

MLOPS Pipeline Jenkins :

This article covers various ways of deploying Jenkins on your server. Use our up-to-date guide for manual installation or launching Jenkins in Docker containers. As a bonus, we'll show how to deploy Jenkins using CloudFormation. Deploy Jenkins on your server in 5 minutes using our guidance.

What is Jenkins ?

Jenkins is the most popular free automation server. You can use Jenkins to automate almost any processes in your company. But the traditional market for Jenkins is the automation of the software development processes.

Jenkins benefits:

- Written in Java, and thus you can run it on all platforms
- Has lots of plugins, which can help you solve almost any problem you can imagine.
- Helps developers to integrate code changes and always produce stable software build

Install Jenkins Manually

The installation process is simple and straightforward. It consists of several easy steps:

- Java installation.
- Adding official Jenkins repositories to your package manager.
- Package installation.
- Firewall configuration.

- Initial Jenkins configuration.

Let's do them one by one.

Java installation

To be able to run Jenkins, you need to install Java first.

First, update the apt package index using this:

```
$ sudo apt update
```

Next, install the default Java OpenJDK package using this:

```
$ sudo apt install openjdk-8-jre
```

Verify the installation using this command:

```
$ java -version
```

Important: you should see that Java version

By the time of this article writing, Jenkins is using the Java 8 version. If you see another version, switch to the correct one:

```
$ sudo update-alternatives --config java
```

Add Jenkins repository

By default, Jenkins is not included in the Ubuntu repositories, and you need to add it.

First, we need to import the repository key:

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

Next step is to append the repository address to the server's repositories list:

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

Once that is done, we need to update the information about the software, which we can use from the new repository:

```
$ sudo apt-get update
```

Jenkins Package Installation

Since we have the repository up to date, let's run the following command:

```
$ sudo apt-get -y install jenkins
```

Running this command will prompt you to confirm the download and installation.

By default, Jenkins starts after installation process. If not, use the following command to do it manually:

```
$ sudo systemctl start jenkins
```

Let check if Jenkins service status:

```
$ sudo systemctl status jenkins
```

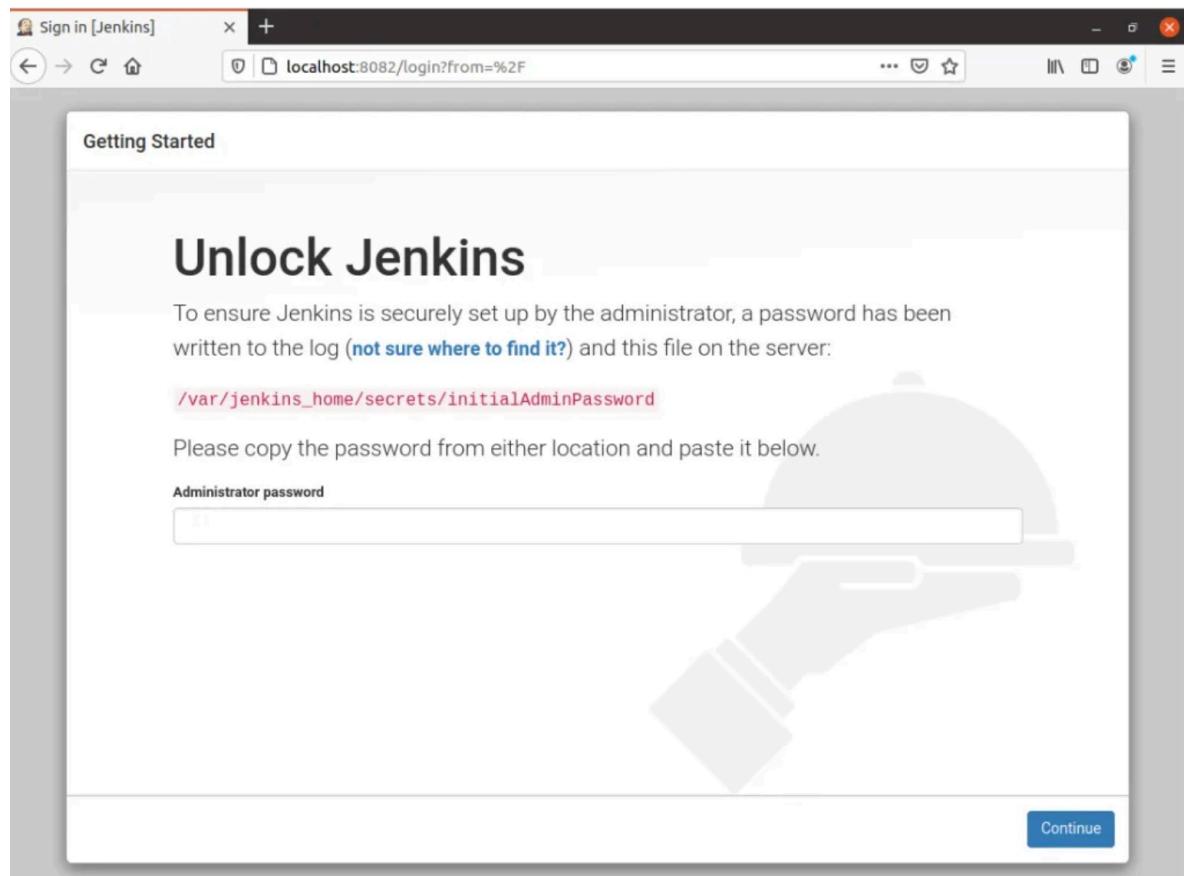
By default, Jenkins runs on port **8080**.

So this can be done through enabling security group and inbound 8080 port for TCP connection on AWS.

