Jenkins - how to access, install and development guide

Jenkins is an open-source automation server that plays a pivotal role in supporting the Continuous Integration (CI) and Continuous Delivery (CD) practices of software development. Developed with extensibility in mind, Jenkins facilitates the automation of building, testing, and deploying software, providing developers with a powerful and flexible platform to streamline their development workflows.

Key Features and Uses:

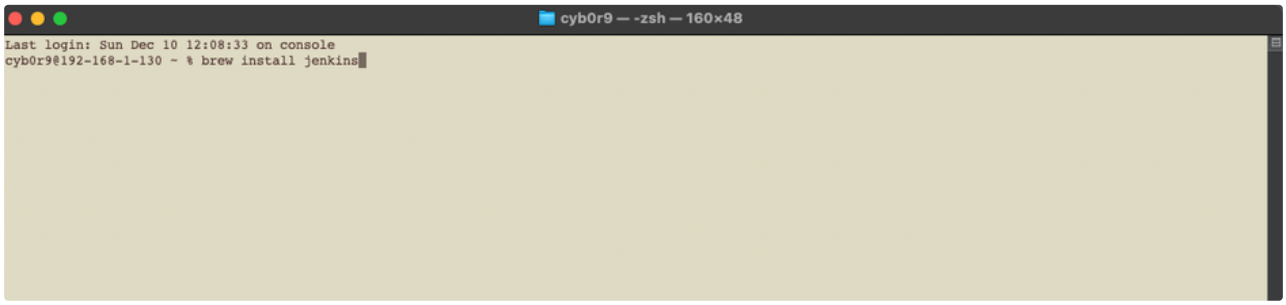
- **Continuous Integration:** Jenkins automates the integration of code changes from multiple contributors in a shared repository, ensuring that the codebase is continuously validated and tested. This helps catch integration issues early in the development process.
- **Build Automation:** Jenkins supports the automation of the build process, allowing developers to compile, package, and assemble their code automatically. This ensures consistency in the build environment and accelerates the delivery of deployable artifacts.
- **Extensibility:** Jenkins boasts a vast ecosystem of plugins that enhance its functionality. These plugins cover a wide range of tools, technologies, and integrations, enabling users to customize Jenkins to suit their specific development and deployment needs.
- **Workflow Orchestration:** Jenkins provides a flexible and extensible workflow engine that allows users to define and automate complex build and deployment pipelines. This enables the creation of sophisticated workflows tailored to the requirements of the software development lifecycle.
- **Distributed Builds:** Jenkins supports the distribution of build and test tasks across multiple machines, allowing for parallel execution. This feature significantly reduces build times and enhances the overall efficiency of the CI/CD pipeline.
- **Easy Integration:** Jenkins can be seamlessly integrated with version control systems (e.g., Git, SVN), build tools (e.g., Maven, Gradle), and deployment platforms (e.g., Docker, Kubernetes). This integration ensures a smooth and cohesive development and deployment process.

Benefits:

- **Accelerated Delivery:** By automating repetitive tasks such as building, testing, and deployment, Jenkins significantly accelerates the software delivery process. This results in faster time-to-market for new features and bug fixes.
- **Improved Code Quality:** Continuous Integration with Jenkins ensures that code changes are validated through automated tests, reducing the likelihood of introducing defects into the codebase. This leads to higher code quality and more reliable software.
- **Enhanced Collaboration:** Jenkins promotes collaboration among development and operations teams by providing a centralized platform for building, testing, and deploying code. This shared environment encourages communication and facilitates a DevOps culture.
- **Cost-Efficiency:** Jenkins is an open-source tool, eliminating the need for costly licenses. Its extensibility and ability to integrate with various tools reduce manual intervention, resulting in cost savings and efficient resource utilization.
- **Flexibility and Customization:** The extensibility of Jenkins through plugins allows organizations to tailor the automation server to their specific needs. This flexibility ensures that Jenkins can adapt to different development and deployment scenarios, making it a versatile tool for various projects.

