

Jenkins

Jenkins is an **automation tool** used to create CI/CD pipelines that build, test and deploy software. It is not limited to that it can automate anything, you can run your python, bash scripts as well as your Ansible playbooks.

Additionally, it will have various plugins that can be used that work well with the testing tools like Junit and Selenium. It can be installed on operating system like windows, Linux, MacOS.

It has a web interface, it can create a job and create a functionality you want to like build triggers, provide RBAC and encryption.

Jenkins Infrastructure

Master server which controls the pipelines and schedules the builds to the agent. Then the **Agents/Minions** will run the build on the workspace.

Example workflow:

1. A developer commits a code to the Git Repository, Master becomes aware of this commit and triggers the appropriate pipeline and then distributes the builds to one of the agents to run, selects the appropriate agent based on the labels.
2. Agents run the builds, which are basically Linux commands to build, test and deploy the code.



Jenkins Agents

Permanent Agent: This are just dedicated, everyday standalone Linux, windows servers set up to run Jenkins job. You need to have JAVA installed on the servers and set up a SSH as master makes the communication using SSH. Necessary tools must also be installed on the servers.

Cloud Agents: These are more popular choices in the real world, Ephemeral/Dynamic agents spun up on the demand based on the agents template you configure.

Example: Docker, Kubernetes and AWS fleet manager.

Build Types

Freestyle builds Project: Simplest method to create a build. You can use UI and plugin to manage the build. Mostly they will be set up to execute a shell script.

Pipeline:

Pipeline used groovy syntax to specify what happens in the build. Pipelines are mostly broken down into stages.

Common stages of pipeline are:

Clone: Usually clone the local repository and set up environment variables on the agent.

Build: It will take up the code and builds it, means generating local artifacts like a JAR file, executable or a container image.

Test: Test against the newly built code.

Package: It gets packaged ready for the deployment.

Deploy: Sent out artifacts to our registry.

Install Jenkins

You can directly install Jenkins on the OS or deploy it in a container like docker or Kubernetes, suggested method is to use in the docker.

Install **blue ocean** is popular addon for the Jenkins which provides a new user experience based on personalized, modern design that allows users to graphically create, visualize and diagnose CI/CD pipelines.

I'm using AWS EC2 instance to launch Jenkins via a docker container, please follow the below steps:

Prerequisites

- A system with Docker installed. You can download Docker from the official website.
- An active internet connection.

Step 1: Connect to your EC2 instance and you can follow Jenkins installation as per your OS form the official Jenkins page: <https://www.jenkins.io/doc/book/installing/docker/> or clone my Git repository link: <https://github.com/manasi09-pd/DOCKER.git>

Step2: I have ubuntu OS, use the build command to build the docker Image, and use docker file.

Add the ubuntu user to the docker group, it should have Sudo permission.

- docker build -t myjenkins-blueocean:2.414.2 .
- #IF you are having problems building the image yourself
- docker pull devopsjourney1/jenkins-blueocean:2.332.3-1 && docker tag devopsjourney1/jenkins-blueocean:2.332.3-1 myjenkins-blueocean:2.332.3-1

```
ubuntu@ip-172-31-39-42:~$ ls
DOCKER
ubuntu@ip-172-31-39-42:~$ cd DOCKER/
ubuntu@ip-172-31-39-42:~/DOCKER$ ls
README.md  first-docker  flask-docker  jenkins-docker  kubernetes-docker
```

```
FROM jenkins/jenkins:2.414.2-jdk11
USER root
RUN apt-get update && apt-get install -y lsb-release python3-pip
RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
    https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
    signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
    https://download.docker.com/linux/debian \
    $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean:1.25.3 docker-workflow:1.28"
```

```
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker$ ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker$ docker build -t myjenkins-blueocean:2.414.2 .
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker.sock:_ping": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker$ sudo docker build -t myjenkins-blueocean:2.414.2 .
[+] Building 61.5s (8/9)
=> sha256:5e0053fb65f36e86b9387ea544e50bf21f9368e872f91ad0a2bc9abdc672472 8.68MB
=> sha256:b705323eaef70a7da4c1eed8bb16f3d3ff2d5c83671170a2c217cf77faa4f15432 3.12kB / 3.12kB
=> sha256:3d6d48f61941fc7cb5253b7efdf932d06ee6e33545f6f535ac3d1049e385a7e 12.62kB / 12.62kB
=> sha256:f6154b0007a4a7dceef550de8b863dd0bf177961a972a33c54b2c4396dc4a873bf 61.32MB / 61.32MB
=> sha256:34d37335955786e00f8bb8e7bd6fa2zfd7739dc047ba49c637c18a5d54b99023c 1.23kB / 1.23kB
=> sha256:2471c7529ac37af126eaac1afad134f08afa5ce84f7e10816e314d28ce7dc38 89.33MB / 89.33MB
=> sha256:012c0b3e9981a0c0bedc712eaaaf1b18850529dd026a04a1ce13fdb39e42b 3.1s
=> extracting sha256:012c0b3e9981a0c0bedc712eaaaf1b18850529dd026a04a1ce13fdb39e42b 4.3s
=> sha256:10cad8b7dbf876c2afb6402c5706eb77d723bde838daef8c2fadab3628e4f768 183B / 183B
=> sha256:655eafc427231f288d1b1c0f0a887a3165d2eb9345ab44ed75063bed17ae7 189B / 189B
=> sha256:c1a5951d1a67db7eb3dd0df7f6734a26340476b9d1f6dd5866888d72b4833847 6.09MB / 6.09MB
=> sha256:614d8aab6ca95f7a9ab39388cf16a55f34eb54e0bcc52d4c3ef59033ac46208cd8 77.15MB / 77.15MB
=> sha256:95df44e08b1d468bae93b830b9aa4d9cf032d5be66c478dc70fb48fb6bf62 1.93kB / 1.93kB
=> sha256:c8f695ed0809beb2a95df25c497c499a65746c448fb5baef79bf42313ec7a49 1.16kB / 1.16kB
=> sha256:0809beb2a95df25c497c499a65746c448fb5baef79bf42313ec7a49 2.4s
```

Step3: Create the network for Jenkins by using the command below. This isolation can be useful when you want to group containers together logically and control their communication.

```
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker$ sudo docker network create jenkins  
680072ee301b5ff73b2e20ec5b71a51812a3e7310b3a6a905566ac8bc082edf5
```

```
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker$ sudo docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
efa26d92a8d6    bridge    bridge      local  
2310246b1da9    host      host       local  
680072ee301b    jenkins   bridge      local  
74929e22421c    none     null       local
```

Step4: Create the docker container. Specifying the network Jenkins and environment variables specifying the location of certificates and volumes.

```
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker$ sudo docker run \  
  --name jenkins-blueocean \  
  --restart=on-failure \  
  --detach \  
  --network jenkins \  
  --env DOCKER_HOST=tcp://docker:2376 \  
  --env DOCKER_CERT_PATH=/certs/client \  
  --env DOCKER_TLS_VERIFY=1 \  
  --publish 8080:8080 \  
  --publish 50000:50000 \  
  --volume jenkins-data:/var/jenkins_home \  
  --volume jenkins-docker-certs:/certs/client:ro \  
  myjenkins-blueocean:2.492.1-1  
9715ad345ef94f260af75a6cf61d241c2e737b567e19fdf9c45e9865afc69e10
```

```
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker$ docker ps  
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS  
NAMES              COMMAND  
9715ad345ef9    myjenkins-blueocean:2.492.1-1   "/usr/bin/tini -- /u..."   About a minute ago   Up About a minute   0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp, 0.0.0.0:50000->50000/tcp, [::]:50000->50000/tcp   jenkins-blueocean
```

Step5: Access your Jenkins WEB page via your private IP address and if its not opening add custom TCP port 8080, edit the inbound traffic rules.

-	sgr-0dd9b4f1a9c904cb5	8080	TCP	0.0.0.0/0
---	-----------------------	------	-----	-----------

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

Step6: To find the password, docker exec command can be used to check the admin password.

```
ubuntu@ip-172-31-39-42:~$ docker exec -it 9715ad345ef9 /bin/bash
```

Step7: Installed the suggested plugins for Jenkins.

Getting Started

Getting Started

✓ Folders Plugin	✓ Pipeline API	✓ Git	✓ LDAP	✓ Pipeline Graph View	✓ Dark Theme	✓ JSON Path API	✓ Token Macro	✓ Build Timeout	✓ Credentials Binding	✓ Timestampper	✓ Resource Disposer	✓ Workspace Cleanup

jenkins@9715ad345ef9:~/secrets\$ pwd
/var/jenkins_home/secrets
jenkins@9715ad345ef9:~/secrets\$ ls
initialAdminPassword jenkins.model.Jenkins.crumbSalt master.key org.jenkinsci.main.modules.instance_identity.InstanceIdentity.KEY
jenkins@9715ad345ef9:~/secrets\$

Step8: Create the Administrator account.

Create First Admin User

Username

Password

Confirm password

Jenkins 2.492.1

[Skip and continue as admin](#)

[Save and Continue](#)

Step9: Default configuration can be used or provide a different hostname or IP.

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.492.1

[Not now](#)

[Save and Finish](#)

Step10: Jenkins WEB GUI page is now ready to use.

Dashboard: It's a home page, easy to navigate through Jenkins.

New Item: creating projects and pipelines.

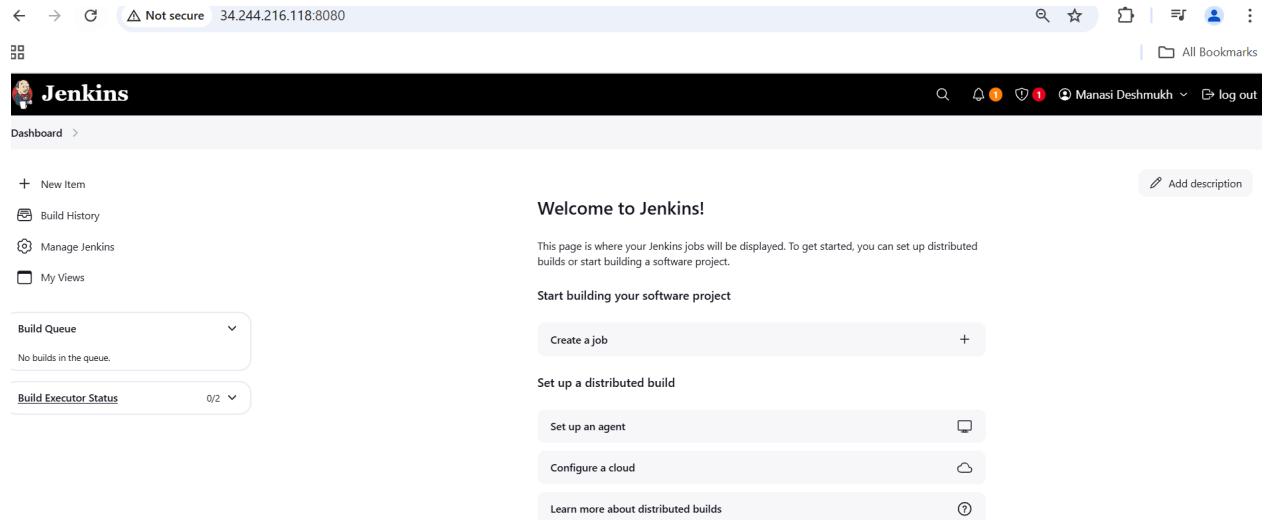
People: You administrate people and user accounts.

Build History: Gives history for all your jobs that have been run.

Manage Jenkins; This has all the management stuff like install plugins, agents.

My view: Just a way to organize your jobs.

