

# All Projects

## 1. Git Repository

The project git repository is given below:

<https://github.com/kpassoubady/copilot>

## 2. Install Homebrew

Use <https://brew.sh> for the latest information.

Homebrew is a package manager for macOS that simplifies the installation of software. If brew has not been installed already, install it following the below steps:

1. Open Terminal.
2. Run the following command

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

## 1 Use HomeBrew to Install VSCode on the MAC (Optional)

1. Using HomeBrew Install Visual Studio Code (VSCode) **if not already installed:**

**brew install --cask visual-studio-code**

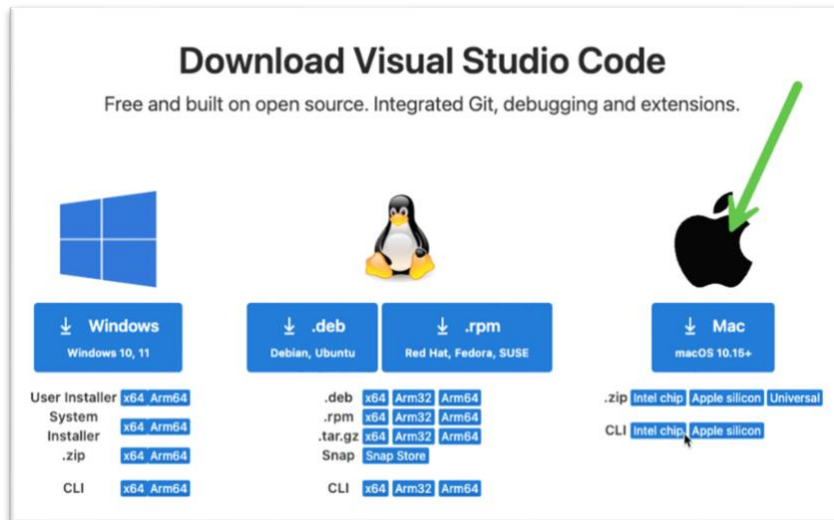
## 2 Installing Visual Studio Code

The below section shows different OS and various ways to install the VSCode.

- *Direct download and Install Visual Studio Code on Mac*

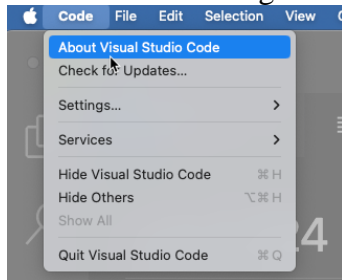
Go to <https://code.visualstudio.com/Download>

Download Visual Studio Code, and unzip if needed.

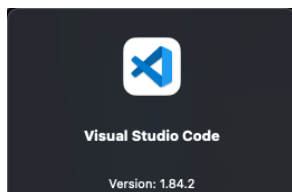


Move the app to your application folder

To find the version go to the top menu on Visual Studio, Code → About Visual Studio Code

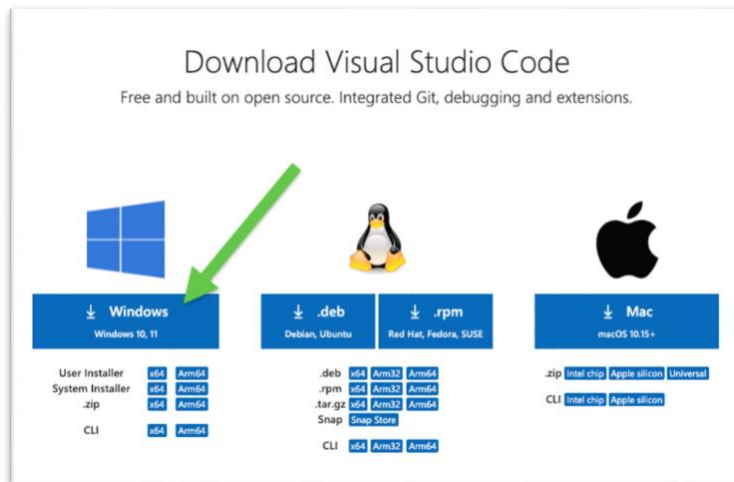


Clicking code → Check for Updates allows you to upgrade to the latest version



- Installing Visual Studio Code (Direct download on Windows)

Go to <https://code.visualstudio.com/download>



If you need step-by-step instructions, follow the below link to install:

<https://pureinfotech.com/install-visual-studio-code-windows-10/>

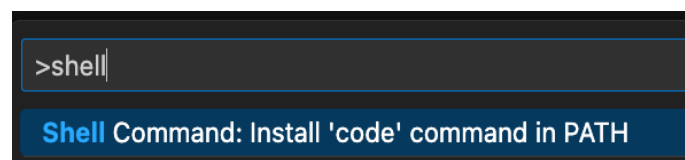
### 3. Install the GitHub Copilot Plugin for VS Code

GitHub Copilot assists in generating code during the project.