The following steps are done on a macOs. In order to create your own Jenkins service running on your local machine, please follow the steps as mentioned below -

1. Open Terminal on your mac and then type the command “brew install jenkins” like below -



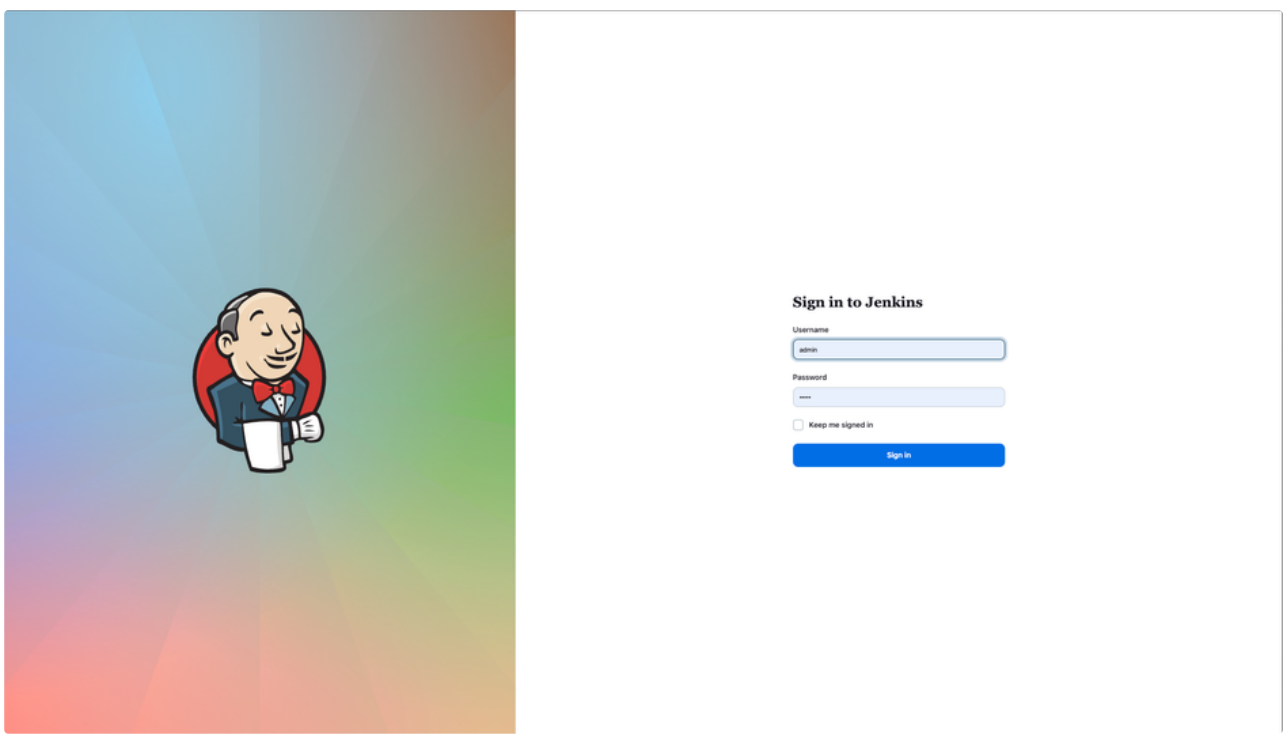
2. Wait for the installation to be completed and then save the password which comes up at the end of installation. At the end of the installation, you will get this message -

```
To start jenkins now and restart at login:
  brew services start jenkins
Or, if you don't want/need a background service you can just run:
  /opt/homebrew/opt/jenkins/bin/jenkins --httpListenAddress\=127.0.0.1 --httpPort\=8080
```