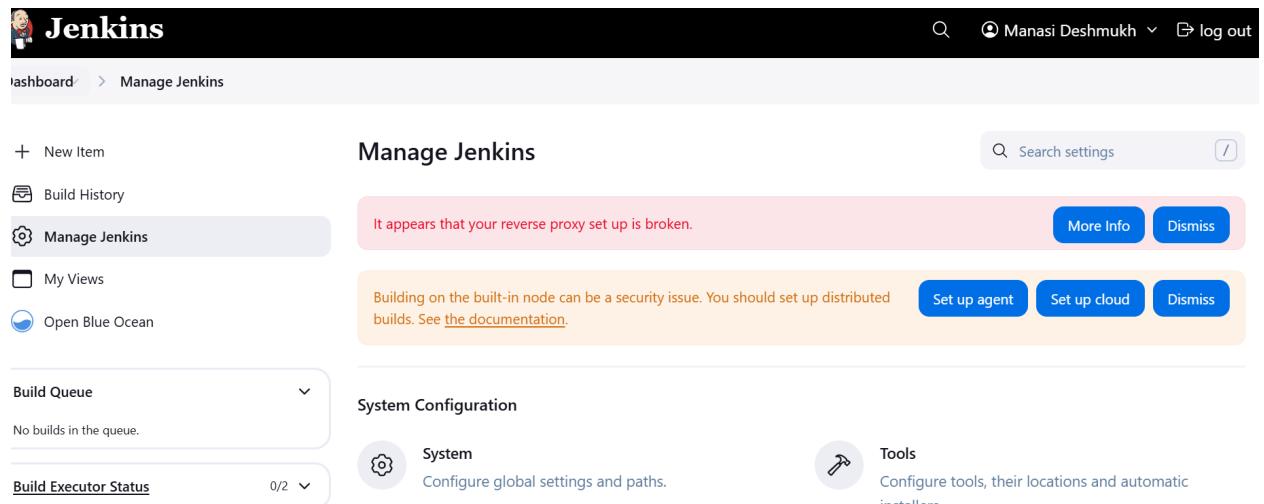
Blue ocean: Plugin that enhances the CI/CD pipelines.



The screenshot shows the Jenkins dashboard at <http://34.244.216.118:8080>. The top navigation bar includes links for 'Dashboard', 'Build History', 'Manage Jenkins', 'My Views', 'Build Queue' (empty), and 'Build Executor Status' (0/2). The main content area features a 'Welcome to Jenkins!' message and a 'Start building your software project' section with links for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'.

On the **Manage Jenkins** page, you would see some errors like reverse proxy this issue occurs if you have used the public IP address for your ec2 instance, wherever the ec2 instance goes down the Public IP address changes so for that you can attach a elastic IP address to the EC2 instance.

Agent needs to be set up so that it can run the pipeline.

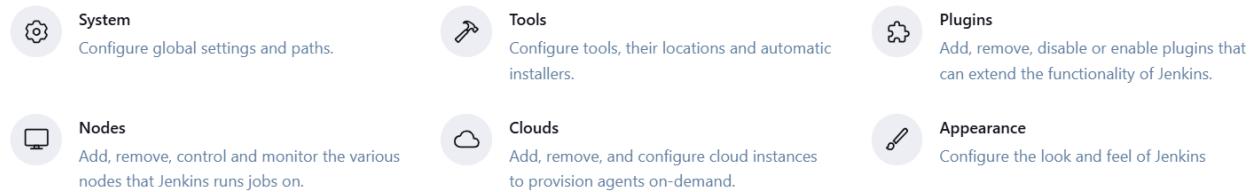


The screenshot shows the 'Manage Jenkins' page. The left sidebar lists 'New Item', 'Build History', 'Manage Jenkins' (selected), 'My Views', and 'Open Blue Ocean'. The main content area displays a red error message: 'It appears that your reverse proxy set up is broken.' Below it, a yellow warning message states: 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)'. There are 'More Info' and 'Dismiss' buttons for the error message. The 'System Configuration' section contains 'System' (with a gear icon) and 'Tools' (with a wrench icon). The 'System' section includes a note: 'Configure global settings and paths.'

System Configuration:

Information of a Jenkins server, set global parameters, manage plugins Jenkins administrator spend lot of time here, manage node an clouds we can set up the agents, docker or Kubernetes.

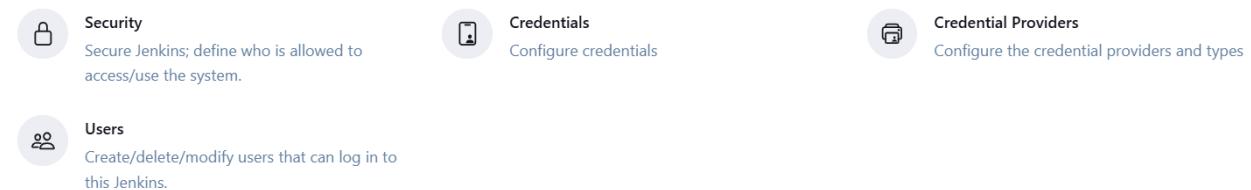
System Configuration



Security:

Configure global security, manage the credentials and the users, store SSH keys, API keys. You can also use AWS secrets manager to store the keys, or Jenkins has a built-in credentials storage manager.

Security



System information, logs and statistics.

Status Information

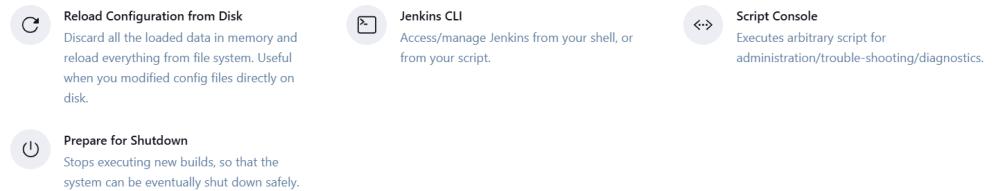


About Jenkins

See the version and license information.

Other tools like preparing for shutdown, use this on the days where you need to shut down the Jenkins server for maintenance, if you reboot it the current jobs will stop, so prepare for shutdown Jenkins will complete the jobs that are running and will not take up any new jobs.

Tools and Actions



Let's start with the freestyle project:

Don't provide space while entering a new item name, as at the back end the Jenkins job will create a folder. It's recommended to use _ or – instead.

New Item

Enter an item name

my_first_job

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



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.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

OK

Then with the project settings, you can set the git repository hosted on this project and build triggers you can set the webhook triggers. If you have a firewall, then you need to open the ports or use a proxy server. Another way is to use SCM Jenkins will reach the GitHub and check regularly for any changes.

The screenshot shows the Jenkins configuration interface for a project named 'my_first_job'. The 'General' tab is active. In the 'Source Code Management' section, the 'None' option is selected. Under the 'Triggers' section, there are several options listed: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. None of these checkboxes are checked.

Another option is built periodically like a corn job, For build environment delete the workspace before build starts make sure has clean environment and there are no artifacts by the previous build on Jenkins.

Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans
- Terminate a build if it's stuck
- With Ant ?

Build steps: what steps work while building a project, post steps after the build like email notification or slack notification by plugin.

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Add build step ▾

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

Add post-build action ▾

Added first shell command “Hello World” and run the build its successful.

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

≡ Execute shell ?

Command

See [the list of available environment variables](#)

```
echo "Hello World"
```

Status

Changes

Console Output

Edit Build Information

Delete build '#1'

Timings

Open Blue Ocean

Started by user [Manasi Deshmukh](#)
 Running as SYSTEM
 Building in workspace /var/jenkins_home/workspace/my_first_job
 [WS-CLEANUP] Deleting project workspace...
 [WS-CLEANUP] Deferred wipeout is used...
 [my_first_job] \$ /bin/sh -xe /tmp/jenkins8123565687590607330.sh
 + echo Hello World
 Hello World
 Finished: SUCCESS

Environment variables: set in Jenkins to view your build, most popular on BUILD_ID will give you current build ID. BUILD_URL gives the URL for the build.

Dashboard >

BRANCH_NAME
 For a multibranch project, this will be set to the name of the branch being built, for example in case you wish to deploy to production from master but not from feature branches; if corresponding to some kind of change request, the name is generally arbitrary (refer to CHANGE_ID and CHANGE_TARGET).

BRANCH_IS_PRIMARY
 For a multibranch project, if the SCM source reports that the branch being built is a primary branch, this will be set to "true"; else unset. Some SCM sources may report more than one branch as a primary branch while others may not supply this information.

CHANGE_ID
 For a multibranch project corresponding to some kind of change request, this will be set to the change ID, such as a pull request number, if supported; else unset.

CHANGE_URL
 For a multibranch project corresponding to some kind of change request, this will be set to the change URL, if supported; else unset.

CHANGE_TITLE
 For a multibranch project corresponding to some kind of change request, this will be set to the title of the change, if supported; else unset.

CHANGE_AUTHOR
 For a multibranch project corresponding to some kind of change request, this will be set to the username of the author of the proposed change, if supported; else unset.

CHANGE_AUTHOR_DISPLAY_NAME
 For a multibranch project corresponding to some kind of change request, this will be set to the human name of the author, if supported; else unset.

CHANGE_AUTHOR_EMAIL
 For a multibranch project corresponding to some kind of change request, this will be set to the email address of the author, if supported; else unset.

CHANGE_TARGET
 For a multibranch project corresponding to some kind of change request, this will be set to the target or base branch to which the change could be merged, if supported; else unset.

CHANGE_BRANCH
 For a multibranch project corresponding to some kind of change request, this will be set to the name of the actual head on the source control system which may or may not be different from BRANCH_NAME. For example in GitHub or Bitbucket this would have the name of the origin branch whereas BRANCH_NAME would be something like PR-24.

CHANGE_FORK
 For a multibranch project corresponding to some kind of change request, this will be set to the name of the forked repo if the change originates from one; else unset.

TAG_NAME
 For a multibranch project corresponding to some kind of tag, this will be set to the name of the tag being built, if supported; else unset.

TAG_TIMESTAMP
 For a multibranch project corresponding to some kind of tag, this will be set to a timestamp of the tag in milliseconds since Unix epoch, if supported; else unset.

TAG_UNIXTIME

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Execute shell

Command

See [the list of available environment variables](#)

```
echo "Hello World"
echo "Build ID of this Job is ${BUILD_ID}"
echo "Build URL of this Job is ${BUILD_URL}"
```

Console Output

```
Started by user Manasi Deshmukh
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/my_first_job
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
[my_first_job] $ /bin/sh -xe /tmp/jenkins8727276288167914102.sh
+ echo Hello World
Hello World
+ echo Build ID of this Job is 2
Build ID of this Job is 2
+ echo Build URL of this Job is http://54.220.92.223/:8080/job/my_first_job/2/
Build URL of this Job is http://54.220.92.223/:8080/job/my_first_job/2/
Finished: SUCCESS
```

- Standard input is with + command and standard output is the command that has run.

In the workspace we will have the files that have been created and stored during the build.

The screenshot shows the Jenkins interface for the 'my_first_job' project. At the top, there's a navigation bar with the Jenkins logo and the text 'Dashboard > my_first_job >'. Below this, on the left, is a sidebar with various project management options: Status (highlighted), Changes, Workspace, Build Now, Configure, Delete Project, Favorite, Open Blue Ocean, and Rename. On the right, the main content area displays the project name 'my_first_job' with a green checkmark icon. Below the project name is a section titled 'Permalinks' containing a bulleted list of four links: 'Last build (#3), 55 sec ago', 'Last stable build (#3), 55 sec ago', 'Last successful build (#3), 55 sec ago', and 'Last completed build (#3), 55 sec ago'.

The screenshot shows the Jenkins workspace interface for the job 'my_first_job'. The top navigation bar includes 'Dashboard > my_first_job > Workspace of my_first_job on Built-In Node'. Below this, there are tabs for 'Status', 'Changes', and 'Workspace', with 'Workspace' being the active tab. A file named 'test.txt' is listed under the workspace, with details: Mar 4, 2025, 5:55:12 AM, 5 B, and a link to download all files in zip. There is also a 'Wipe Out Current Workspace' button.