Then copy your AWS Instance IPV4 :

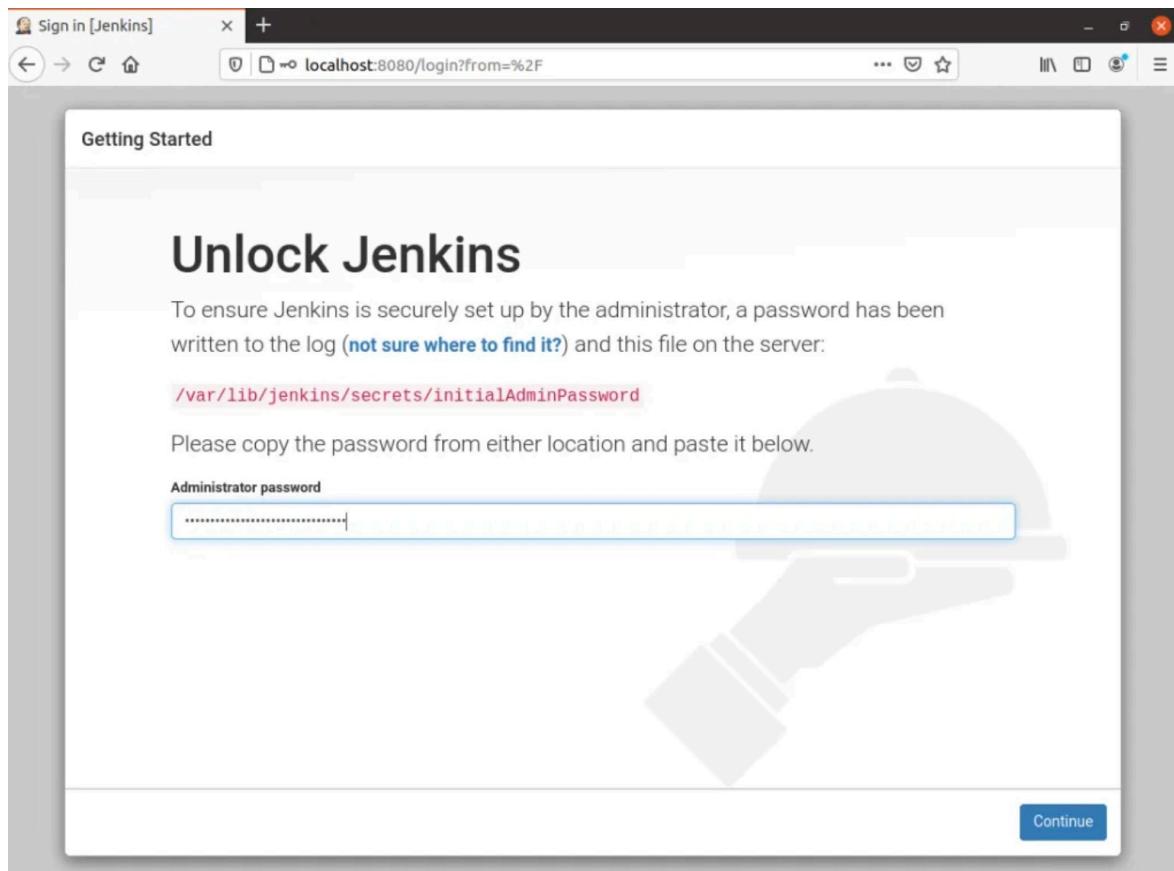
`ubuntu@IP.region_name.compute.amazonaws.com:8080`

This command results to opening the Jenkins docker container on port **8080**.



Initial password is available here:

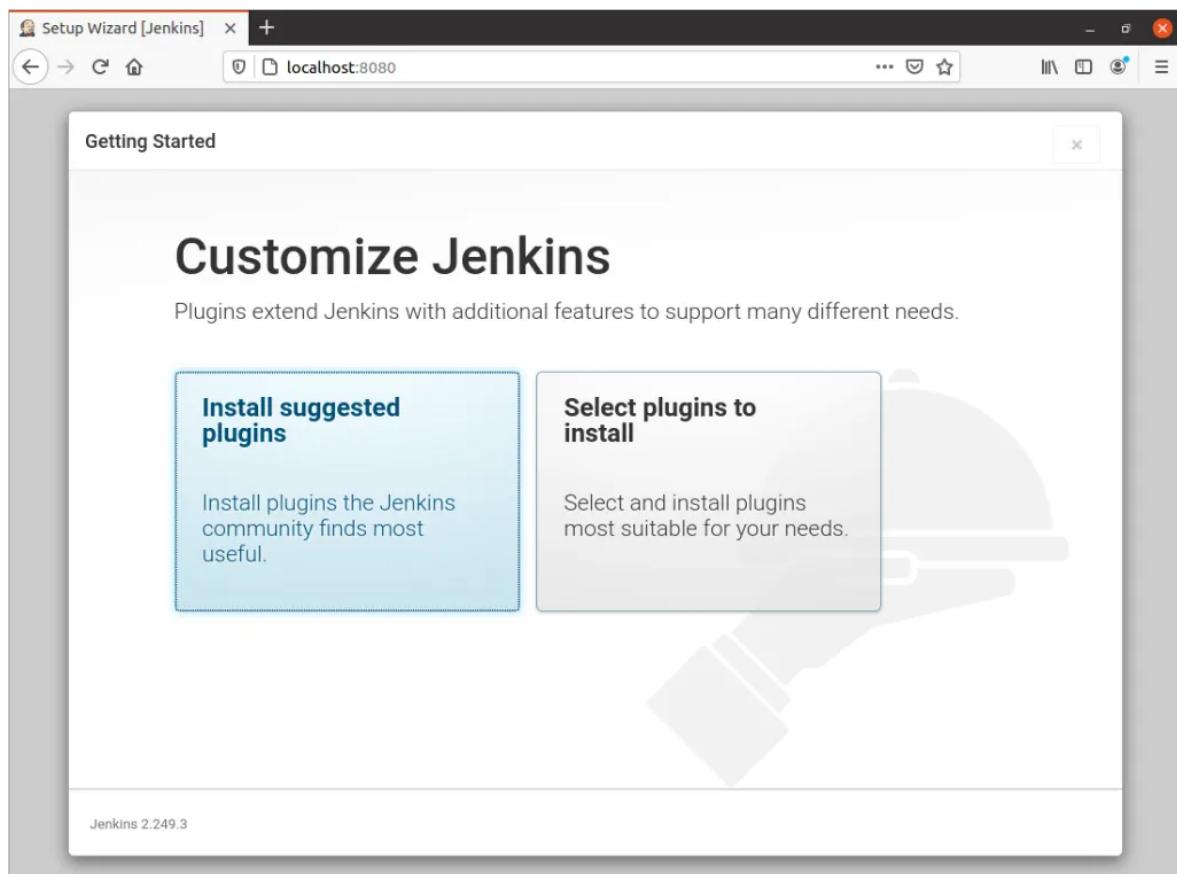
```
$ sudo cat /var/lib/jenkins/secrets/  
initialAdminPassword
```