3. Run the command provided to start up the Jenkins server -

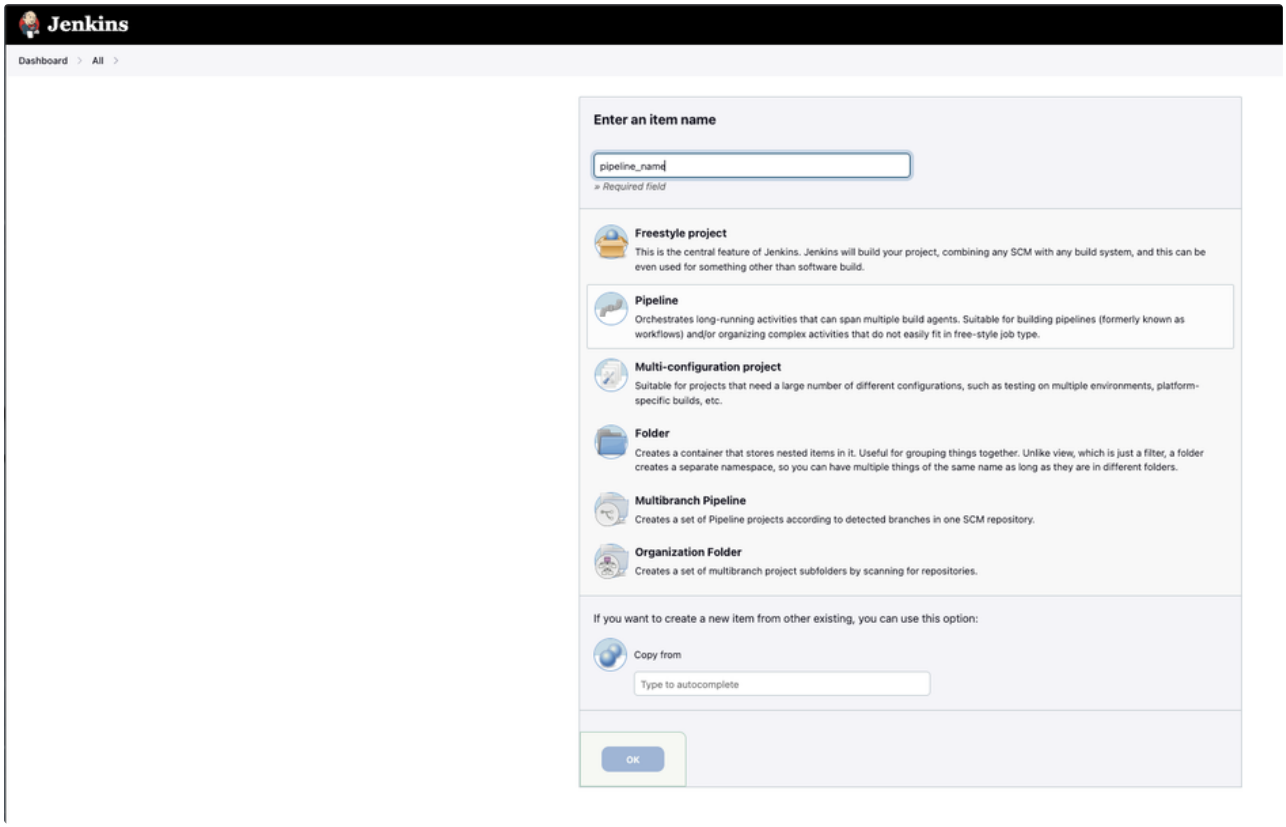
```
cyb0r9@192-168-1-130 ~ % /opt/homebrew/opt/jenkins/bin/jenkins --httpListenAddress=127.0.0.1 --httpPort=8080

Running from: /opt/homebrew/Cellar/jenkins/2.435/libexec/jenkins.war
webroot: /Users/cyb0r9/.jenkins/war
2023-12-10 04:12:42.734+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war file
2023-12-10 04:12:43.237+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2023-12-10 04:12:43.258+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-10.0.18; built: 2023-10-27T01:59:58.245Z; git: 8545fd9bf4cd0d0838f62
6b405fd4963441546b7; jvm 21.0.1
2023-12-10 04:12:43.374+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJsp
Servlet
2023-12-10 04:12:43.391+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: Session workerName=node0
2023-12-10 04:12:43.582+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory: /Users/cyb0r9/.jenkins found at: $user.home/.jenkins
2023-12-10 04:12:44.609+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started w.@3e1624c7{Jenkins v2.435,/,file:///Users/cyb0r9/.jenkins/war/,AVA
ILABLE}{/Users/cyb0r9/.jenkins/war}
2023-12-10 04:12:44.616+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started ServerConnector@14fa86ae{HTTP/1.1, (http/1.1)}{127.0.0.1:8080}
2023-12-10 04:12:44.623+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started Server@7cbd9d24{STARTING}[10.0.18,sto=0] @2102ms
2023-12-10 04:12:44.623+0000 [id=41]    INFO    winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2023-12-10 04:12:44.683+0000 [id=46]    INFO    hudson.PluginManager#loadDetachedPlugins: Upgrading Jenkins. The last running version was 2.415. This Jenkins is ve
rsion 2.435.
2023-12-10 04:12:44.684+0000 [id=47]    INFO    jenkins.InitReactorRunner$1#onAttained: Started initialization
2023-12-10 04:12:44.712+0000 [id=46]    INFO    hudson.PluginManager#loadDetachedPlugins: Upgraded Jenkins from version 2.415 to version 2.435. Loaded detached plu
gins (and dependencies): [matrix-auth.hpi]
2023-12-10 04:12:44.787+0000 [id=60]    INFO    jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2023-12-10 04:12:45.846+0000 [id=59]    INFO    jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2023-12-10 04:12:45.851+0000 [id=56]    INFO    jenkins.InitReactorRunner$1#onAttained: Started all plugins
2023-12-10 04:12:45.853+0000 [id=62]    INFO    jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2023-12-10 04:12:45.995+0000 [id=55]    INFO    h.p.b.g.GlobalTimeOutConfiguration#load: global timeout not set
2023-12-10 04:12:46.127+0000 [id=50]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2023-12-10 04:12:46.127+0000 [id=46]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2023-12-10 04:12:46.161+0000 [id=52]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2023-12-10 04:12:46.167+0000 [id=61]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2023-12-10 04:12:46.182+0000 [id=55]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2023-12-10 04:12:46.210+0000 [id=40]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
```