If the Job is run again the file is new file is stored along with older file, you can enable the option of delete the workspace for more cleaner builds.

Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

Delete workspace before build starts

Advanced ▾

To Understand the actual **file system of Jenkins**, you can go to the Jenkins master container.

```
ubuntu@ip-172-31-39-42:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
9715ad345ef9 myjenkins-blueocean:2.492.1-1 "/usr/bin/tini -- /u..." 16 hours ago Up 3 hours 0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp, 0.0.0.0:50000->50000/[...]
ubuntu@ip-172-31-39-42:~$ docker exec -it 9715ad345ef9 /bin/bash
jenkins@9715ad345ef9:$ 
```

```
jenkins@9715ad345ef9:~/workspace$ ls -ltra
total 12
drwxr-xr-x  3 jenkins jenkins 4096 Mar  4 06:47 .
drwxr-xr-x  2 jenkins jenkins 4096 Mar  4 06:47 my_first_job
drwxr-xr-x 14 jenkins jenkins 4096 Mar  4 06:48 ..
jenkins@9715ad345ef9:~/workspace$ pwd
/var/jenkins_home/workspace
jenkins@9715ad345ef9:~/workspace$ 
```

Whenever you create a build, it will have its own workspace, with a folder of the job name.

In the home folder you have some other files that you can troubleshoot like plugins, secrets etc.,

```
jenkins@9715ad345ef9:~/workspace/my_first_job$ cd ~
jenkins@9715ad345ef9:~$ pwd
/var/jenkins_home
jenkins@9715ad345ef9:~$ ls -ltra
total 256
drwxr-xr-x  1 root    root    4096 Feb  5 13:33 ..
drwxr-xr-x 10 jenkins jenkins 4096 Mar  3 15:07 war
drwxr-xr-x  4 jenkins jenkins 4096 Mar  3 15:07 .cache
drwxr-xr-x  3 jenkins jenkins 4096 Mar  3 15:07 .java
-rw-r--r--  1 jenkins jenkins   64 Mar  3 15:07 secret.key
-rw-r--r--  1 jenkins jenkins     0 Mar  3 15:07 secret.key.not-so-secret
-rw-r--r--  1 jenkins jenkins  171 Mar  3 15:07 jenkins.telemetry.Correlator.xml
drwxr-xr-x  3 jenkins jenkins 4096 Mar  3 15:07 .groovy
-rw-----  1 jenkins jenkins 1680 Mar  3 15:07 identity.key.enc
-rw-r--r--  1 jenkins jenkins   370 Mar  3 15:07 hudson.plugins.git.GitTool.xml
drwxr-xr-x  2 jenkins jenkins 4096 Mar  3 15:07 userContent
drwxr-xr-x 119 jenkins jenkins 28672 Mar  3 15:19 plugins
drwxr-xr-x  2 jenkins jenkins 4096 Mar  3 15:19 updates
```

Let's start with a new Job:

New Item

Enter an item name

my_python_job

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



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.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

OK

Add the Git repository, for public repository no need for credentials, if your using private repository add the credentials.

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

None

Git [?](#)

Repositories [?](#)

Repository URL [?](#)

`https://github.com/manasi09-pd/DOCKER.git`



Credentials [?](#)

- none -



+ Add

Advanced [▼](#)

If the Jenkins docker container does not have python3 installed on it, you can install it while switching to root user and execute the docker container and run the commands.

- `docker exec -u root -it 9715ad345ef9 /bin/bash`
- `apt update`
- `apt install -y python3`

Write the shell command to execute the script and mention the script name.

With Ant [?](#)

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Execute shell [?](#)



Command

See [the list of available environment variables](#)

`python3 helloworld.py`

```

Started by user brad
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/my_python_job
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/devopsjourney1/jenkins-101
> git init /var/jenkins_home/workspace/my_python_job # timeout=10
Fetching upstream changes from https://github.com/devopsjourney1/jenkins-101
> git --version # timeout=10
> git --version # 'git version 2.30.2'
> git fetch --tags --force --progress -- https://github.com/devopsjourney1/jenkins-101 +refs/heads/*:refs/remotes/origin/*
> git config remote.origin.url https://github.com/devopsjourney1/jenkins-101 # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/*
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/*
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision ec8b502be2c3632538b5121b1715aac2ed36fd1d (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f ec8b502be2c3632538b5121b1715aac2ed36fd1d # timeout=10
Commit message: "Update readme.md"
First time build. Skipping changelog.
[my_python_job] $ /bin/sh -xe /tmp/jenkins14093174226420022649.sh
+ python3 helloworld.py
Hello world!
Finished: SUCCESS

```

Setting up a cloud Agent:

The new node is basically a new Linux server or windows sort of always available as a permanent agent via SSH, it's a older method new way to set a cloud agent like Docker, Kubernetes and Amazon EC2. Install the plugin Docker.

Clouds

There is no plugin installed that supports clouds.

[Install a plugin](#)

[Learn more about distributed builds](#)

Jenkins

Dashboard > Manage Jenkins > Nodes > New node

New node

Node name

Type

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

[Create](#)

Let's set up the docker cloud, First thing you need to set uri of the host that is running docker, For Jenkins 127.0.0.1:2376 and port for docker, or add the remote server IP that has the docker installed on it.

New cloud

Cloud name

Type

- Amazon EC2
- Docker
- Kubernetes

Create

Create a **new alpine container** on Jenkins Master with the command below.

➤ docker run -d --restart=always -p 127.0.0.1:2376:2375 --network jenkins -v /var/run/docker.sock:/var/run/docker.sock alpine/socat tcp-listen:2375,fork,reuseaddr unix-connect:/var/run/docker.sock

```
ubuntu@ip-172-31-39-42:~$ docker ps
CONTAINER ID IMAGE NAMES COMMAND CREATED STATUS PORTS
7d67c9f4ef02 alpine/socat gifted_germain "socat tcp-listen:23..." 24 seconds ago Up 23 seconds 127.0.0.1:2376->2375/tcp
9715ad345ef9 myjenkins-blueocean:2.492.1-1 "/usr/bin/tini -- /u..." 22 hours ago Up 9 hours 0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp, 0.0.0.0:50000->50000/tcp
ubuntu@ip-172-31-39-42:~$
```

Perform the docker inspect on the docker container. To get the IP address of the container.

```
ubuntu@ip-172-31-39-42:~$ docker inspect 7d67c9f4ef02
[
  {
    "Id": "7d67c9f4ef022399d68cda3551713d7052190521d969aa7ef7c0c78a61a22e15",
    "Created": "2025-03-04T12:41:22.842938289Z",
    "Path": "socat",
    "Args": [
      "tcp-listen:2375,fork,reuseaddr",
      "unix-connect:/var/run/docker.sock"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
```

Add the IP address to cloud agent configuration. And test the connection.

Docker Host URI ?

URI to the Docker Host you are using. May be left blank to use the Docker default (defined by DOCKER_HOST environment variable) (typically unix:///var/run/docker.sock or tcp://127.0.0.1:2376).

(from [Docker Commons Plugin](#))

Server credentials

- none -

+ Add

Advanced ▾

Version = 28.0.1, API Version = 1.48

Test Connection

Enabled ?

Error Duration ?

Save

Set up the docker agent template.

Cloud docker Configuration

Name ?

Docker Cloud details ▾ Edited

Docker Agent templates ▾

Save

Apply

Labels are added so that the master decides which agent should be selected for a particular build. I'm using the image from Docker Hub. If you want to use your own image you can point it to the docker hub or set up a private repository.

Instance capacity is how many agents you want to spin up. **Remote File System Root** is where your root folder gets created.

Instance Capacity ?

Remote File System Root ?

Usage ?

Docker-agent-alpine

Enabled ?

Name ?
Docker-agent-alpine

Docker Image ?
jenkins/agent:alpine-jdk11

Registry Authentication ▾

Container settings ▾

Configuring a job with the agent we just created and trigger new the build.

my_first_job > Configuration

Execute concurrent builds if necessary ?

Restrict where this project can be run ?

Label Expression ?
Docker-agent-alpine

Label Docker-agent-alpine matches no nodes and 1 cloud. Permission

Advanced ▾

#5
All nodes of label 'Docker-agent-alpine' are offline

The agent is not started so build is stuck, let's troubleshoot it by going to the nodes and triggering the launch agent.

Dashboard > Nodes > Docker-agent-alpine-0000bwmbx7wls > Log

Status
Delete Agent
Configure
Build History
Load Statistics
 Log
Open Blue Ocean

```
Connecting to docker container 8d313d553b897d342debe296c4d7d6b8b7423d715b81dc40f59f37089e5b76e4, running command java -jar /home/jenkins/remoting-3283.v92c105ef819.jar -noReconnect -noKeepAlive -agentLog /home/jenkins/agent.log
HTTP/1.1 101 UPGRADED
Content-Type: application/vnd.docker.raw-stream
Connection: Upgrade
Upgrade: tcp
Api-Version: 1.48
Docker-Experimental: false
Ostype: linux
Server: Docker/28.0.1 (linux)
```

Launched the agent and the build used the agent docker-agent-alpine.

Console Output

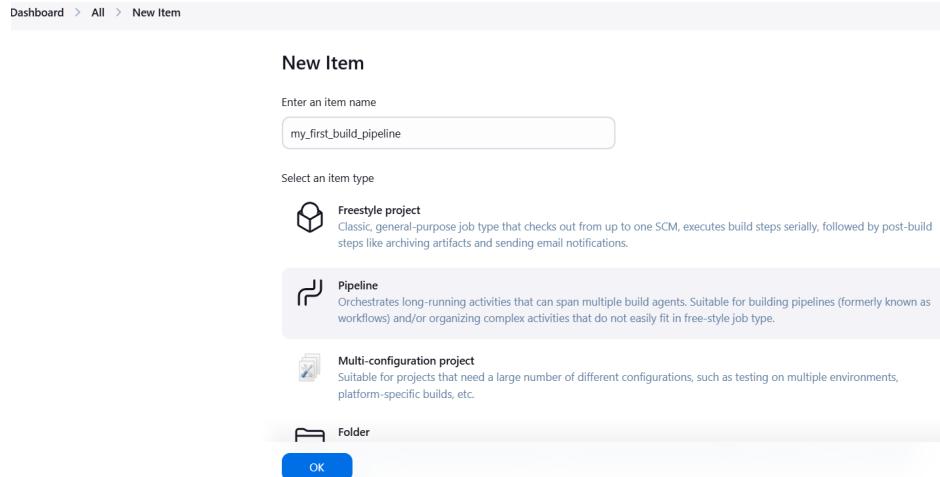
```
Started by user brad
Running as SYSTEM
Building remotely on docker-agent-alpine-00025dr47dizy_on_docker (docker-agent-alpine) in workspace /home/jenkins/workspace/my_first_job
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[my_first_job] $ /bin/sh -xe /tmp/jenkins16400106177649063182.sh
+ echo 'Hello World'
Hello World
+ echo 'The build ID of this job is ?'
The build ID of this job is 7
+ echo 'The build URL is http://127.0.0.1:8080/job/my_first_job/7/'
The build URL is http://127.0.0.1:8080/job/my_first_job/7/
+ ls -ltr
total 8
+ echo 1234
+ ls -ltr
total 4
-rw-r--r-- 1 jenkins jenkins 5 Jun 8 01:27 test.txt
Finished: SUCCESS
```

Setting the automatic triggers in the builds: **Poll SCM** is the method to monitor your GitHub repository check for any changes. It's better to use this method if your Jenkins is set up against a firewall like Cron Job it would run when a change is made.

The screenshot shows the Jenkins job configuration page for 'my_first_job'. Under the 'Triggers' section, the 'Poll SCM' option is selected (indicated by a checked checkbox). The 'Schedule' field contains the cron expression */5 * * * *. A warning message at the bottom states: '⚠ Spread load evenly by using '*/5 * * * *' rather than '**/5 * * * *'. Would last have run at Tuesday, March 4, 2025 at 6:00:06 PM Coordinated Universal Time; would next run at Tuesday, March 4, 2025 at 6:05:06 PM Coordinated Universal Time.'