Jenkins Plugins

At the next screen, you will be required to choose plugins for initial installation. If you do not know which plugins to set up

yet, use the “**Install suggested plugins**” option



The below is the screen you will be prompted after clicking on Install Suggested plugins.

Getting Started

The screenshot shows the Jenkins 'Getting Started' page. At the top, there's a large title 'Getting Started'. Below it is a blue horizontal bar with diagonal stripes. Underneath is a table with two columns of dependency matrices.

Dependency Matrix	
✓ Folders	✓ OWASP Markup Formatter
✓ Timestamper	✓ Workspace Cleanup
✓ Pipeline	✓ GitHub Branch Source
✓ Git	✓ SSH Build Agents
LDAP	Email Extension

Each cell in the matrix contains a green checkmark icon followed by a brief description of the dependency. To the right of the matrix is a vertical stack of dependency descriptions:

- ** Lockable Resources
- Pipeline
 - ** Java JSON Web Token (JJWT)
 - ** OkHttp
 - ** GitHub API
- Git
 - ** GitHub
 - GitHub Branch Source
 - Pipeline: GitHub Groovy Libraries
- ** Pipeline Graph Analysis
- ** Pipeline: REST API
- ** JavaScript GUI Lib: Handlebars bundle
- ** JavaScript GUI Lib: Moment.js bundle
- Pipeline: Stage View
- Git
- SSH Build Agents
- Matrix Authorization Strategy
- ** jnr-posix API
- PAM Authentication
- LDAP

At the bottom of the list is a note: '** - required dependency'.

Thereafter you will be prompted with the below screen.

Getting Started

Create First Admin User

Username:	<input type="text"/>
Password:	<input type="password"/>
Confirm password:	<input type="password"/>
Full name:	<input type="text"/>
E-mail address:	<input type="text"/>

You need to create your Jenkins user

account to get the Jenkins dashboard.

Getting Started

Create First Admin User

Username:	<input type="text" value="admin"/>
Password:	<input type="password" value="*****"/>
Confirm password:	<input type="password" value="*****"/>
Full name:	<input type="text" value="Jeet"/>
E-mail address:	<input type="text" value="170800116031@gmail.com"/>

After successfully entering the user details, you will be directed to the instance configuration page to confirm the URL for the Jenkins instance.

Getting Started

Instance Configuration

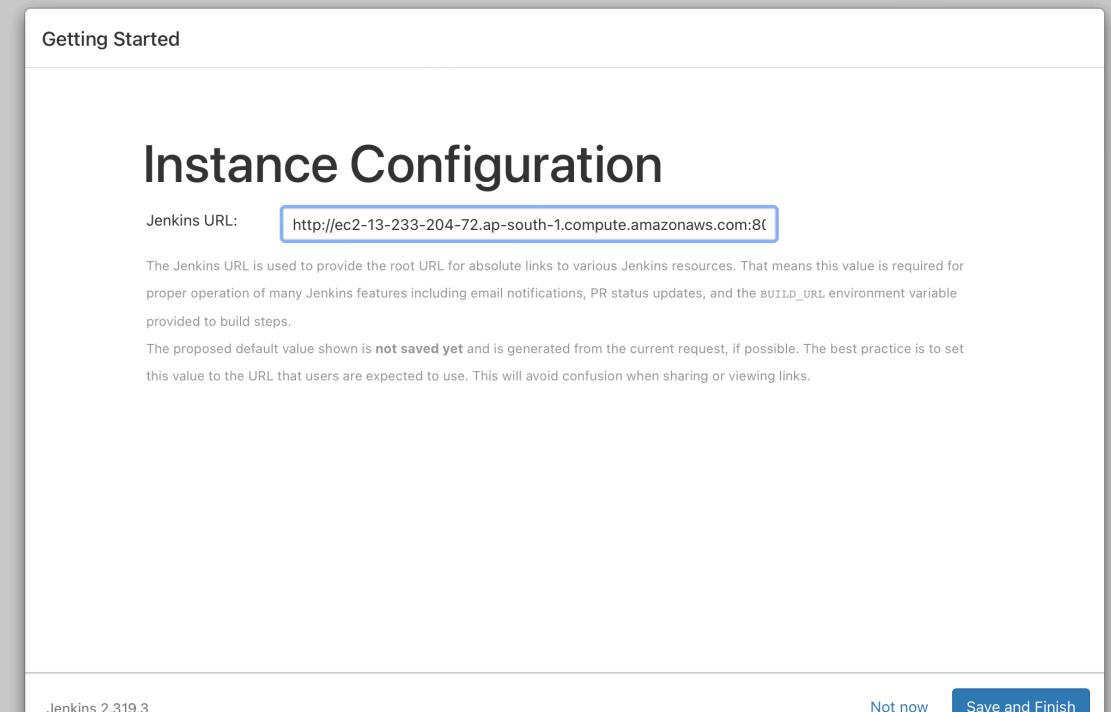
Jenkins URL: <http://ec2-13-233-204-72.ap-south-1.compute.amazonaws.com:80>

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 2.319.3

Not now Save and Finish



There after click on save and finish button.
The below screen will appear.