4. Once you get the following confirmation that Jenkins is up and running, open your browser and navigate to the URL - localhost:8080. You will get a page similar to this -

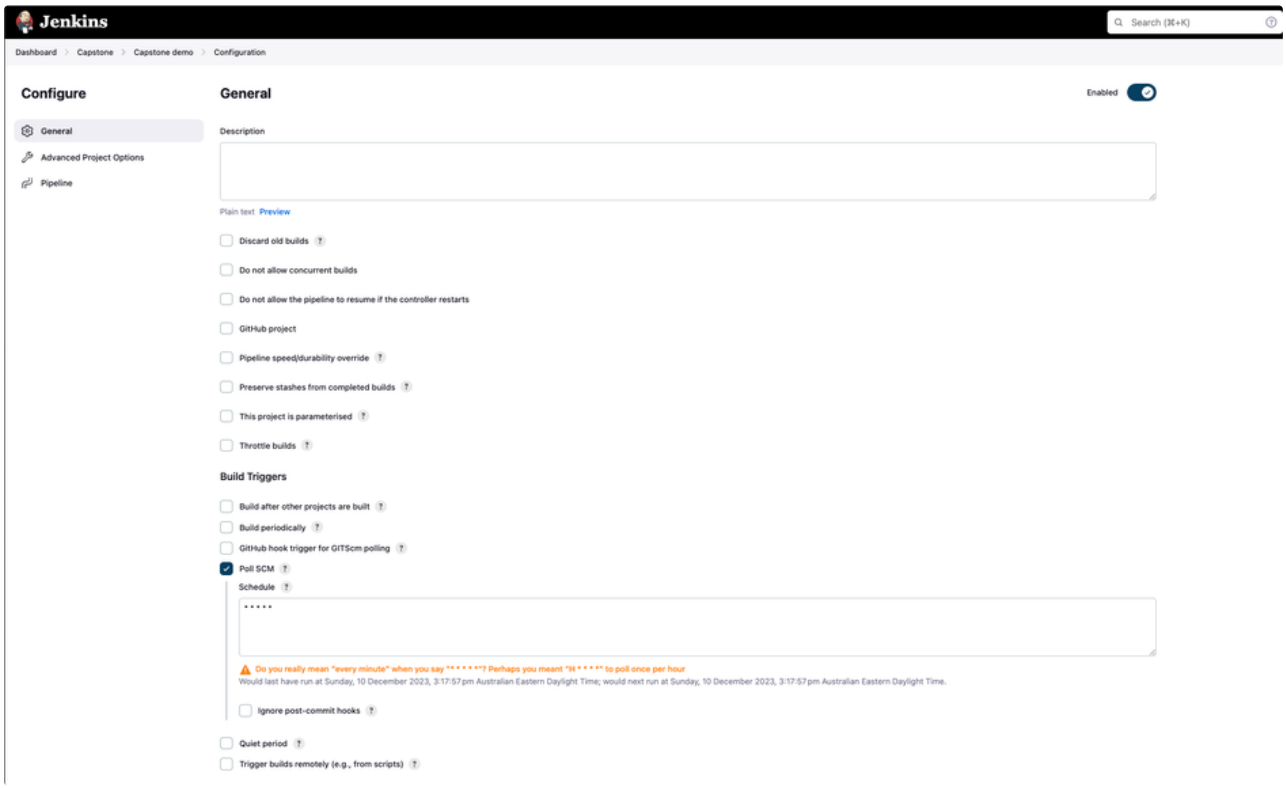


Use the credentials which you created during installation process and click on “Sign in”.