Setting CI/CD pipeline in a Declarative pipeline:

In the configuration you don't have a lot of options, you can define your pipeline script in the Jenkins UI page, or you can use Jenkins file on your source repository. Point the pipeline over to the Jenkins file and it will take it there.



Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script

```

1 > pipeline {
2 >     agent {
3 >         node {
4 >             label 'jenkins-agent-goes-here'
5 >         }
6 >     }
7 >     stages {
8 >         stage('Build') {
9 >             steps {
10 >                 echo "Building.."
11 >                 sh ...
12 >                 echo "doing build stuff.."
13 >             }
14 >         }
15 >         stage('Test') {
16 >             steps {
17 >

```

try sample Pipeline...

I have written a declarative pipeline in groovy language, which is easy to collaborate and write.

Setting up a new docker agent for python I have pulled the image of python from docker hub to Jenkins server and built the image into the container using the docker file for python.

```

ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker/python$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
devopsjourney1/myjenkinsagents  python    3093cc90c354  54 seconds ago  198MB
myjenkins-blueocean  2.492.1-1  ecbd36856e47  2 days ago   811MB
alpine/socat        latest    5c51ae7b8113  4 days ago   9.1MB
jenkins/agent       alpine-jdk11 eb03a60b3dba  5 months ago  134MB
devopsjourney1/myjenkinsagents  <none>   fe573725a2e4  2 years ago   196MB
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker/python$ docker pull devopsjourney1/myjenkinsagents:python
python: Pulling from devopsjourney1/myjenkinsagents
Digest: sha256:b8aa6f53acd4ee8a304fcc93f6a667d5acda02df977647f2cd5e15e3854d7941
Status: Downloaded newer image for devopsjourney1/myjenkinsagents:python
docker.io/devopsjourney1/myjenkinsagents:python

```

```
ubuntu@ip-172-31-39-42:~/DOCKER/jenkins-docker/python$ docker build -t devopsjourney1/myjenkinsagents:python .
[+] Building 9.8s (7/7) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 134B
=> [internal] load metadata for docker.io/jenkins/agent:alpine-jdk11
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/jenkins/agent:alpine-jdk11
=> [2/3] RUN apk add python3
=> [3/3] RUN apk add py3-pip
=> exporting to image
=> => exporting layers
=> => writing image sha256:3093cc90c354fbe3df41b05dc2fa3b07c0c5b764e151128cfebe4ed6029ebb83
```

Added one more docker agent on the docker cloud.

The screenshot shows the Jenkins Docker Cloud configuration page for a new agent. The agent is named "docker-agent-python". It is enabled and uses the Docker image "devopsjourney1/myjenkinsagents:agent". The "Container settings" dropdown is open. The "Instance Capacity" is set to 2. The "Remote File System Root" field is empty.

Enabled

Name

Docker Image

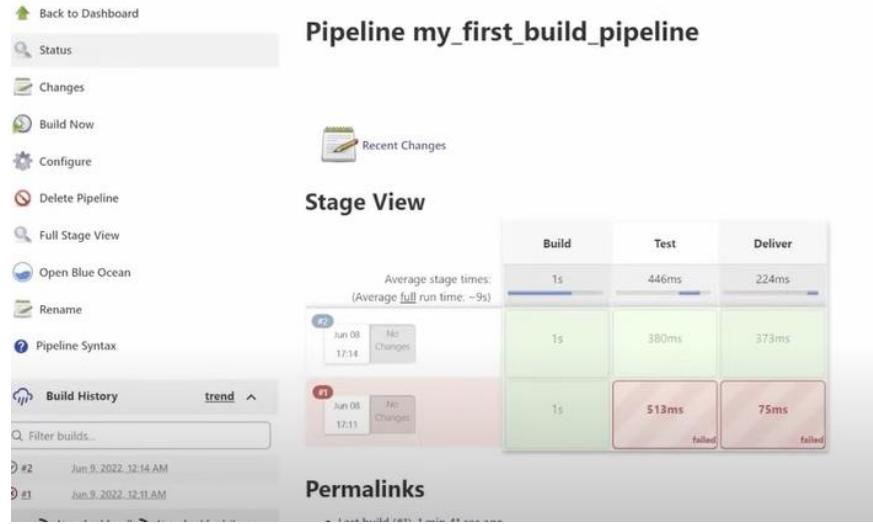
Registry Authentication

Container settings

Instance Capacity

Remote File System Root

After setting up the docker agent, run the build pipeline. The pipeline looks quite different, like it's separated into three different stages, like **BUILD**, **TEST** and **DELVIER**. Test and deliver stage failed you can check the logs. If the Test stage failed it's an upstream stage so the deliver stage failed, once this is fixed all the stages are successfully built.



Let's use a Jenkins file to run a pipeline job:

First let's build a Jenkins file with different stages like mentioning the Agent, Poll how frequently the job will be triggered, then stages like build, test and deliver.

```
test.py  X  Jenkinsfile ●
Users > manasi.deshmukh > Jenkinsfile
pipeline {
    agent {
        node {
            label 'docker-agent-python'
        }
    }
    triggers {
        pollSCM '*/* * * * *'
    }
    stages {
        stage('Build') {
            steps {
                echo "Building.."
                sh ''
                cd myapp
                pip install -r requirements.txt
                ...
            }
        }
        stage('Test') {
            steps {
                echo "Testing.."
                sh ''
                cd myapp
                python3 hello.py
                python3 hello.py --name=Brad
                ...
            }
        }
    }
}
```

```
        }
    stage('Test') {
        steps {
            echo "Testing.."
            sh ''
            cd myapp
            python3 hello.py
            python3 hello.py --name=Brad
            ...
        }
    }
    stage('Deliver') {
        steps {
            echo 'Deliver....'
            sh ''
            echo "doing delivery stuff.."
            ...
        }
    }
}
```

Changing my first_build_pipeline pipeline script to SCM.

my_first_build_pipeline > Configuration

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
1 pipeline {
2     agent {
3         node {
4             label 'Docker-agent-alpine'
5         }
6     }
7     stages {
8         stage('Build') {
9             steps {
10            echo "Building.."
11            sh ''
12            echo "doing build stuff.."
13        }
14    }
15    stage('Test') {
16        steps {
17    }
```

SCM ?

Git

Repositories ?

Repository URL ?

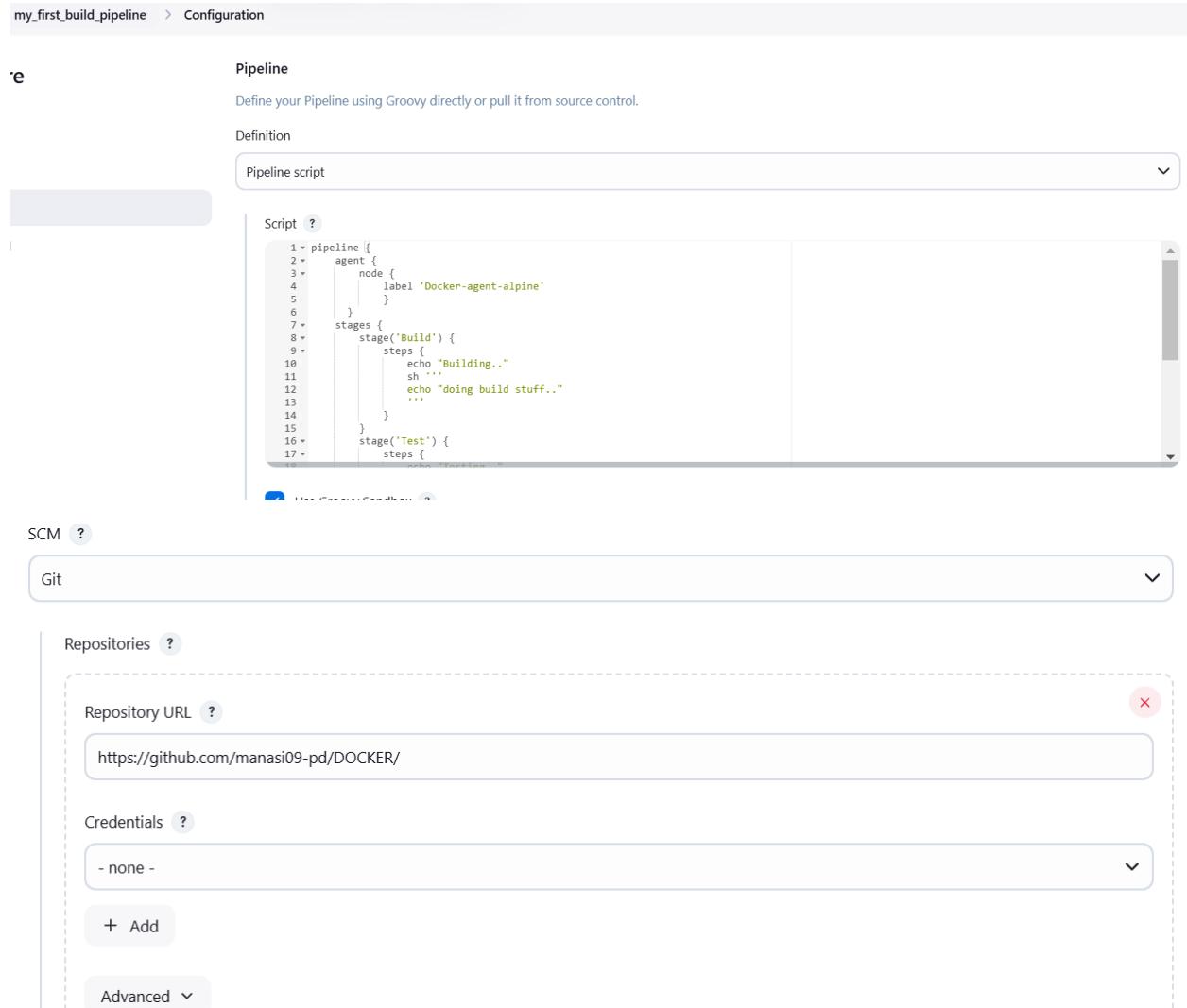
https://github.com/manasi09-pd/DOCKER/

Credentials ?

- none -

+ Add

Advanced ▾



Every 5 minutes the job will run, it added one more stage that is git checkout since it will take the pipeline script from SCM, will add the git checkout stage.

The screenshot shows the Jenkins interface for the pipeline 'my_first_build_pipeline'. On the left, there's a sidebar with various options like Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, Pipeline Syntax, Git Polling Log, and Build History. The Build History section shows two builds: #4 (Jun 9, 2022, 12:47 AM) and #3 (Jun 9, 2022, 12:36 AM). The main area is titled 'Pipeline my_first_build_pipeline' and 'Stage View'. It displays a table of stage times:

Declarative: Checkout SCM	Build	Test	Deliver
6s	582ms	357ms	345ms

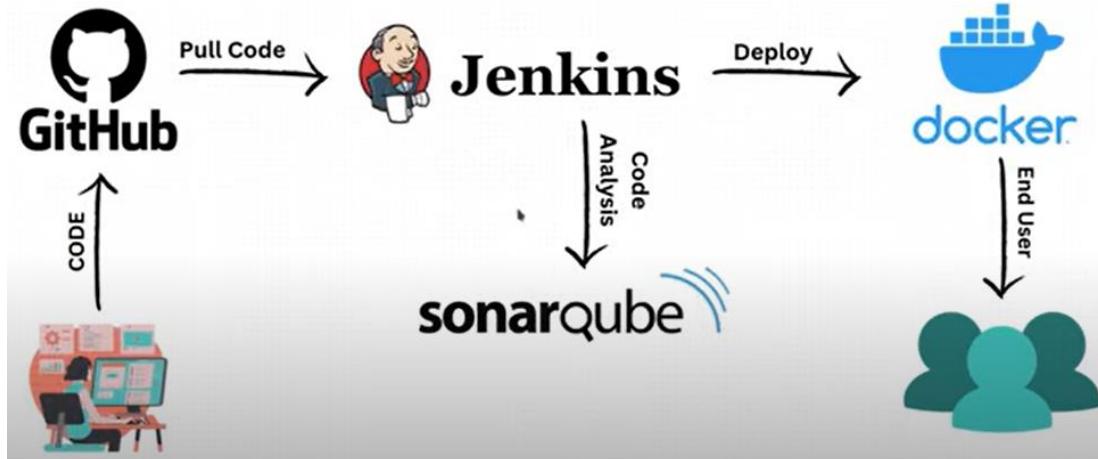
Average stage times: (Average full run time: ~12s)

Permalinks: Last build (#4), 1 min 11 sec ago.

PROJECT

In the below CI/CD pipeline Jenkins will pull the code from the GitHub and we will test the code on SonarQube which is a static code analysis tool where you can scan the vulnerabilities of the code also, we can add the rules for scanning.

After the code is tested it will be deployed on the docker. For this you will need three EC2 instances for Jenkins, SonarQube and Docker. After deploying it on the docker we will verify by accessing it through browser.



I have downloaded basic html CSS template from the internet for a website. I have extracted and pushed the code to my GitHub repository, please use the link below to access the scripts for the web page.

Link: https://github.com/manasi09-pd/Jenkins_Project.git

The screenshot shows a GitHub repository interface. At the top, the URL 'github.com/manasi09-pd/Jenkins_Project/tree/main/mediplus-lite' is visible. The main area displays a list of files and folders under the 'mediplus-lite' directory. The files listed include: .., css (New CSS script), fonts (first commit), img (first commit), js (New CSS script), mail (New CSS script), 404.html (New CSS script), blog-single.html (first commit), contact.html (first commit), index.html (first commit), portfolio-details.html (New CSS script), and style.css (first commit). The 'css' folder has a note indicating it is a 'New CSS script'. The 'fonts', 'img', 'js', 'mail', '404.html', 'blog-single.html', 'contact.html', 'index.html', 'portfolio-details.html', and 'style.css' files also have notes indicating they are 'first commit'.

Create Three EC2 instances for Jenkins, SonarQube and Docker.

Jenkins EC2 instance is created, with Ubuntu OS and t2.medium storage with 2VCPU and 4 GB memory and created a new SSH key pair.

SonarQube instance is created, with Ubuntu OS, it will consume lot of memory at least have 2GB memory, select the same key.

Docker instance is created, with Ubuntu OS and t2.medium storage with 2VCPU and 4 GB memory and created a new SSH key pair.

Instances (2/3) Info						
Find Instance by attribute or tag (case-sensitive)						
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input checked="" type="checkbox"/>	Docker	i-0aea8894fb41b01da	Running View logs Logs	t2.medium	2/2 checks passed View alarms +	eu-west-1b
<input checked="" type="checkbox"/>	SonarQube	i-0b7bbc082a62d274f	Running View logs Logs	t2.medium	2/2 checks passed View alarms +	eu-west-1b
<input type="checkbox"/>	Jenkins	i-03c84af76ecb794e	Running View logs Logs	t2.medium	Initializing View alarms +	eu-west-1b

Login to your Jenkins instance with PEM key file and update the packages using the below commands.