Getting Started

The screenshot shows a Jenkins setup completion page. At the top, it says "Jenkins is ready!". Below that, it says "Your Jenkins setup is complete." A blue button labeled "Start using Jenkins" is visible. The background features a large, faint Jenkins logo.

Jenkins 2.319.3

To access the Jenkins main dashboard, click the “**Start using Jenkins**” button. This action will bring you to the main Jenkins interface

The screenshot shows the Jenkins main dashboard. At the top, there's a navigation bar with a Jenkins logo, a search bar, and user information. The main content area has a title "Welcome to Jenkins!" and a sub-section "Start building your software project" with a "Create a job" button. Below that is a section "Set up a distributed build" with links for "Set up an agent", "Configure a cloud", and "Learn more about distributed builds". On the left, there's a sidebar with links like "New Item", "People", "Build History", "Manage Jenkins", "My Views", "Lockable Resources", and "New View". Under "Build Queue", it says "No builds in the queue.". Under "Build Executor Status", it lists "1 Idle" and "2 Idle".

To run MLOps pipeline in Jenkins you need to have Docker installed because in docker we have code dependencies.

The complete installation process is described in the Docker official documentation, so we'll provide the necessary commands here:

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common

$ curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -

$ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/
linux/ubuntu \
    $(lsb_release -cs) \
    stable"
$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli
containerd.io
```

Once Docker is successfully installed, you can run the Docker container.

Todays In Machine Learning world hyperparameter optimization we can say tuning is the problem of choosing a set of optimal Hyperparameter for a machine learning

algorithms. Hyperparameter is a parameter which value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned or in other words Hyper Parameters are those parameters which are assigned before the building of the model like number of filters , size of filter , number of neurons , number of epochs etc. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem. So to deal with this problem we can automate it where the hyper parameters will change and will make the accuracy of the model to the desire of the developer. So, instead of manually tuning our Hyperparameters, We are giving this job to Jenkins for Automation.

Work flow of Process or we say in term of jenkins automation jobs.

Sudo visudo

GNU nano 2.9.3

/etc/sudoers.tmp

```
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL
jenkins  ALL=(ALL) NOPASSWD: ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "#include" directives:

#include /etc/sudoers.d
```

[Read 29 lines]

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos **M-U** Undo
^X Exit **^R** Read File **^V** Replace **^U** Uncut Text **^T** To Spell **^L** Go To Line **M-E** Redo

sudo vi /etc/
default/jenkins

```
# defaults for Jenkins automation server
# pulled in from the init script; makes things easier.
NAME=jenkins

# arguments to pass to java

# Allow graphs etc. to work even when an X server is present
JAVA_ARGS="-Djava.awt.headless=true"

#JAVA_ARGS="-Xmx256m"

# make jenkins listen on IPv4 address
#JAVA_ARGS="-Djava.net.preferIPv4Stack=true"

PIDFILE=/var/run/$NAME/$NAME.pid

# user and group to be invoked as (default to jenkins)
JENKINS_USER="root"
JENKINS_GROUP=$NAME

# location of the jenkins war file
JENKINS_WAR=/usr/share/$NAME/$NAME.war

# jenkins home location
JENKINS_HOME=/var/lib/$NAME

# set this to false if you don't want Jenkins to run by itself
# in this set up, you are expected to provide a servlet container
# to host jenkins.
RUN_STANDALONE=true

# log location. this may be a syslog facility.priority
"/etc/default/jenkins" 79L, 2776C
```

19,19

Top

sudo service jenkins restart

Now we should get into EC2 instance and pull GitHub repository of our Machine Learning project.

```
$ git clone https://github.com/aniforverizon/
MLOPS-Pipeline-Jenkins.git
```

Then apply the following commands

```
$ mkdir MLOPS-project
$ cd MLOPS-project/
$ ls

$ sudo cp -R * /mlops/

$ sudo mkdir /mlops

$ sudo cp -R * /mlops/
$ cd /mlops/
$ ls
$ sudo usermod -aG Docker $USER
$ newgrp docker
$ sudo chmod 777 *
```

First we will create a container image that has python installed (in addition to all the other libraries which will be necessary for a ML/DL code to run) using Dockerfile.

```
FROM centos

RUN cd /etc/yum.repos.d/
RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
RUN sed -i 's|baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-*
RUN yum update -y

RUN yum install -y python36
RUN yum install -y epel-release
RUN yum groupinstall -y "development tools"
RUN yum install -y python36-devel
RUN pip3 install numpy
RUN pip3 install pandas
RUN pip3 install tensorflow
RUN pip3 install keras
RUN pip3 install --upgrade pip
RUN pip3 install setuptools
RUN pip3 install pillow
RUN pip3 install opencv-python
~

-- INSERT --
```

Now build this dockerfile with new image name. so we run this command on ubuntu:18.04 " docker build -t your_imagename:your_tagname ."

Thereafter go to Jenkins dashboard and click on “**New Items**” then below screen will be prompted.

The screenshot shows the Jenkins interface for creating a new item. A search bar at the top says "Enter an item name". Below it is a list of project types:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- 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**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

OK Copy from Type to autocomplete

Here enter the name of the job and click on **Freestyle project** then click on **OK**

Now you need to execute docker container through Jenkins.

JOB 1 :-

Jenkins should automatically start the respective container on pre install libraries to deploy code and start training.

For that below is the process. Move to the Build Environment and select “**Execute shell**” enter the

command to execute container

The screenshot shows the Jenkins configuration interface for a project named 'MLOPS-Docker_Container_Setup'. The top navigation bar includes links for 'Dashboard', 'Jenkins', 'Search', notifications (2), a user icon ('jeet'), and 'log out'. The main configuration page has tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab is selected.

Description: A large text area with a placeholder '[Plain text] Preview'.

Build Options:

- Discard old builds
- GitHub project
- This build requires lockable resources
- This project is parameterised
- Throttle builds
- Disable this project
- Execute concurrent builds if necessary

Source Code Management: Set to 'None' (selected) or 'Git'.

Build Triggers: Set to 'None' (selected). Buttons for 'Save' and 'Apply' are visible.

The screenshot shows the Jenkins 'Build Triggers' configuration page. The 'Build Triggers' tab is selected. Under 'Build Triggers', there are five options: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. Each option has a question mark icon to its right. Below the triggers is the 'Build Environment' section with six options: 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Abort the build if it's stuck', 'Add timestamps to the Console Output', 'Inspect build log for published Gradle build scans', and 'With Ant'. Each option has a question mark icon to its right. Under the 'Build' section, there is a single 'Execute shell' step. The 'Command' field contains the following Jenkinsfile code:

```
sudo docker container run -dit -v /mllops/:/root/mllops/ --name jobs2 jeet:v1
```

Below the command field, there is a link to 'See the list of available environment variables'. At the bottom of the step, there are 'Save' and 'Apply' buttons, and an 'Advanced...' button to the right.

Job 2: This job will check the health of the container, if somehow container is stopped then it will restart the container.

To do this again create a new item that will check the health of the container from Job1(MLOPS-Docker_Container_Setup). For that

Dashboard > MLOPS-Health_Checker >

- [General](#)
- [Source Code Management](#)
- [Build Triggers](#)
- [Build Environment](#)
- [Build](#)
- [Post-build Actions](#)

Source Code Management

None Git [?](#)

Build Triggers

Trigger builds remotely (e.g., from scripts) [?](#)
 Build after other projects are built [?](#)

Projects to watch

MLOPS-Docker_Container_Setup, [?](#)

Trigger only if build is stable
 Trigger even if the build is unstable
 Trigger even if the build fails
 Always trigger, even if the build is aborted

Build periodically [?](#)
 GitHub hook trigger for GITScm polling [?](#)
 Poll SCM [?](#)

Build Environment

Delete workspace before build starts
 Use secret text(s) or file(s)
 Abort the build if it's stuck [?](#)

Add timestamps to the Console Output [?](#)
 With Ant

[Save](#) [Apply](#) [Gradle build scans](#) [?](#)

Dashboard > MLOPS-Health_Checker >

- [General](#)
- [Source Code Management](#)
- [Build Triggers](#)
- [Build Environment](#)
- [Build](#)
- [Post-build Actions](#)

Build Environment

Delete workspace before build starts
 Use secret text(s) or file(s)
 Abort the build if it's stuck [?](#)
 Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 With Ant [?](#)

Build

[Execute shell](#) [x](#) [?](#)

Command

```
if sudo docker ps -a | grep jobs2
then
exit 0
else
sudo docker container run -dir -v /mlops:/root/mlops/ --name jobs2 jeet:v1
exit 1
```

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

[Advanced...](#)

[Add build step ▾](#)

Post-build Actions

[Add post-build action ▾](#)

[Save](#) [Apply](#)

Job3 : This job will start the training for the Machine Learning model if and only if the docker container is in running mode.

The screenshot shows the Jenkins job configuration interface for a job named "MLOPS-Exec_ProgramFile". The top navigation bar includes links for Dashboard, MLOPS-Exec_ProgramFile, General, Source Code Management, Build Triggers (which is selected), Build Environment, Build, and Post-build Actions. The Build Triggers section contains the following configuration:

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Projects to watch:
 - MLOPS-Docker_Container_Setup
- Trigger only if build is stable ?
- Trigger even if the build is unstable ?
- Trigger even if the build fails ?
- Always trigger, even if the build is aborted ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

The Build Environment section contains the following configuration:

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

The Build section contains two buttons: Save (blue) and Apply (white). There is also a red X button and a question mark icon in the top right corner of the build section.

Dashboard > MLOPS-Exec_ProgramFile >

General Source Code Management Build Triggers **Build Environment** Build Post-build Actions

- Add timestamps to the Console Output
- Inspect build log for published Gradle build scans
- With Ant

Build

Execute shell

Command

```
sudo docker exec jobs2 python3 /root/mlops/program.py
```

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

[Advanced...](#)

[Add build step ▾](#)

Post-build Actions

[Add post-build action ▾](#)

Save **Apply**

You can check the output for the training process by the following process.

Jenkins

Dashboard > MLOPS-Exec_ProgramFile >

[Back to Dashboard](#) [Status](#) [Changes](#) [Workspace](#) [Build Now](#) [Configure](#) [Delete Project](#) [Rename](#)

Project MLOPS-Exec_ProgramFile

[Add description](#) [Disable Project](#)

Upstream Projects

[MLOPS-Docker_Container_Setup](#)

Downstream Projects

[Omloepsjeet2](#)

Permalinks

- Last build (#11), 20 sec ago
- Last stable build (#11), 20 sec ago
- Last successful build (#11), 20 sec ago
- Last failed build (#10), 3 min 28 sec ago
- Last unsuccessful build (#10), 3 min 28 sec ago
- Last completed build (#11), 20 sec ago

#	Build Number	Last Run
11	#11	11-Mar-2022 04:28
6	#6	10-Mar-2022 04:02
5	#5	09-Mar-2022 06:12
4	#4	09-Mar-2022 06:09

 Jenkins

Dashboard > MLOPS-Exec_ProgramFile > #11

 Back to Project

 Status

 Changes

 Console Output

 View as plain text

 Edit Build Information

 Delete build '#11'

 Previous Build

Console Output

Started by user **jeet**
 Running as SYSTEM
 Building in workspace /var/lib/jenkins/workspace/MLOPS-Exec_ProgramFile
 [MLOPS-Exec_ProgramFile] \$ /bin/sh -xe /tmp/jenkins6989810909729746277.sh
 + sudo docker exec jobs2 python3 /root/mlops/program.py
 Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>