5. Once you are logged in, navigate to Dashboard > New Item. Provide a name to your pipeline and select “Pipeline” from the list provided. Click on “OK” to create the blank pipeline.

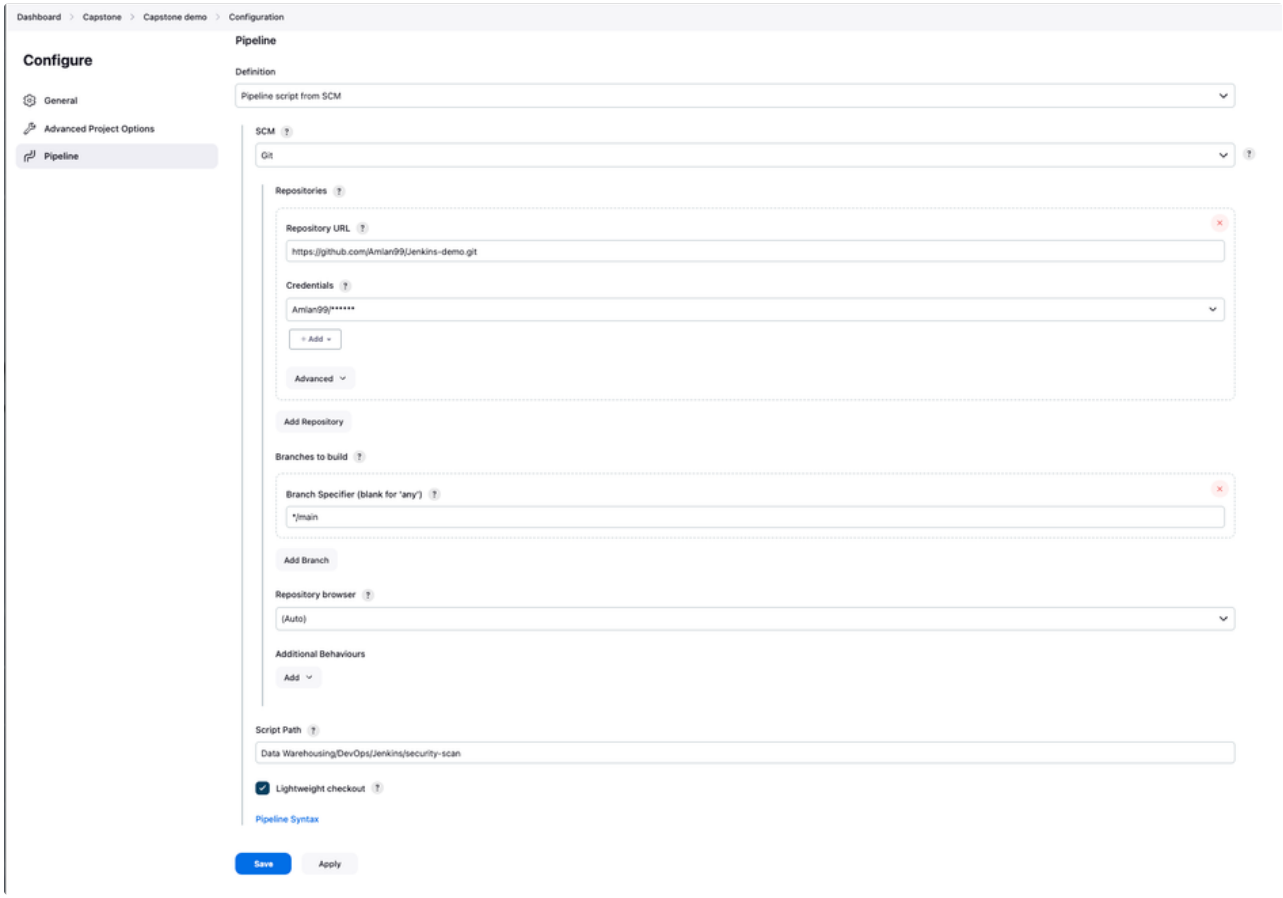


6. For this setup, we want Jenkins to be integrated with GitHub, so that it can trigger the pipeline as soon as it recognises any change in the GitHub repository. For this, in the Configure space, under General, we will select the checkbox next to “Poll SCM” and provide the Schedule as “* * * * *”, which means run this every minute. Note that this is crontab notation.

