool is: NONE
using credential cecuser
 > git rev-parse --resolve-git-dir /home/CECUSER/jenkins/workspace/gitjenkins3/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url git@github.com:ravisankar-PIO/gitjenkins.git # timeout=10
Fetching upstream changes from git@github.com:ravisankar-PIO/gitjenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.3'
using GIT_SSH to set credentials
Verifying host key using known hosts file
 > git fetch --tags --force --progress -- git@github.com:ravisankar-PIO/gitjenkins.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 0d613d830d6625a527ee7ed45bab60c2a961d71a (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 0d613d830d6625a527ee7ed45bab60c2a961d71a # timeout=10
Commit message: "added another line for print"
 > git rev-list --no-walk e46dbf56f48f1bf14d397406acb6dca37deb472e # timeout=10
[gitjenkins3] $ /bin/sh -xe /tmp/jenkins1246574355650944548 4.sh
N@@$@@@@@@@@%N@@$@@@@@@@@@@@@K@@%hello World
hello git
Finished: SUCCESS
```

REST API Jenkins 2.440.3

Jenkins

Search (CTRL+K)

Ravisankar log out

Dashboard > gitbucketjobonssh > #4 > Console Output

- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#4'
- Git Build Data
- Previous Build

Console Output

```
Started by user Ravisankar
Running as SYSTEM
Building in workspace /home/CECUSER/jenkins/workspace/gitbucketjobonssh
The recommended git tool is: NONE
using credential sshrsakey
 > git rev-parse --resolve-git-dir /home/CECUSER/jenkins/workspace/gitbucketjobonssh/.git #
timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url ssh://git@129.40.94.33:29418/root/newnode.git # timeout=10
Fetching upstream changes from ssh://git@129.40.94.33:29418/root/newnode.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.3'
using GIT_SSH to set credentials
Verifying host key using known hosts file
 > git fetch --tags --force --progress -- ssh://git@129.40.94.33:29418/root/newnode.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/main
Seen 1 remote branch
 > git show-ref --tags -d # timeout=10
Checking out Revision 58176fbff404e26a8ca2927be17945ce026740ae (origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 58176fbff404e26a8ca2927be17945ce026740ae # timeout=10
Commit message: "added new file"
First time build. Skipping changelog.
[gitbucketjobonssh] $ /bin/sh -xe /tmp/jenkins10164004544684664304.sh
N@@$$$$$$$$@$$$K$$%hello world
Finished: SUCCESS
```

REST API Jenkins 2.440.3