```
>ssh -i "7-3-2025.pem" ubuntu@34.245.172.164
```

- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get install openjdk-11-jre (Java installation is necessary for Jenkins).

Follow Jenkins.io page to install Jenkins on your preferred operating system.

The screenshot shows a web browser displaying the Jenkins User Handbook. The URL in the address bar is jenkins.io/doc/book/installing/linux/#debianubuntu. The page content is as follows:

You need to choose either the Jenkins Long-term Support release or the Jenkins weekly release.

Long Term Support release

A LTS (Long-Term Support) release is chosen every 12 weeks from the stream of regular releases as the stable release for that time period. It can be installed from the [debian-stable apt repository](#).

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

Weekly release

A new release is produced weekly to deliver bug fixes and features to users and plugin developers. It can be installed from the [debian apt repository](#).

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

```
ubuntu@ip-172-31-40-239:~$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
--2025-03-07 18:23:45-- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 199.232.26.133, 2a04:4e42:43::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|199.232.26.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins-keyring.asc    100%[=====] 0.00/0.00
```

Allow default port 8080 to access Jenkins web page.



Or run Jenkins on an EC2 instance with Docker, follow the commands given below.

Install the Docker packages.

- sudo apt-get upgrade
- sudo apt-get update
- sudo apt-get install docker.io -y
- sudo systemctl start docker
- sudo usermod -a -G docker \$(whoami)

Next, let's run Jenkins using Docker:

- docker network create Jenkins
- docker run --name jenkins-docker --rm --detach \
--privileged --network jenkins --network-alias docker \
--env DOCKER_TLS_CERTDIR=/certs \
--volume jenkins-docker-certs:/certs/client \
--volume jenkins-data:/var/jenkins_home \
--publish 2376:2376 \
docker:dind --storage-driver overlay2

Create a Docker file and build the docker images and run the docker container, Follow the Jenkins page.

Link: <https://www.jenkins.io/doc/book/installing/docker/>

The screenshot shows two windows. The top window is a terminal session on an Ubuntu system (Ubuntu@ip-172-31-43-130) displaying the command 'docker ps'. It lists three containers: 'myjenkins-blueocean' (status: Up 9 seconds), 'jenkins-blueocean' (status: Up 3 minutes), and 'jenkins-docker' (status: Up 3 minutes). The bottom window is a web browser showing the Jenkins dashboard. The address bar shows 'Not secure 34.251.24.129:8080'. The dashboard header says 'Jenkins' and shows 'Dashboard'. On the left sidebar, there are links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area has a heading 'Welcome to Jenkins!' with the subtext 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' Below this is a section titled 'Start building your software project' with a 'Create a job' button and a '+' icon. At the bottom, there are sections for 'Build Queue' (0 builds in the queue), 'Build Executor Status' (0/2), 'Set up a distributed build', and 'Set up an agent'.

Next, we will pull the code from Jenkins and test it on SonarQube instance if it is fine we will deploy it on the docker instance.

Create pipeline in Jenkins.

New Item

Enter an item name

Automated-Pieline

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as

Added my git repository in the source code management of Jenkins.

The screenshot shows the Jenkins 'Git' configuration page for a repository. The 'Repository URL' field contains the value `https://github.com/manasi09-pd/Jenkins_Project.git`. The 'Branch Specifier (blank for 'any')' field contains the value `*/main`. Other fields shown include 'Credentials' (set to '- none -'), 'Advanced' options, and buttons for 'Add Repository' and 'Branches to build'.

Also select the trigger, whenever there is a change in the repository the build gets triggered.

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

Add the webhook from the GitHub account, add the Jenkins URL.

The screenshot shows the GitHub project settings page for 'Jenkins_Project'. The 'Webhooks' tab is selected. On the left, there's a sidebar with 'General', 'Access', 'Collaborators', 'Moderation options', 'Code and automation' (with 'Branches', 'Tags', 'Rules', 'Actions'), and 'Webhooks' (which is highlighted). The main area has a heading 'Webhooks / Add webhook'. It says: '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](#)'. There are fields for 'Payload URL *' (containing 'http://34.251.24.129:8080/github-webhook/'), 'Content type *' (set to 'application/x-www-form-urlencoded'), and 'Secret' (empty). A red box highlights the payload URL field.

Select webhook triggers like pushes and pull requests.

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 delete

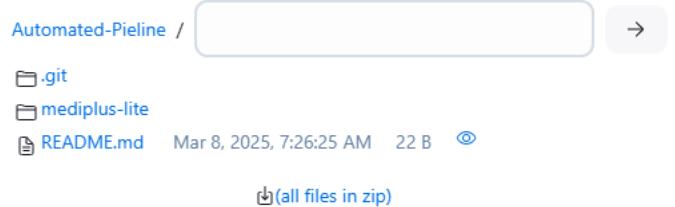
Manually triggered the build.

The screenshot shows the Jenkins console output for a build. The top navigation bar includes 'Dashboard', 'Automated-Pipeline', '#1', and 'Console Output'. The left sidebar has links for 'Status', 'Changes', 'Console Output' (which is selected), 'Edit Build Information', 'Delete build #1', 'Timings', and 'Git Build Data'. The main area is titled 'Console Output' with a green checkmark icon. It shows the Jenkins pipeline starting, cloning a repository from GitHub, and performing a successful build. The output text is as follows:

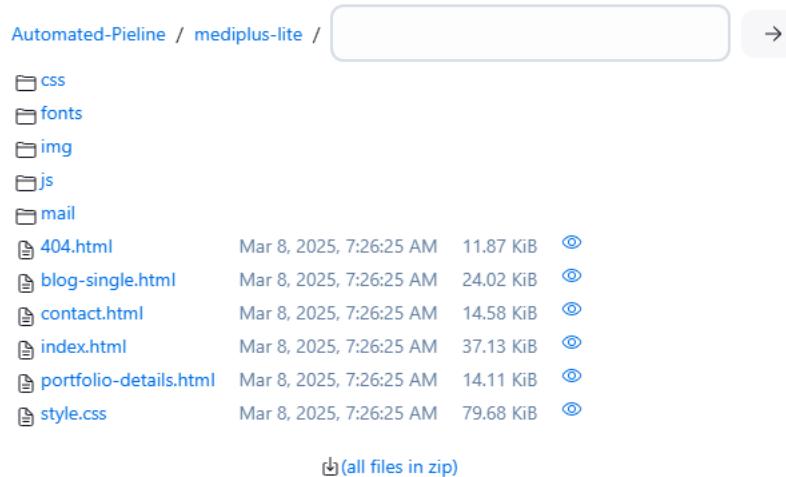
```
Started by user Manasi Deshmukh
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/Automated-Pipeline
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
  Cloning repository https://github.com/manasi09-pd/Jenkins_Project.git
    > git init /var/jenkins_home/workspace/Automated-Pipeline # timeout=10
    > git --version # timeout=10
    > git version 2.39.5'
    > git fetch --tags --force --progress -- https://github.com/manasi09-pd/Jenkins_Project.git +refs
    > git config remote.origin.url https://github.com/manasi09-pd/Jenkins_Project.git # timeout=10
    > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/*
    Avoid second fetch
    > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision fe6ab535770dc646c8e1aa9ca70e6368fff53382 (refs/remotes/origin/main)
  > git config core.sparsecheckout # timeout=10
  > git checkout -f fe6ab535770dc646c8e1aa9ca70e6368fff53382 # timeout=10
Commit message: "New CSS script"
First time build. Skipping changelog.
Finished: SUCCESS
```

I have two folders, one is .git folder and other one is CSS folder, in that I have multiple files and folders.

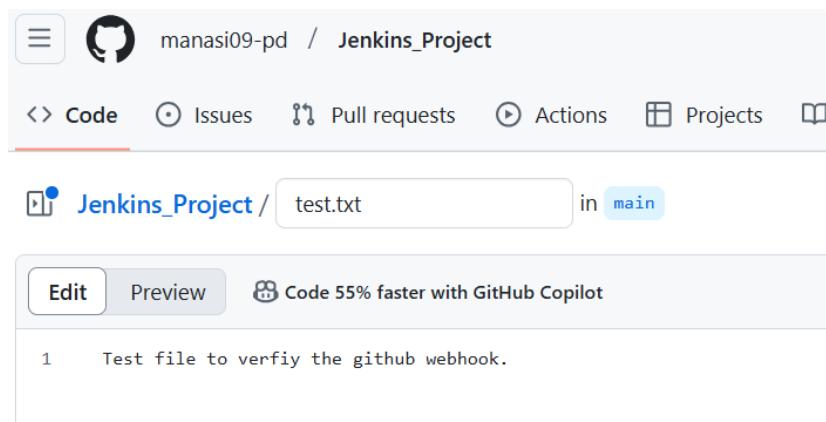
Workspace of Automated-Pipeline on Built-In Node



Workspace of Automated-Pipeline on Built-In Node



I have created a test file in GitHub to check if the build gets triggered for new push.



As you can see, the pipeline is triggered. So whenever the developer changes the code, adds a new file or any changes pushed the Jenkins build is automatically triggered from webhook.

The screenshot shows the Jenkins dashboard for an 'Automated-Pipeline' project. On the left, a sidebar lists project management options: Status, Changes, Workspace, Build Now, Configure, Delete Project, GitHub Hook Log, and Rename. Below this is a 'Builds' section with a 'Filter' input field. Under 'Today', two builds are listed: #2 at 7:37 AM and #1 at 7:26 AM, both marked with green checkmarks.

Login to SonarQube and Jenkins for instance and change the hostname. Update the repository and for SonarQube to be installed on the instance we need java 11 with the below command.

- sudo apt-get install openjdk-17-jre