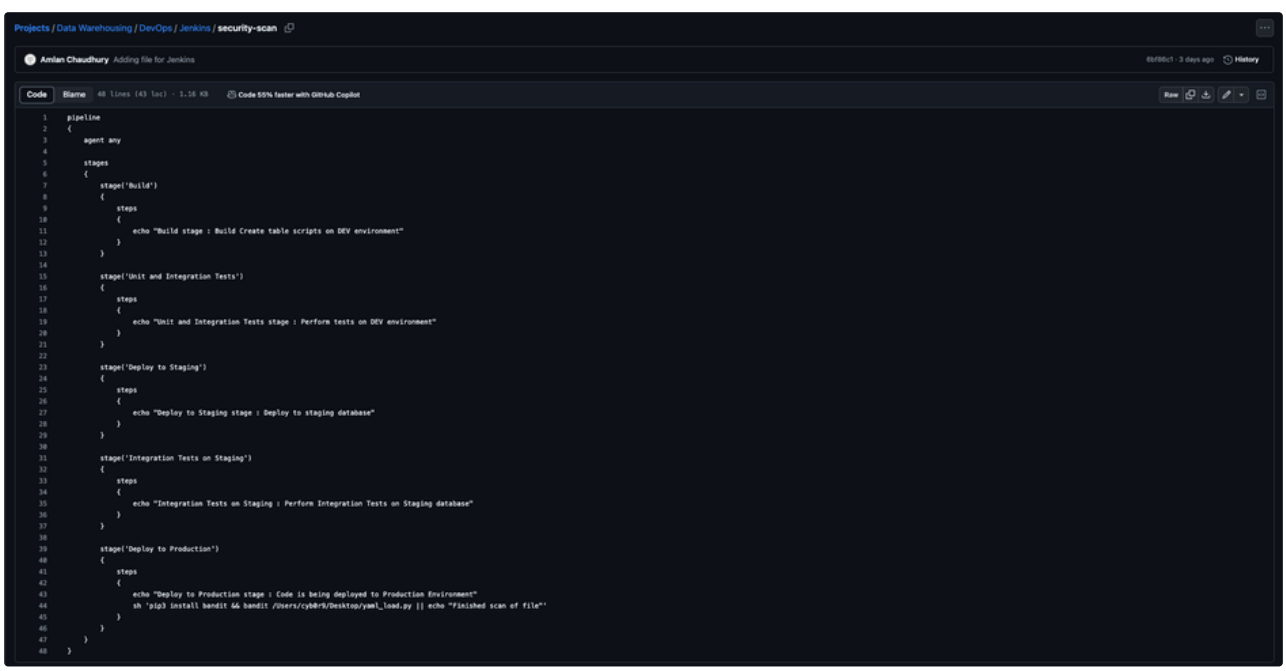


7. Navigate next to “Pipeline” from the left-hand pane. Here under “Definition”, select “Pipeline script from SCM”, select the “SCM” as “Git” and next provide the Repository URL. Add your GitHub credentials in the next step under “Credentials”. After that, under “Branches to

build”, select the GitHub branch. In this POC, I have used the “main” branch, but it can be any branch. Finally, select the “Script Path” and provide the full path to the Jenkinsfile present in the GitHub repository.