1. Open VSCode and go to the **Extensions** view (Cmd+Shift+X).
2. Search for **GitHub Copilot** and install the extension.
3. Authenticate your GitHub account to enable Copilot.

### 4. Install VS Code in your Shell for Mac

4. Go to the top of VS and select menu View → Command Palette...
5. Open the Command Palette via `⇧⌘P` and type the shell command to find the Shell Command:



## 5. Github Copilot CLI

URL: <https://github.com/cli/cli#installation>

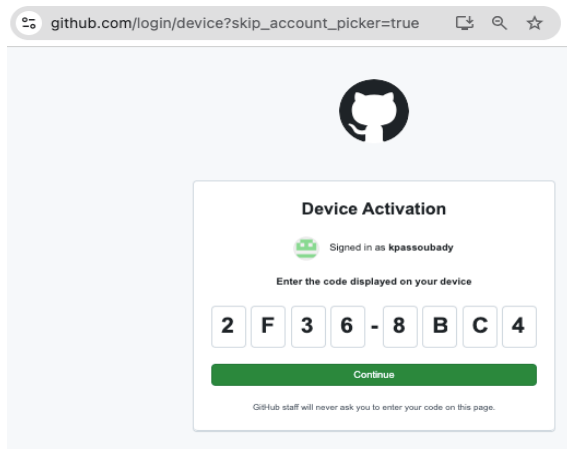
### brew install gh

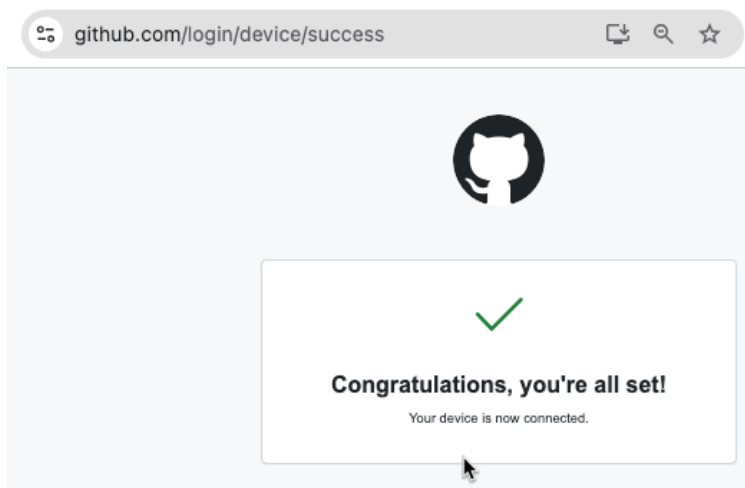
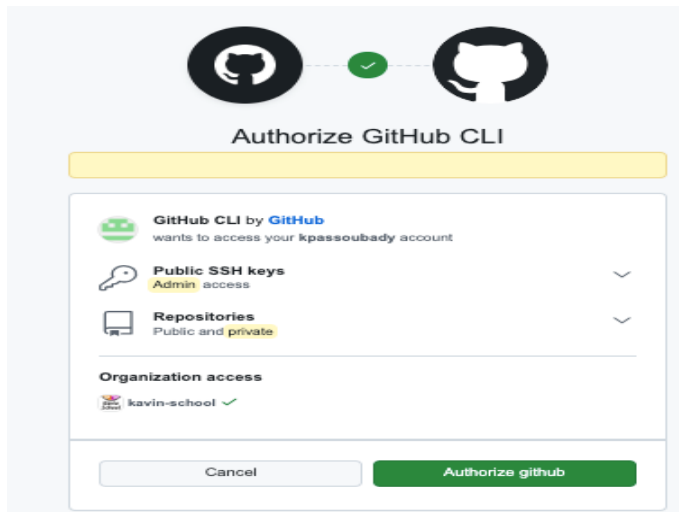
```
meera@Kangs-Home-Mac ~ % gh auth login

? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /Users/meera/.ssh/kp_git_id_rsa.pub
? Title for your SSH key: kp_git_id_rsa
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 2F36-8BC4
Press Enter to open https://github.com/login/device in your browser... █
```

[https://github.com/login/device?skip\\_account\\_picker=true](https://github.com/login/device?skip_account_picker=true)