```
ubuntu@ip-172-31-34-61:~$ sudo hostnamectl set-hostname sonarqube
ubuntu@ip-172-31-34-61:~$ /bin/bash
ubuntu@sonarqube:~$ 
```

- Install PostgreSQL

```
➤ curl https://www.postgresql.org/media/keys/ACCC4CF8.asc | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/apt.postgresql.org.gpg >/dev/null
➤ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
➤ sudo apt update
➤ sudo apt install postgresql postgresql-contrib
➤ sudo systemctl status postgresql
➤ sudo -u postgres psql
➤ postgres=# CREATE ROLE sonaruser WITH LOGIN ENCRYPTED PASSWORD 'your_password';
➤ postgres=# CREATE DATABASE sonarqube;
➤ postgres=# \q
```

```
➤ exit
```

Install sonarqube

- wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.6.1.59531.zip>
- wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.6.1.59531.zip>

```
ubuntu@sonarqube:~$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.6.1.59531.zip
--2025-03-08 10:47:22-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.6.1.59531.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 18.66.171.117, 18.66.171.63, 18.66.171.104, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|18.66.171.117|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 290033351 (277M) [binary/octet-stream]
Saving to: 'sonarqube-9.6.1.59531.zip'

sonarqube-9.6.1.59531.zip          100%[=====] 290033351/290033351

2025-03-08 10:47:31 (31.8 MB/s) - 'sonarqube-9.6.1.59531.zip' saved [290033351/290033351]

ubuntu@sonarqube:~$ ls
sonarqube-9.6.1.59531.zip
```

- unzip -q sonarqube-9.6.1.59531.zip

Move the files to the /opt/sonarqube directory.

- sudo mv sonarqube-9.6.1.59531 /opt/sonarqube

```
ubuntu@sonarqube:~$ ls
sonarqube-9.6.1.59531  sonarqube-9.6.1.59531.zip
ubuntu@sonarqube:~$ sudo mv sonarqube-9.6.1.59531 /opt/sonarqube
```

IN the /opt/sonarqube path there is bin folder which has options for Linux, windows and MacOS, since I have Linux instance we execute the which provide various options like start, stop, reset.

I have used the command console to check the logs

- ./sonar.sh console

```
ubuntu@sonarqube:/opt/sonarqube/bin/linux-x86-64$ ./sonar.sh
/usr/bin/java
Usage: ./sonar.sh { console | start | stop | force-stop | restart | status | dump }
```

Edit the security groups by adding the port 9000 for the SonarQube with 0.0.0.0/0 which will accept all the IP addresses.

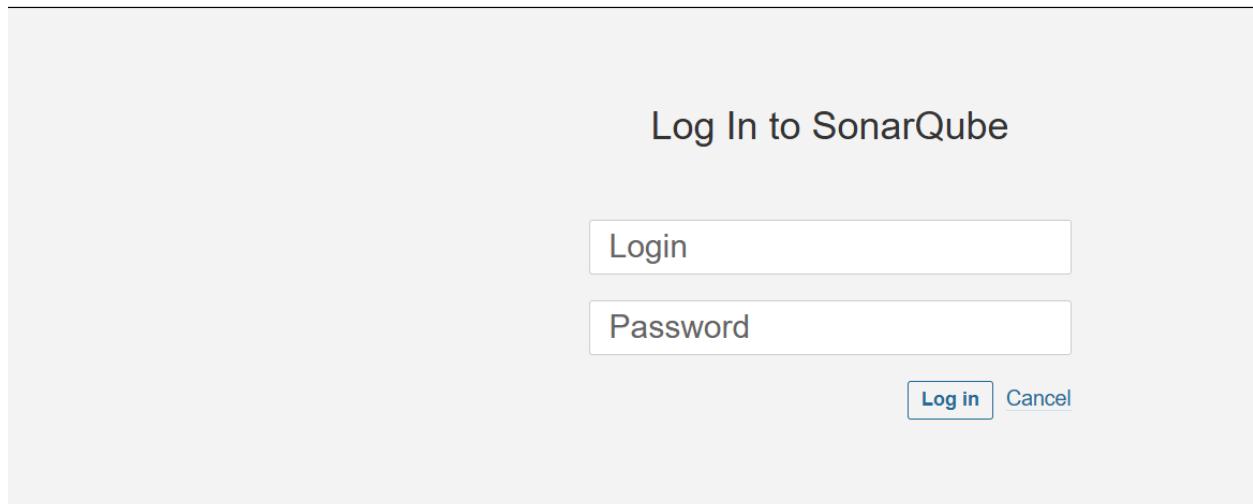
Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules <small>Info</small>						Description - optional <small>Info</small>	
Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description		
sgr-0758875befdd1031a	SSH	TCP	22	Custom	0.0.0.0/0	Delete	
-	Custom TCP	TCP	9000	Anywhere	0.0.0.0/0	Delete	

Try to access the SonarQube web page with our public IP address. Default login is admin/admin then set the new password.

 Not secure 54.74.61.134:9000/sessions/new?return_to=%2F



Logged into SonarQube page, selected the manual project.

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform. First, you need to set up a DevOps platform configuration.

 From Azure DevOps Set up global configuration	 From Bitbucket Set up global configuration	 From GitHub Set up global configuration	 From GitLab Set up global configuration
---	--	---	---

Are you just testing or have an advanced use-case? Create a project manually.


Manually

Given a name for the project, same as well as for the key.

Create a project

All fields marked with * are required

Project display name *

 ✓

Up to 255 characters. Some scanners might override the value you provide.

Project key *

 ✓

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Set Up

To analyze the repository, select the Jenkins. Select the GitHub option.

Analyze your project with Jenkins

Select your DevOps platform

[Bitbucket Cloud](#) [Bitbucket Server](#) [GitHub](#) [GitLab](#)

Configure the analysis, click on the continuation option for the pipeline. For creating the Jenkins file choose the other option and save the project key onto some text folder and at last finish the tutorial.

3 Create a Jenkinsfile

1 What option best describes your build?

[Maven](#) [Gradle](#) [.NET](#) [Other \(for JS, TS, Go, Python, PHP, ...\)](#)

2 Create a `sonar-project.properties` file in your repository and paste the following code:

`sonar.projectKey=Mediplus-project`

[Copy](#)

3 Create a `Jenkinsfile` file in your repository and paste the following code:

i Make sure to replace `SonarScanner` with the name you gave to your SonarQube Scanner tool in Jenkins. [?](#)

```
node {
    stage('SCM') {
        checkout scm
    }
    stage('SonarQube Analysis') {
        def scannerHome = tool 'SonarScanner';
        withSonarQubeEnv() {
            sh "${scannerHome}/bin/sonar-scanner"
        }
    }
}
```

[Copy](#)

[Finish this tutorial >](#)

To create a token, go to Administration > My Account > Security, If you select the project analysis token it will be used for specific project, the token will be authenticated for a particular project. The Global Analysis token will be used for all the projects on your SonarQube. Generate the token and save it.

The screenshot shows the 'Tokens' section of the SonarQube administration interface. At the top, there is a navigation bar with 'Profile', 'Security' (which is underlined), 'Notifications', and 'Projects'. Below the navigation, a green header bar indicates the user is 'Administrator'. The main content area has a heading 'Tokens' and a sub-instruction: 'If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.' A 'Generate Tokens' button is present. Below this, a table lists tokens with columns: Name, Type, Project, and Expires in. One row is shown with 'sonarqube-token' as the name, 'Project Analysis Token' as the type, 'Mediplus-project' as the project, and '30 days' as the expiration. A 'Generate' button is next to the row. Below the table, a message says 'No tokens'.

Go back to your Jenkins page, install two plugins in manage Jenkins section.

The screenshot shows the Jenkins 'Manage Jenkins' > 'Plugins' page. The search bar at the top contains 'sonar'. Under the 'Available plugins' tab, two plugins are listed: 'SonarQube Scanner' and 'SSH2'. Both have an 'Install' button. The 'SonarQube Scanner' plugin is checked for installation. A note below it says: 'This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.' and '1 mo 9 days ago'. The 'SSH2' plugin also has an 'Install' button and a note: 'This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.' and '1 yr 7 mo ago'.

Next go to manage jenkins and tools configuration. There you will see the SonarQube scanner. Add it and save it.

The screenshot shows the Jenkins 'Manage Jenkins' > 'Tools Configuration' > 'SonarQube Scanner installations' page. It has a title 'SonarQube Scanner installations' and a 'Add SonarQube Scanner' button. A configuration panel for 'SonarQube Scanner' is open. It includes a 'Name' field with 'Sonarscanner' and a checked 'Install automatically' checkbox. Below this is a 'Install from Maven Central' section with a 'Version' field containing 'SonarQube Scanner 7.0.2.4839'. A 'Add Installer' dropdown is at the bottom.

Go to the configure system, you will be able to see SonarQube servers. Add the server's name and URL of your SonarQube server.

SonarQube installations

[List of SonarQube installations](#)

The screenshot shows a configuration form for a new SonarQube installation. It includes fields for 'Name' (set to 'Sonar-server') and 'Server URL' (set to 'http://54.74.61.134:9000'). The 'Server URL' field has a note: 'Default is http://localhost:9000'.

Name	Sonar-server
Server URL	Default is http://localhost:9000 http://54.74.61.134:9000/

In the pipeline select the configuration, add the build steps. Select the “**Execute SonarQube Scanner**” option and add the **project key**.

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

The screenshot shows the 'Execute SonarQube Scanner' build step configuration. It includes fields for 'JDK' (set to '(Inherit From Job)'), 'Path to project properties' (empty), and 'Analysis properties' (containing the value 'sonar.projectKey=Mediplus-project').

Execute SonarQube Scanner	
JDK	(Inherit From Job)
Path to project properties	
Analysis properties	sonar.projectKey=Mediplus-project

Then go to manage Jenkins to add the SonarQube token, click on add in SonarQube servers. Select the kind as Secret Text and add the token generated in SonarQube.

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations Document was last saved: 4h ago

Name
Sonar-server

Server URI
Default is http://localhost:9000
http://54.74.61.134:9000/

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
- none -
+ Add

Advanced ▾

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain
Global credentials (unrestricted)

Kind
Secret text

Scope
Global (Jenkins, nodes, items, all child items, etc)

Secret
.....

ID
Sonar-token

Description

Cancel Add

Select the token generated and save.

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Sonar-token

Go to the pipeline and build the pipeline, if it's working or not. The build is running where the SonarQube is scanning the code we have received the status as success. Let's refresh the page of SonarQube and go to the project section you will find that it has scanned our code, and the code passed with no vulnerabilities, some bugs. Once it is passed, we will deploy it on the docker server.