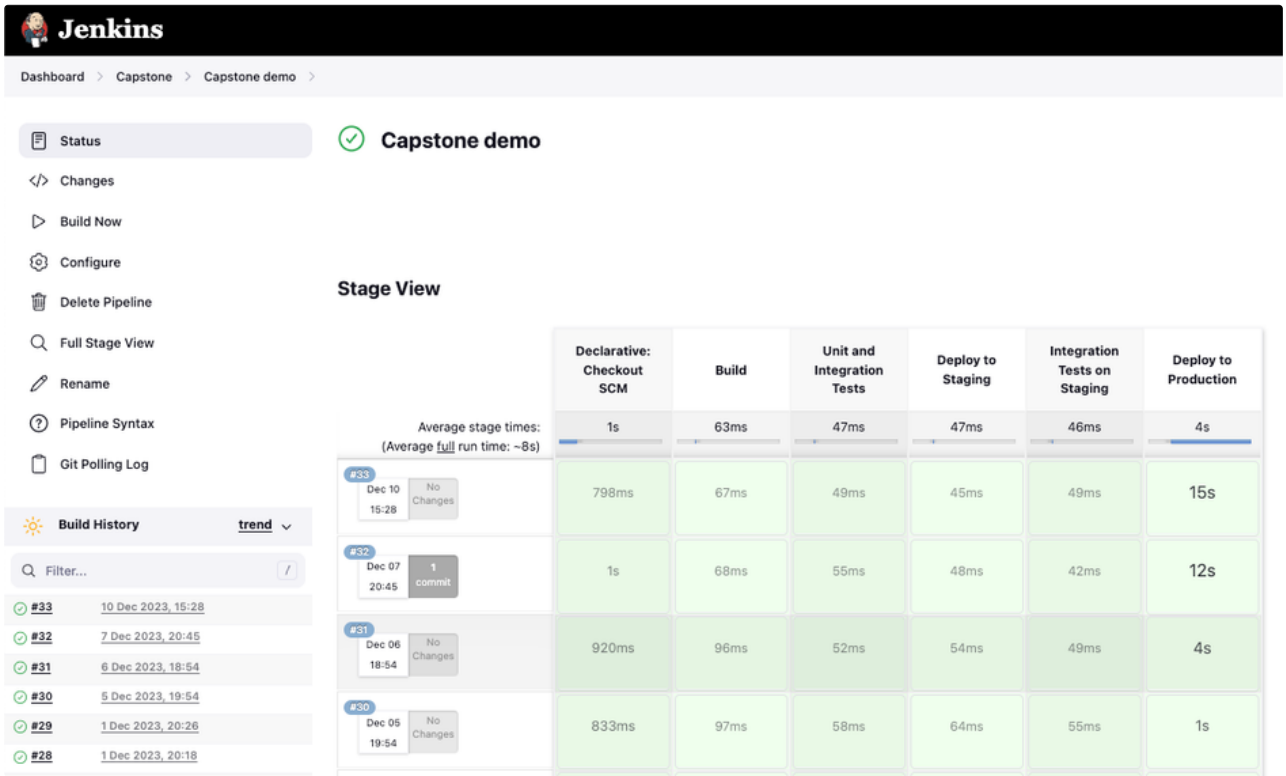
8. This is the file which has been used for the Jenkins POC. I am not selecting any agent, but choosing it as “any”. Under “stages”, I have created some print statements to show what each step is doing. In the “Deploy to Production” step, I have installed bandit package and used it to perform a security scan on the file named as yaml_load.py. This is a sample script for checking vulnerabilities using the bandit library in python.



I have used the command - ***sh 'pip3 install bandit && bandit /Users/cyb0r9/Desktop/yaml_load.py || echo "Finished scan of file"*** to first install the bandit library, then perform checking of the sample file and finally echo a successful statement at the end. I have used a small trick to ensure that the process completes successfully by adding the final print statement in a OR condition so that the process completion will always succeed.

Click “Save” and exit the pipeline configuration.

9. Now push your Jenkinsfile into the GitHub repository as mentioned in the Configuration and click on the pipeline created. The pipeline has been configured to run every minute so wait 1 min for the pipeline to auto-trigger. After the pipeline completes, the page will look like this -



10. Click on the latest build and navigate to “Console Output” to view the complete log of the build. You will be able to see the following outputs -

Jenkins

Search (⌘+K)

Dashboard > Capstone > Capstone demo > #33

- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete build '#33'
- Git Build Data
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces
- Previous Build