```

8192/11490434 [.....] - ETA: 0s
4202496/11490434 [=====>.....] - ETA: 0s
9723904/11490434 [=====>.....] - ETA: 0s
11493376/11490434 [=====>.....] - ETA: 0s
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
...np._qint8 = np.dtype((("qint8", np.int8, 1)))
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
...np._quint8 = np.dtype((("quint8", np.uint8, 1)))
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
...np._qint16 = np.dtype((("qint16", np.int16, 1)))
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:519: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
...np._quint16 = np.dtype((("quint16", np.uint16, 1)))
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:520: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
...np._qint32 = np.dtype((("qint32", np.int32, 1)))
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
...np._quint32 = np.dtype((("quint32", np.uint32, 1)))
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
...np._qint8 = np.dtype((("qint8", np.int8, 1)))
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or 'ltype'

```

Dashboard > MLOPS-Exec_ProgramFile > #11

```

2022-03-11 04:28:54.26652: I tensorflow/compiler/xla/service/service.cc:110] StreamExecutor device (0): <underlined>
2022-03-11 04:28:54.347023: W tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412] (One-time warning): Not using XLA:CPU for
cluster because envvar TF_XLA_FLAGS=--tf_xla_cpu_global_jit was not set. If you want XLA:CPU, either set that envvar, or use
experimental_jit_scope to enable XLA:CPU. To confirm that XLA is active, pass --vmodule=xla_compilation_cache=1 (as a proper
command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=-xla_hlo_profile.
size of image (28, 28)
Model: "sequential"

```

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 26, 26, 8)	80
max_pooling2d (MaxPooling2D)	(None, 13, 13, 8)	0
flatten (Flatten)	(None, 1352)	0
dense (Dense)	(None, 10)	13530
dense_1 (Dense)	(None, 10)	110
<hr/>		
Total params: 13,720		
Trainable params: 13,720		
Non-trainable params: 0		
<hr/>		
None		

Train on 60000 samples, validate on 10000 samples

```

128/60000 [.....] - ETA: 1:57 - loss: 44.0682 - acc: 0.0859
512/60000 [.....] - ETA: 35s - loss: 31.1298 - acc: 0.0820
896/60000 [.....] - ETA: 23s - loss: 24.2630 - acc: 0.1083
1280/60000 [.....] - ETA: 19s - loss: 18.9010 - acc: 0.1266
1664/60000 [.....] - ETA: 16s - loss: 15.2765 - acc: 0.1460
2048/60000 [.....] - ETA: 14s - loss: 12.9203 - acc: 0.1489
2432/60000 [.....] - ETA: 13s - loss: 11.2406 - acc: 0.1530
2816/60000 [.....] - ETA: 12s - loss: 10.0148 - acc: 0.1499
3200/60000 [.....] - ETA: 12s - loss: 9.0761 - acc: 0.1541
3584/60000 [.....] - ETA: 11s - loss: 8.3391 - acc: 0.1554
3968/60000 [.....] - ETA: 11s - loss: 7.7470 - acc: 0.1552
4352/60000 [=>.....] - ETA: 10s - loss: 7.2563 - acc: 0.1560
4736/60000 [=>.....] - ETA: 10s - loss: 6.8467 - acc: 0.1582
5120/60000 [=>.....] - ETA: 10s - loss: 6.4913 - acc: 0.1621

```