```

ne > #3 > Console Output
11:24:07.067 INFO 1/1 source file has been analyzed
11:24:07.071 INFO No PHPUnit tests reports provided (see 'sonar.php.tests.reportPath' property)
11:24:07.072 INFO No PHPUnit coverage reports provided (see 'sonar.php.coverage.reportPaths' property)
11:24:07.072 INFO Sensor PHP sensor [php] (done) | time=199ms
11:24:07.072 INFO Sensor Analyzer for "php.ini" files [php]
11:24:07.075 INFO Sensor Analyzer for "php.ini" files [php] (done) | time=3ms
11:24:07.076 INFO Sensor Text Sensor [text]
11:24:07.076 INFO 20 source files to be analyzed
11:24:07.160 INFO 20/20 source files have been analyzed
11:24:07.160 INFO Sensor Text Sensor [text] (done) | time=84ms
11:24:07.161 INFO Sensor VB.NET Project Type Information [vbnet]
11:24:07.161 INFO Sensor VB.NET Project Type Information [vbnet] (done) | time=1ms
11:24:07.162 INFO Sensor VB.NET Analysis Log [vbnet]
11:24:07.173 INFO Sensor VB.NET Analysis Log [vbnet] (done) | time=11ms
11:24:07.173 INFO Sensor VB.NET Properties [vbnet]
11:24:07.173 INFO Sensor VB.NET Properties [vbnet] (done) | time=0ms
11:24:07.178 INFO -----
11:24:07.241 INFO Sensor Analysis Warnings Import [csharp]
11:24:07.242 INFO Sensor Analysis Warnings Import [csharp] (done) | time=1ms
11:24:07.242 INFO Sensor Zero Coverage Sensor
11:24:07.252 INFO Sensor Zero Coverage Sensor (done) | time=18ms
11:24:07.254 INFO SCM Publisher SCM provider for this project is: git
11:24:07.256 INFO SCM Publisher 20 source files to be analyzed
11:24:07.735 INFO SCM Publisher 20/20 source files have been analyzed (done) | time=478ms
11:24:07.751 INFO CPD Executor Calculating CPD for 6 files
11:24:07.815 INFO CPD Executor CPD calculation finished (done) | time=63ms
11:24:07.893 INFO Analysis report generated in 76ms, dir size=792.0 kB
11:24:07.945 INFO Analysis report compressed in 52ms, zip size=198.6 kB
11:24:08.034 INFO Analysis report uploaded in 88ms
11:24:08.036 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://54.74.61.134:9000/dashboard?id=Mediplus-project
11:24:08.036 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
11:24:08.037 INFO More about the report processing at http://54.74.61.134:9000/api/ce/task?id=AZV6peenavNaJyU\_aA1
11:24:08.729 INFO Analysis total time: 10.084 s

```

Dashboard > Automated-Pipeline >

 Status  **Automated-Pipeline**

 Changes

 Workspace

 Build Now

 Configure

 Delete Project

 GitHub Hook Log

 Favorite

 SonarQube

 Mediplus-project Passed

server-side processing: Success

Permalinks

- Last build (#3), 4 min 12 sec ago
- Last stable build (#3), 4 min 12 sec ago
- Last successful build (#3), 4 min 12 sec ago
- Last completed build (#3), 4 min 12 sec ago

 Open Blue Ocean

 Rename

Builds

 Filter

Today  #3 11:23 AM

The screenshot shows the SonarQube interface with the following details:

- Projects:** Mediplus-project
- Status:** Passed
- Bugs:** 132 (B)
- Vulnerabilities:** 0 (A)
- Hotspots Reviewed:** 0.0% (E)
- Code Smells:** 0 (A)
- Coverage:** 0.0%
- Duplications:** 11.7%
- Lines:** 9.1K (S) CSS, HT...

Connect to our three EC2 instances where we will be installing docker.

Install docker of the EC2 instance of ubuntu by following the official docker page.

Link: <https://docs.docker.com/engine/install/ubuntu/>

```
ubuntu@docker:~$ docker --version
Docker version 28.0.1, build 068a01e
```

Login to Jenkins server and execute the docker instance with Jenkins user and try to access the docker instance from Jenkins docker instance, where I faced permission denied error.

```
ubuntu@jenkins:~$ docker exec -it 0fe749c7e5f0 /bin/bash
jenkins@0fe749c7e5f0:/$ ssh ubuntu@34.240.100.90
ubuntu@34.240.100.90: Permission denied (publickey).
jenkins@0fe749c7e5f0:/$ 
```

To solve the problem, go to docker instance and edit the sshd_config file.

```
ubuntu@docker:~$ sudo su
root@docker:/home/ubuntu# vi /etc/ssh/sshd_config
```

Uncomment on the below line

```
#MaxSessions 10
#
#PubkeyAuthentication yes
#
# To disable tunneled clear text passwords, change
#PasswordAuthentication yes
#PermitEmptyPasswords no
```

Restart the ssh services

```
root@docker:/home/ubuntu# systemctl restart ssh
Warning: The unit file, source configuration file or drop-in
root@docker:/home/ubuntu# systemctl daemon-reload
```

I am now asking for ubuntu user password.

```
jenkins@0fe749c7e5f0:/$ ssh ubuntu@34.240.100.90
(ubuntu@34.240.100.90) Password: 
```

Changed the ubuntu password

```
root@docker:/home/ubuntu# passwd ubuntu
New password:
Retype new password:
passwd: password updated successfully
root@docker:/home/ubuntu#
```

Successfully logged into the docker machine from Jenkins machine.

```
ubuntu@jenkins:~$ docker exec -it 0fe749c7e5f0 /bin/bash
jenkins@0fe749c7e5f0:/$ ssh ubuntu@34.240.100.90
(ubuntu@34.240.100.90) Password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1021-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Sun Mar  9 15:01:32 UTC 2025

System load:  0.0          Processes:           135
Usage of /:   39.9% of 6.71GB  Users logged in:    1
Memory usage: 8%          IPv4 address for enX0: 172.31.46.1
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

137 updates can be applied immediately.
44 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun Mar  9 14:56:59 2025 from 157.45.74.194
ubuntu@docker:~$
```

Generate the ssh key in Jenkins instance and copy the key to the docker instance.

```
ubuntu@jenkins:~$ docker exec -it 0fe749c7e5f0 /bin/bash
jenkins@0fe749c7e5f0:/$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/jenkins_home/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/jenkins_home/.ssh/id_rsa
Your public key has been saved in /var/jenkins_home/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:47mSclGIyvaou5BBnbMwODea2+DYMSrvcxAPoRUEhls jenkins@0fe749c7e5f0
The key's randomart image is:
```

```

jenkins@0fe749c7e5f0:/$ ssh-copy-id ubuntu@34.240.100.90
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/jenkins_home/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
(ubuntu@34.240.100.90) Password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ubuntu@34.240.100.90'"
and check to make sure that only the key(s) you wanted were added.

jenkins@0fe749c7e5f0:/$

```

Next steps go to Jenkins web page and add the docker server in the manage Jenkins page.

Server Groups Center

Server Group List :

[Create the server groups for your projects](#)

Group Name ?

SSH Port ?

User Name ?

Password ?

Add the docker server to the group list with docker ec2 instance IP.

[Server LIST :](#)

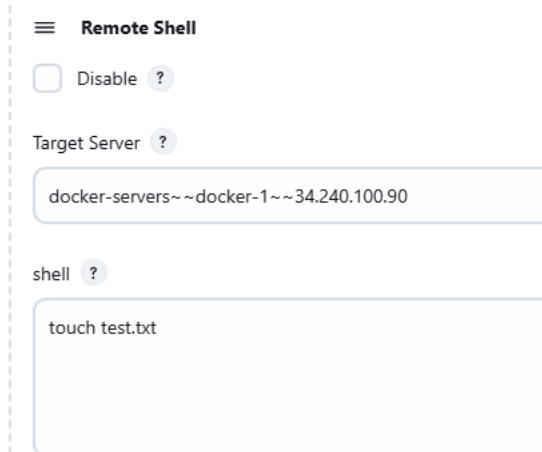
[add the server under this server group for your projects](#)

Server Group:

Server Name ?

Server IP ?

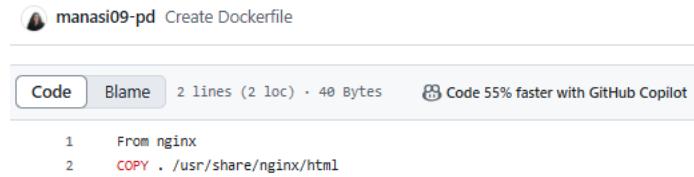
In the automation pipeline, go to configuration and at the end add the build step as Remote shell. To check if it's working added a line like touch command to create a file and triggered the build to check.



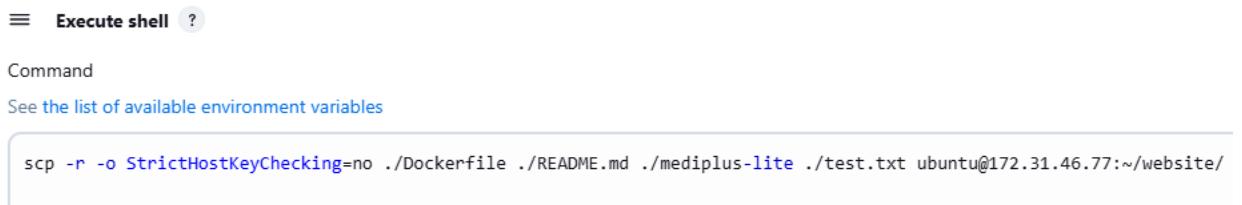
The build has run the remote command on the server.

```
touch test.txt
exit $?
ubuntu@docker:~$ touch test.txt
ubuntu@docker:~$ exit $?
logout
#####
execute command exit status -->0
#####
Finished: SUCCESS
```

Go to the repository in GitHub and add a Dockerfile. After the commit the pipeline will be automatically triggered.



Go to the pipeline and configuration, remove the remote shell and add a build step, copy the contents of the current server to docker.



Build is successful, lets verify it.

```
ubuntu@docker:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cf7edc0ef51a mywebsite "/docker-entrypoint..." 13 seconds ago Up 12 seconds 0.0.0.0:8085->80/tcp, [::]:8085->80/tcp Mediplus-Website
ubuntu@docker:~$
```

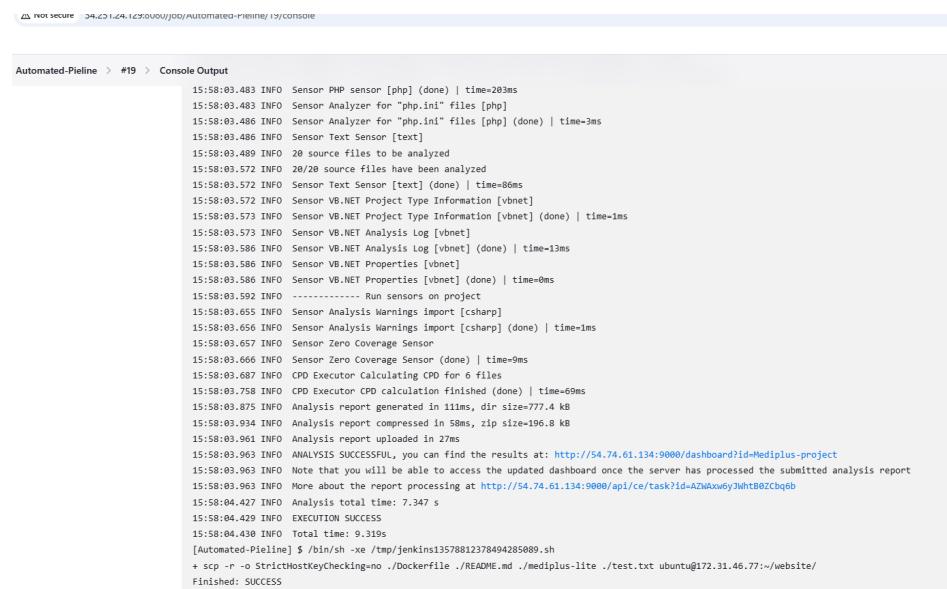
```

ubuntu@docker:~$ pwd
/home/ubuntu
ubuntu@docker:~$ mkdir website
ubuntu@docker:~$ ls
website

```

Create a website directory on docker server.

Build is successful.

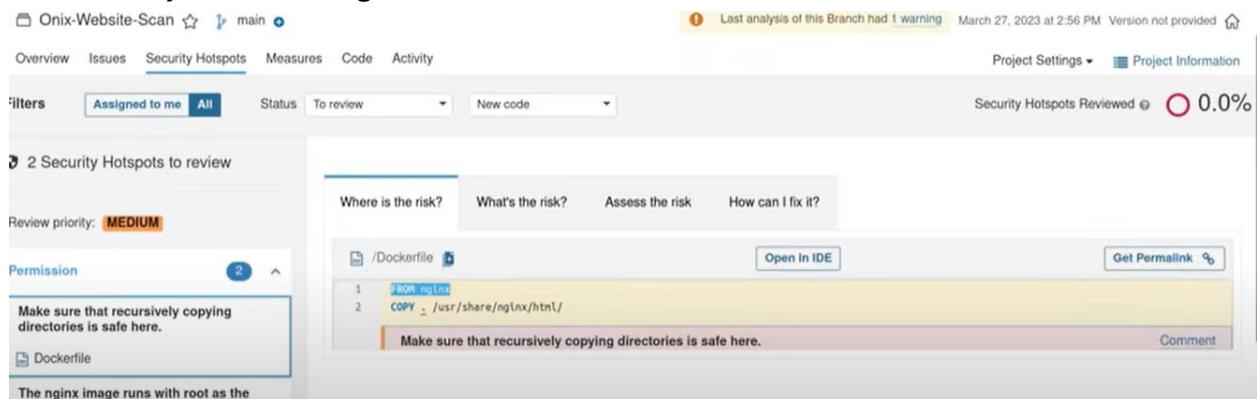