Console Output

```
Started by user Amlan Chaudhury
Obtained Data Warehousing/DevOps/Jenkins/security-scan from git https://github.com/Amlan99/Jenkins-demo.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /Users/cyb0r9/.jenkins/workspace/Capstone demo
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential 0f4159da-b7f6-4500-9da1-46468d2cc736
 > git rev-parse --resolve-git-dir /Users/cyb0r9/.jenkins/workspace/Capstone demo/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Amlan99/Jenkins-demo.git # timeout=10
Fetching upstream changes from https://github.com/Amlan99/Jenkins-demo.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.3 (Apple Git-145)'
using GIT_ASKPASS to set credentials
 > git fetch --tags --force --progress -- https://github.com/Amlan99/Jenkins-demo.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 24cc3582328f726da32967a8627d0a21073e8cd0 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 24cc3582328f726da32967a8627d0a21073e8cd0 # timeout=10
Commit message: "Renamed folders and files"
 > git rev-list --no-walk 24cc3582328f726da32967a8627d0a21073e8cd0 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Build stage : Build Create table scripts on DEV environment
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Unit and Integration Tests)
[Pipeline] echo
Unit and Integration Tests stage : Perform tests on DEV environment
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Staging)
[Pipeline] echo
Deploy to Staging stage : Deploy to staging database
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Integration Tests on Staging)
[Pipeline] echo
Integration Tests on Staging : Perform Integration Tests on Staging database
[Pipeline] }
```

Dashboard > Capstone > Capstone demo > #33

```
[Pipeline] echo
Deploy to Production stage : Code is being deployed to Production Environment
[Pipeline] sh
+ pip3 install bandit
Requirement already satisfied: bandit in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (1.7.5)
Requirement already satisfied: GitPython>=1.0.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from bandit) (3.1.40)
Requirement already satisfied: PyYAML>=5.3.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from bandit) (6.0.1)
Requirement already satisfied: stevedore>=1.20.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from bandit) (5.1.0)
Requirement already satisfied: rich in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from bandit) (13.7.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from GitPython>=1.0.1->bandit) (4.0.11)
Requirement already satisfied: pbr!=2.1.0,>=2.0.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from stevedore>=1.20.0->bandit) (6.0.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from rich->bandit) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /Users/cyb0r9/Library/Python/3.10/lib/python/site-packages (from rich->bandit) (2.16.1)
Requirement already satisfied: smmap<6,>=3.0.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from gitdb<5,>=4.0.1->GitPython>=1.0.1->bandit) (5.0.1)
Requirement already satisfied: mdurl~=0.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from markdown-it-py>=2.2.0->rich->bandit) (0.1.2)
+ bandit /Users/cyb0r9/Desktop/yaml_load.py
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.10.11
Run started:2023-12-10 04:28:34.527905

Test results:
>> Issue: [B506:yaml_load] Use of unsafe yaml load. Allows instantiation of arbitrary objects. Consider yaml.safe_load().
   Severity: Medium   Confidence: High
   CWE: CWE-20 (https://cwe.mitre.org/data/definitions/20.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b506_yaml_load.html
   Location: /Users/cyb0r9/Desktop/yaml_load.py:9:8
8           ystr = yaml.dump({'a': 1, 'b': 2, 'c': 3})
9           y = yaml.load(ystr)
10          yaml.dump(y)

--------------------------------------------------
>> Issue: [B506:yaml_load] Use of unsafe yaml load. Allows instantiation of arbitrary objects. Consider yaml.safe_load().
   Severity: Medium   Confidence: High
   CWE: CWE-20 (https://cwe.mitre.org/data/definitions/20.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b506_yaml_load.html
   Location: /Users/cyb0r9/Desktop/yaml_load.py:22:0
21
22      yaml.load("{}", Loader=yaml.Loader)
23

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

Code scanned:
        Total lines of code: 19
        Total lines skipped (#nosec): 0
        Total potential issues skipped due to specifically being disabled (e.g., #nosec BXXX): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 2
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 2
Files skipped (0):
+ echo 'Finished scan of file'
Finished scan of file
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

As can be seen above, the process installed the bandit package and then performed the security vulnerability assessment for the sample file. The run metrics of the file output can be seen above and we can see 2 medium issues which are scanned with high confidence.

If you have followed the above steps in order, you will be able to setup Jenkins on your local machine, integrate it with GitHub and perform vulnerability assessment of sample file.

For installation on windows, please follow the steps mentioned here - Windows.