```

Dashboard > MLOPS-Exec_ProgramFile > #11

46720/60000 [=====>.....] - ETA: 1s - loss: 2.2967 - acc: 0.3157
47104/60000 [=====>.....] - ETA: 1s - loss: 2.2921 - acc: 0.3163
47488/60000 [=====>.....] - ETA: 1s - loss: 2.2871 - acc: 0.3167
47872/60000 [=====>.....] - ETA: 1s - loss: 2.2818 - acc: 0.3169
48256/60000 [=====>.....] - ETA: 1s - loss: 2.2766 - acc: 0.3176
48640/60000 [=====>.....] - ETA: 1s - loss: 2.2708 - acc: 0.3181
49024/60000 [=====>.....] - ETA: 1s - loss: 2.2656 - acc: 0.3186
49408/60000 [=====>.....] - ETA: 1s - loss: 2.2598 - acc: 0.3192
49792/60000 [=====>.....] - ETA: 1s - loss: 2.2548 - acc: 0.3196
50176/60000 [=====>.....] - ETA: 1s - loss: 2.2487 - acc: 0.3206
50560/60000 [=====>.....] - ETA: 1s - loss: 2.2437 - acc: 0.3212
50944/60000 [=====>.....] - ETA: 1s - loss: 2.2386 - acc: 0.3217
51456/60000 [=====>.....] - ETA: 1s - loss: 2.2315 - acc: 0.3225
51840/60000 [=====>.....] - ETA: 1s - loss: 2.2269 - acc: 0.3231
52224/60000 [=====>.....] - ETA: 1s - loss: 2.2220 - acc: 0.3235
52608/60000 [=====>.....] - ETA: 1s - loss: 2.2170 - acc: 0.3243
52992/60000 [=====>.....] - ETA: 0s - loss: 2.2116 - acc: 0.3250
53376/60000 [=====>.....] - ETA: 0s - loss: 2.2060 - acc: 0.3258
53760/60000 [=====>.....] - ETA: 0s - loss: 2.2001 - acc: 0.3266
54144/60000 [=====>.....] - ETA: 0s - loss: 2.1959 - acc: 0.3272
54528/60000 [=====>.....] - ETA: 0s - loss: 2.1910 - acc: 0.3278
54912/60000 [=====>.....] - ETA: 0s - loss: 2.1867 - acc: 0.3282
55296/60000 [=====>.....] - ETA: 0s - loss: 2.1847 - acc: 0.3289
55680/60000 [=====>.....] - ETA: 0s - loss: 2.1798 - acc: 0.3295
56192/60000 [=====>.....] - ETA: 0s - loss: 2.1737 - acc: 0.3300
56576/60000 [=====>.....] - ETA: 0s - loss: 2.1685 - acc: 0.3307
56960/60000 [=====>.....] - ETA: 0s - loss: 2.1643 - acc: 0.3312
57344/60000 [=====>.....] - ETA: 0s - loss: 2.1601 - acc: 0.3316
57728/60000 [=====>.....] - ETA: 0s - loss: 2.1563 - acc: 0.3319
58112/60000 [=====>.....] - ETA: 0s - loss: 2.1523 - acc: 0.3323
58496/60000 [=====>.....] - ETA: 0s - loss: 2.1486 - acc: 0.3326
58880/60000 [=====>.....] - ETA: 0s - loss: 2.1450 - acc: 0.3327
59264/60000 [=====>.....] - ETA: 0s - loss: 2.1408 - acc: 0.3332
59648/60000 [=====>.....] - ETA: 0s - loss: 2.1371 - acc: 0.3333
60000/60000 [=====>.....] - 9s 152us/sample - loss: 2.1336 - acc: 0.3336 - val_loss: 1.5168 - val_acc: 0.3979
Test loss: 1.5168055267333984
Test accuracy: 0.3979
Accuracy is : 0.3979 %
Triggering a new build of mlopsjeet2
Finished: SUCCESS

```

Once the training process is done then here comes mailing service in the picture.

Job 4: This job will trigger the mail regarding your model's accuracy. It will check the accuracy of the model, if it is below 90% then it will send mail to the developer that the accuracy doesn't match with the expected accuracy. And then it will trigger next job for tuning hyperparameters. It will also depend on the previous job.

Dashboard > MLOPS-Triggering_Mail >

General	Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions
Build Triggers					
<input type="checkbox"/> Trigger builds remotely (e.g., from scripts) ? <input checked="" type="checkbox"/> Build after other projects are built ? Projects to watch MLOPS-Exec_ProgramFile <input checked="" type="radio"/> Trigger only if build is stable <input type="radio"/> Trigger even if the build is unstable <input type="radio"/> Trigger even if the build fails <input type="radio"/> Always trigger, even if the build is aborted <input type="checkbox"/> Build periodically ? <input type="checkbox"/> GitHub hook trigger for GITScm polling ? <input type="checkbox"/> Poll SCM ?					
Build Environment					
<input type="checkbox"/> Delete workspace before build starts ? <input type="checkbox"/> Use secret text(s) or file(s) ? <input type="checkbox"/> Abort the build if it's stuck ? <input type="checkbox"/> Add timestamps to the Console Output ? <input type="checkbox"/> Inspect build log for published Gradle build scans ? <input type="checkbox"/> With Ant ?					
Build <div style="border: 1px solid #ccc; padding: 5px; display: flex; justify-content: space-between;"> Save Apply X ? </div>					

Dashboard > MLOPS-Triggering_Mail >

General	Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions
<input type="checkbox"/> With Ant ?					
Build <div style="border: 1px solid #ccc; padding: 5px;"> Execute shell Command <pre>if [[\$(sudo cat /mllops/accuracy.txt) < "0.9000"]] then echo "Accuracy not matched!!!" sudo docker exec jobs1 python3 /root/mllops/mail.py else echo "Accuracy matched !!!" sudo docker exec jobs1 python3 /root/mllops/mail_success.py fi</pre> </div>					
See the list of available environment variables <div style="text-align: right; margin-top: -10px;"> Advanced... </div>					
Add build step ▾					
Post-build Actions <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Add post-build action ▾ </div>					
<div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Save Apply </div>					

JOB 5 : This job will be responsible for hyper parameter tuning of the model and will make changes in the code in order to achieve the required accuracy. Then send the mail to developer regarding the accuracy.

The screenshot shows the Jenkins job configuration page for 'MLOPS-Hyper_Parameter_Tuning'. The top navigation bar includes 'Dashboard', 'MLOPS-Hyper_Parameter_Tuning', and tabs for 'General', 'Source Code Management' (which is selected), 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Source Code Management' section has 'None' selected. The 'Build Triggers' section contains a 'Projects to watch' list with 'MLOPS-Triggering_Mail' and several trigger options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', 'Trigger even if the build fails', and 'Always trigger, even if the build is aborted'. It also lists 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. The 'Build Environment' section includes options for 'Delete workspace before build starts', 'Use secret text(s) or file(s)', and 'Abort the build if it's stuck'. At the bottom, there are buttons for 'Save' (highlighted in blue), 'Apply' (highlighted in green), and 'Gradle build scans'.

General Source Code Management Build Triggers **Build Environment** Build Post-build Actions

Add timestamps to the Console Output
 Inspect build log for published Gradle build scans
 With Ant

Build

Execute shell

Command

```
sudo docker exec jobs2 python3 /root/mlops/tweaker.py
```

See the list of available environment variables

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save Apply

To check the accuracy of tuned model you can check the below process

Dashboard > MLOPS-Hyper_Parameter_Tuning > #5

Back to Project Status Changes **Console Output** View as plain text Edit Build Information Delete build '#5' Previous Build

Console Output

Skipping 49 KB.. [Full Log](#)

Epoch 2/2