```

Automated-Pipeline > #19 > Console Output
15:58:03.483 INFO Sensor PHP sensor [php] (done) | time=203ms
15:58:03.483 INFO Sensor Analyzer for "php.ini" files [php]
15:58:03.486 INFO Sensor Analyzer for "php.ini" files [php] (done) | time=3ms
15:58:03.486 INFO Sensor Text Sensor [text]
15:58:03.489 INFO 28 source files to be analyzed
15:58:03.572 INFO 28/28 source files have been analyzed
15:58:03.572 INFO Sensor Text Sensor [text] (done) | time=86ms
15:58:03.572 INFO Sensor VB.NET Project Type Information [vbnet]
15:58:03.573 INFO Sensor VB.NET Project Type Information [vbnet] (done) | time=1ms
15:58:03.573 INFO Sensor VB.NET Analysis Log [vbnet]
15:58:03.586 INFO Sensor VB.NET Analysis Log [vbnet] (done) | time=13ms
15:58:03.586 INFO Sensor VB.NET Properties [vbnet]
15:58:03.586 INFO Sensor VB.NET Properties [vbnet] (done) | time=0ms
15:58:03.592 INFO ----- Run sensors on project
15:58:03.655 INFO Sensor Analysis Warnings Import [csharp]
15:58:03.656 INFO Sensor Analysis Warnings Import [csharp] (done) | time=1ms
15:58:03.657 INFO Sensor Zero Coverage Sensor
15:58:03.666 INFO Sensor Zero Coverage Sensor (done) | time=9ms
15:58:03.687 INFO CPD Executor Calculating CPD for 6 files
15:58:03.758 INFO CPD Executor CPD calculation finished (done) | time=50ms
15:58:03.875 INFO Analysis report generated in 11ms, dir size=777.4 kB
15:58:03.934 INFO Analysis report compressed in 58ms, zip size=196.8 kB
15:58:03.961 INFO Analysis report uploaded in 27ms
15:58:03.963 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://54.74.61.134:9000/dashboard?id=Mediplus-project
15:58:03.963 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
15:58:03.963 INFO More about the report processing at http://54.74.61.134:9000/api/ce/task?id=AZuAxu6yJhntB0ZCbq6b
15:58:04.427 INFO Analysis total time: 7.347 s
15:58:04.429 INFO EXECUTION SUCCESS
15:58:04.438 INFO Total time: 9.319s
[Automated-Pipeline] $ /bin/sh -xe /tmp/jenkins13578812378494285089.sh
+ scp -r -o StrictHostKeyChecking=no .Dockerfile ./README.md ./mediplus-lite ./test.txt ubuntu@172.31.46.77:~/website/
Finished: SUCCESS

```

Vulnerability scan is failing due to the addition of Docker file.



The screenshot shows the SonarQube interface for the 'Onix-Website-Scan' project. It displays a 'Security Hotspots' section with one item. The hotspot details are as follows:

- Review priority:** MEDIUM
- Permission:** 2
- Description:** Make sure that recursively copying directories is safe here.
- File:** Dockerfile
- Line 1:** FROM nginx
- Line 2:** COPY . /usr/share/nginx/html/
- Comment:** Make sure that recursively copying directories is safe here.

After the build is successful, check if the contents are copied to docker server.

```

ubuntu@docker:~/website$ ls
Dockerfile README.md mediplus-lite test.txt
ubuntu@docker:~/website$ 

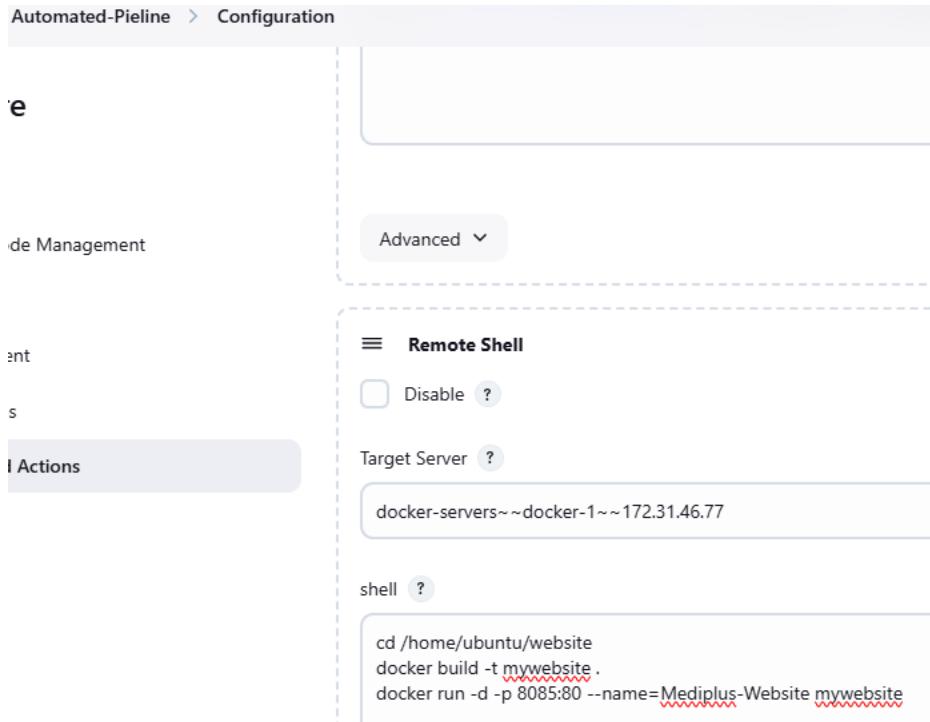
```

Next go to the pipeline and select configure and add a build step as remote shell, go to the /home/ubuntu/website path and run the docker build.

First let's check if ubuntu user can perform the docker commands, otherwise add the ubuntu to the docker group with the below command.

- Sudo usermod -aG docker \$user
- Newgrp docker

```
ubuntu@docker:~/website$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
ubuntu@docker:~/website$
```



The remote shell will go to the path and perform docker build command to build my website and then run the container in detached mode, with port 80 that is container and 8085 port of the system.

```
ubuntu@docker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
cf7edc0ef51a        mywebsite          "/docker-entrypoint..."   13 seconds ago    Up 12 seconds      0.0.0.0:8085->80/tcp, [::]:8085->80/tcp   NAMES
ubuntu@docker:~$
```

Allow port 8085 in the inbound rules of the docker instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0ada0ca9ed0f89907	SSH	TCP	22	Custom	0.0.0.0/0
-	Custom TCP	TCP	8085	Anywh...	0.0.0.0/0

The website is working but the CSS is not working.

```
108.128.126.67:8085
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

I have made some changes to the Docker file and pushed the code to GitHub and run the build same is copied on the docker instance.

```
# Use the official Nginx image as a base
From nginx:latest

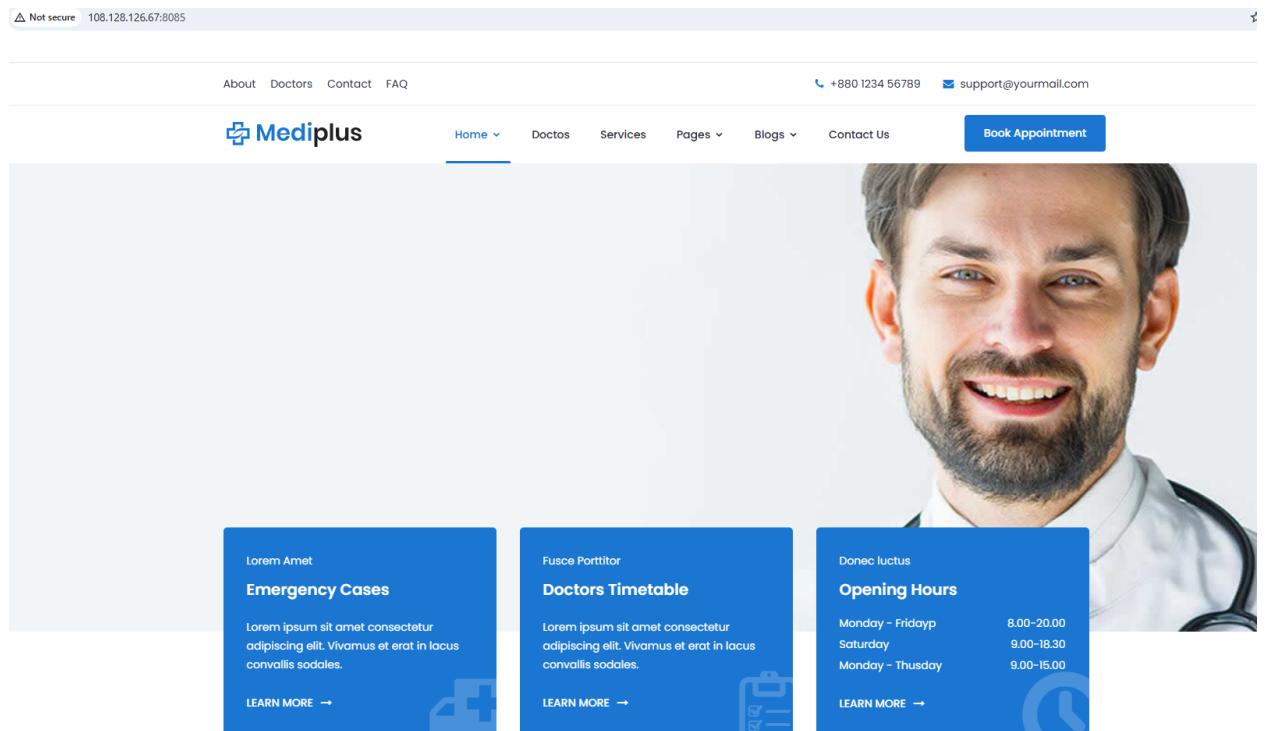
# Copy the website files into the appropriate Nginx directory
COPY ./mediplus-lite /usr/share/nginx/html

# Expose port 80 so that the website can be accessed
EXPOSE 80
~
```

Used the command in the Jenkins remote script.

- docker build -t mediplus-lite-web .
- docker run -d -p 8085:80 --name mediplus-container mediplus-lite-web

Now the website is hosted on the docker container.





Blog Categories

Men's Apparel
Women's Apparel
Bags Collection
Accessories
Sun Glasses

