

NAME : VADLA LAKSHMI SWETHA

BATCH NO. : 126

MAIL ID: [lakshmiswethavadla@gmail.com](mailto:lakshmiswethavadla@gmail.com)

## PROJECT :

### Create your own Image by using required Dockerfile instructions

Launch an instance and connect to it →

EC2 > Instances > Launch an instance

### Launch an instance info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

#### Name and tags info

Name

 [Add additional tags](#)

#### ▼ Application and OS Images (Amazon Machine Image) info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recents | **Quick Start**

Amazon Linux

aws

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debi

Debi

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

#### ▼ Summary

Number of instances info

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...read more  
ami-0d53d72369355a9d5

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

**Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100

Cancel **Launch instance** [Review commands](#)

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.medium

Family: t2 2 vCPU 4 GiB Memory Current generation: true  
On-Demand RHEL base pricing: 0.084 USD per Hour  
On-Demand Linux base pricing: 0.0552 USD per Hour  
On-Demand Windows base pricing: 0.0732 USD per Hour  
On-Demand SUSE base pricing: 0.1552 USD per Hour

☒ All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

docker

 [Create new key pair](#)

VPC - *required* [Info](#)

vpc-0a57ce03d7d57af63  
172.31.0.0/16

(default) ▼



Subnet [Info](#)

subnet-0a34a08d056e55cd6  
VPC: vpc-0a57ce03d7d57af63 Owner: 869935071032  
Availability Zone: us-west-1a Zone type: Availability Zone  
IP addresses available: 4091 CIDR: 172.31.16.0/20

 [Create new subnet](#) 

Auto-assign public IP [Info](#)

Enable ▼

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

Security group name - *required*

docker

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and . \_ - / ! # , @ [ ] + = & ; ' ! \$ \*

Description - *required* [Info](#)

launch-wizard-1 created 2024-09-15T12:13:08.257Z

Inbound Security Group Rules

▼ Security group rule 1 (All, All, 0.0.0.0/0)

[Remove](#)

Type [Info](#)

All traffic ▼

Protocol [Info](#)

All

Port range [Info](#)

All

[EC2](#) > [Instances](#) > Launch an instance

 Success

Successfully initiated launch of instance (i-08ed31daf1bc36269)

```
aws Services Search [Alt+S]
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

System information as of Sun Sep 15 12:16:52 UTC 2024

System load:  0.39      Processes:      118
Usage of /:   22.8% of 6.71GB   Users logged in:  0
Memory usage: 5%          IPv4 address for enX0: 172.31.17.108
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-17-108:~$ sudo -i
```

## Step 1: Install Docker

Make sure Docker is installed on your machine.

```
root@ip-172-31-17-108:~# apt update -y
Hit:1 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
0% [4 Packages 6431 kB/15.0 MB 43%] [5 InRelease 14.2 kB/126 kB 11%]

root@ip-172-31-17-108:~# apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debotstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 133 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Get:1 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:4 http://us-west-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
17% [4 containerd 14.7 kB/38.6 MB 0%]

root@ip-172-31-17-108:~# docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu4.1
root@ip-172-31-17-108:~#
```

## Step 2: Create a Project Directory

Create a new directory for your project. This will hold your `Dockerfile` and any other files you need.

- `mkdir my-docker-project`
- `cd my-docker-project`

```
root@ip-172-31-17-108:~# mkdir my-docker-project
root@ip-172-31-17-108:~# cd my-docker-project/
```

## Step 3: Create the Dockerfile

Inside your project directory, create a file named `Dockerfile` (with no extension). This file will contain the instructions for building your Docker image.

- **vi Dockerfile**

```
root@ip-172-31-17-108:~/my-docker-project# vi Dockerfile
root@ip-172-31-17-108:~/my-docker-project#
```

## Step 4: Write Dockerfile Instructions

Open the `Dockerfile` with a text editor and add the necessary instructions. Below is an example `Dockerfile` for creating a simple Python application image.

### Example: Dockerfile for a Python Application

- Step 1: Use an official Python runtime as a parent image  
FROM python:3.9-slim
- Step 2: Set the working directory in the container  
WORKDIR /app
- Step 3: Copy the current directory contents into the container at /app  
COPY . /app
- Step 4: Install any needed packages specified in requirements.txt  
RUN pip install --no-cache-dir -r requirements.txt
- Step 5: Make port 80 available to the world outside this container  
EXPOSE 80
- Step 6: Define environment variable  
ENV NAME World
- Step 7: Run app.py when the container launches  
CMD ["python", "app.py"]

[illegible]

## Explanation of Dockerfile Instructions

1. FROM: Specifies the base image to use. In this case, `python:3.9-slim` is used, which is a lightweight version of the Python image.
2. WORKDIR: Sets the working directory inside the container. All subsequent instructions will be run in this directory.
3. COPY: Copies files from your local directory (where the `Dockerfile` is) to the working directory inside the container.
4. RUN: Executes commands inside the container. Here, it installs Python packages listed in `requirements.txt`.
5. EXPOSE: Documents the port on which the container will listen for connections.
6. ENV: Sets an environment variable inside the container.
7. CMD: Specifies the command to run when the container starts. Here, it runs `app.py` using Python.

## Step 5: Create Additional Files

For the example Dockerfile above, you would need an `app.py` and a `requirements.txt` file in the same directory.

**app.py:**

python

```
root@ip-172-31-17-108:~/my-docker-project# vi app.py
```

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

**requirements.txt:**

```
root@ip-172-31-17-108:~/my-docker-project# vi requirements.txt
root@ip-172-31-17-108:~/my-docker-project#
```

[illegible]

## Step 6: Build the Docker Image

Use the `docker build` command to build the image from your Dockerfile.

```
docker build -t my-python-app .
```

```
root@ip-172-31-17-108:~/my-docker-project# docker build -t my-python-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  4.096kB
Step 1/7 : FROM python:3.9-slim
--> 397ed8d31636
Step 2/7 : WORKDIR /app
--> Using cache
--> 04868a926717
Step 3/7 : COPY . /app
--> b2321905484a
Step 4/7 : RUN pip install --no-cache-dir -r requirements.txt
--> Running in 7084774d9b7d
```

- `-t my-python-app`: Tags the image with the name `my-python-app`.

- ```: Specifies the build context, which is the current directory.

## Step 7: Run the Docker Container

Run a container from your newly created image to test it.

```
docker run -p 4000:80 my-python-app
```

```
root@ip-172-31-17-108:~/my-docker-project# docker run -p 4000:80 my-python-app
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:80
* Running on http://172.17.0.2:80
Press CTRL+C to quit
```

- `-p 4000:80`: Maps port 80 in the container to port 4000 on your host machine.

- `my-python-app`: The name of the image to run.

## Step 8: Test Your Application

Open a web browser and go to `http://localhost:4000`. You should see "Hello, World!" displayed.



## Step 9: Clean Up

To stop the container, press `Ctrl+C` in the terminal. To remove the container and image, use the following commands:

**`docker ps -a`** # List all containers

**`docker rm <container_id>`** # Remove a container

**`docker rmi my-python-app`** # Remove an image