```

128/60000 [.....] - ETA: 9s - loss: 1.5283 - acc: 0.3438
512/60000 [.....] - ETA: 9s - loss: 1.6375 - acc: 0.3672
896/60000 [.....] - ETA: 9s - loss: 1.6634 - acc: 0.3571
1280/60000 [.....] - ETA: 9s - loss: 1.6664 - acc: 0.3641
1664/60000 [.....] - ETA: 9s - loss: 1.5993 - acc: 0.3630
2048/60000 [>.....] - ETA: 9s - loss: 1.6284 - acc: 0.3672
2432/60000 [>.....] - ETA: 9s - loss: 1.6051 - acc: 0.3787
2816/60000 [>.....] - ETA: 9s - loss: 1.6040 - acc: 0.3807
3200/60000 [>.....] - ETA: 9s - loss: 1.5993 - acc: 0.3787
3584/60000 [>.....] - ETA: 8s - loss: 1.6116 - acc: 0.3800
3968/60000 [>.....] - ETA: 8s - loss: 1.6133 - acc: 0.3798
4352/60000 [=>.....] - ETA: 8s - loss: 1.6110 - acc: 0.3791
4736/60000 [=>.....] - ETA: 8s - loss: 1.6063 - acc: 0.3790
5120/60000 [=>.....] - ETA: 8s - loss: 1.6116 - acc: 0.3797
5504/60000 [=>.....] - ETA: 8s - loss: 1.6065 - acc: 0.3814
5888/60000 [=>.....] - ETA: 8s - loss: 1.6070 - acc: 0.3806
6272/60000 [=>.....] - ETA: 8s - loss: 1.6027 - acc: 0.3819
6656/60000 [=>.....] - ETA: 8s - loss: 1.6051 - acc: 0.3806
7040/60000 [=>.....] - ETA: 8s - loss: 1.6096 - acc: 0.3791
7424/60000 [=>.....] - ETA: 8s - loss: 1.6049 - acc: 0.3789
7808/60000 [==>.....] - ETA: 8s - loss: 1.5996 - acc: 0.3781
8192/60000 [===>.....] - ETA: 8s - loss: 1.5976 - acc: 0.3766
8576/60000 [===>.....] - ETA: 8s - loss: 1.5976 - acc: 0.3745
8960/60000 [===>.....] - ETA: 8s - loss: 1.5920 - acc: 0.3762
9344/60000 [===>.....] - ETA: 7s - loss: 1.5913 - acc: 0.3751
9728/60000 [===>.....] - ETA: 7s - loss: 1.5847 - acc: 0.3758
10112/60000 [=====>.....] - ETA: 7s - loss: 1.5860 - acc: 0.3757
10496/60000 [=====>.....] - ETA: 7s - loss: 1.5902 - acc: 0.3725
10880/60000 [=====>.....] - ETA: 7s - loss: 1.5906 - acc: 0.3736
11264/60000 [=====>.....] - ETA: 7s - loss: 1.5881 - acc: 0.3728

```

```
Dashboard > MLOPS-Hyper_Parameter_Tuning > #5
.....
53632/60000 [=====>>>....] - ETA: 1s - loss: 0.1436 - acc: 0.9637
53889/60000 [=====>>>....] - ETA: 1s - loss: 0.1435 - acc: 0.9637
54144/60000 [=====>>>....] - ETA: 1s - loss: 0.1432 - acc: 0.9637
54400/60000 [=====>>>....] - ETA: 1s - loss: 0.1430 - acc: 0.9638
54656/60000 [=====>>>....] - ETA: 1s - loss: 0.1428 - acc: 0.9638
54912/60000 [=====>>>....] - ETA: 1s - loss: 0.1424 - acc: 0.9639
55168/60000 [=====>>>....] - ETA: 1s - loss: 0.1428 - acc: 0.9639
55424/60000 [=====>>>....] - ETA: 0s - loss: 0.1429 - acc: 0.9639
55680/60000 [=====>>>....] - ETA: 0s - loss: 0.1426 - acc: 0.9639
55936/60000 [=====>>>....] - ETA: 0s - loss: 0.1426 - acc: 0.9639
56192/60000 [=====>>>....] - ETA: 0s - loss: 0.1425 - acc: 0.9638
56448/60000 [=====>>>....] - ETA: 0s - loss: 0.1428 - acc: 0.9637
56704/60000 [=====>>>....] - ETA: 0s - loss: 0.1427 - acc: 0.9637
56960/60000 [=====>>>....] - ETA: 0s - loss: 0.1426 - acc: 0.9637
57216/60000 [=====>>>....] - ETA: 0s - loss: 0.1426 - acc: 0.9638
57472/60000 [=====>>>....] - ETA: 0s - loss: 0.1427 - acc: 0.9637
57728/60000 [=====>>>....] - ETA: 0s - loss: 0.1424 - acc: 0.9638
57984/60000 [=====>>>....] - ETA: 0s - loss: 0.1423 - acc: 0.9637
58240/60000 [=====>>>....] - ETA: 0s - loss: 0.1422 - acc: 0.9638
58496/60000 [=====>>>....] - ETA: 0s - loss: 0.1422 - acc: 0.9638
58752/60000 [=====>>>....] - ETA: 0s - loss: 0.1422 - acc: 0.9637
59008/60000 [=====>>>....] - ETA: 0s - loss: 0.1423 - acc: 0.9637
59264/60000 [=====>>>....] - ETA: 0s - loss: 0.1426 - acc: 0.9637
59520/60000 [=====>>>....] - ETA: 0s - loss: 0.1425 - acc: 0.9637
59776/60000 [=====>>>....] - ETA: 0s - loss: 0.1424 - acc: 0.9637
60000/60000 [=====>>>....] - 14s 230us/sample - loss: 0.1424 - acc: 0.9637 - val_loss: 0.1440 - val_acc: 0.9634
Test loss : 14.403797857314348
-----Accuracy of the model : 96.34000062942505
```

REST API Jenkins 2.319.3

Once the pipeline is done then you will get the below related dashboard .

Jenkins

Dashboard >

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
		job1	N/A	N/A	N/A
		MLOPS-Docker_Container_Setup	17 min - #20	22 hr - #19	1 sec
		MLOPS-Exec_ProgramFile	13 min - #11	17 min - #10	18 sec
		MLOPS-Hyper_Parameter_Tuning	2 min 59 sec - #5	5 min 21 sec - #4	1 min 21 sec
		MLOPS-Triggering_Mail	5 min 33 sec - #15	8 min 17 sec - #14	3.2 sec

Icon: S M L

Icon legend

Atom feed for all

Atom feed for failures

Atom feed for just latest builds

Search

?

2

!

1

jeet

log out

REST API Jenkins 2.319.3

Keep learning !!!!