Follow the online prompts and authenticate your github account.

```
meera@Kangs-Home-Mac ~ % gh auth login

? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /Users/meera/.ssh/kp_git_id_rsa.pub
? Title for your SSH key: kp_git_id_rsa
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 2F36-8BC4
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ SSH key already existed on your GitHub account: /Users/meera/.ssh/kp_git_id_rsa.pub
✓ Logged in as kpassoubady
meera@Kangs-Home-Mac ~ %
```

## 6. Install GitHub Copilot Extension

You can use the below link to install

<https://docs.github.com/en/copilot/managing-copilot/configure-personal-settings/installing-github-copilot-in-the-cli>

In the terminal run the below command

**gh extension install github/gh-copilot**

```
meera@Kangs-Home-Mac ~ % gh extension install github/gh-copilot
✓ Installed extension github/gh-copilot
```

## 7. Install Node

- *Installing Node using Package Manager on Mac*

Go to the below link:

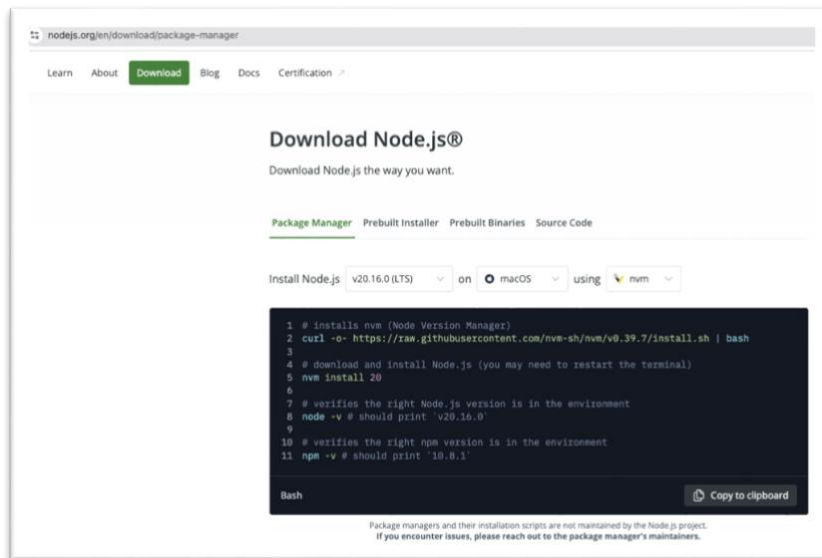
<https://nodejs.org/en/download/package-manager>

From the dropdown, select the node version, OS, and package manager and follow the instructions to install.

Select Node Version v20.x.y+ (recommended v20+)

Select your OS (Mac shown below)

Select your package manager (nvm is shown below)  
Follow the instructions provided to install.



The screenshot shows the Node.js download page for the Package Manager. The page has a navigation bar with links for Learn, About, Download (highlighted), Blog, Docs, and Certification. The main heading is "Download Node.js®" with the subtext "Download Node.js the way you want." Below this, there are tabs for Package Manager, Prebuilt Installer, Prebuilt Binaries, and Source Code. The Package Manager tab is active, showing a configuration section where "Install Node.js" is set to "v20.16.0 (LTS)", "on" is set to "macOS", and "using" is set to "nvm". Below this is a terminal window with the following commands:

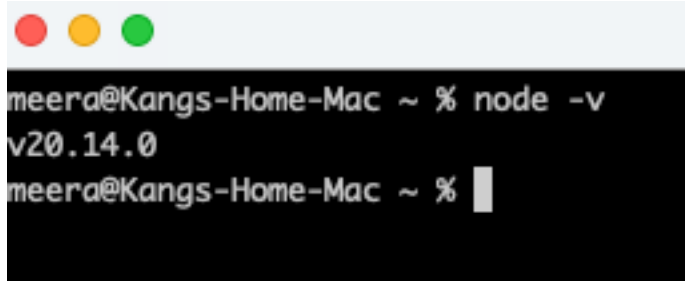
```
1 # installs nvm (Node Version Manager)
2 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
3
4 # download and install Node.js (you may need to restart the terminal)
5 nvm install 20
6
7 # verifies the right Node.js version is in the environment
8 node -v # should print 'v20.16.0'
9
10 # verifies the right npm version is in the environment
11 npm -v # should print '10.8.1'
```

At the bottom of the terminal window, there is a "Copy to clipboard" button. Below the terminal window, a disclaimer states: "Package managers and their installation scripts are not maintained by the Node.js project. If you encounter issues, please reach out to the package manager's maintainers."

- *Verify Node Exists*

To install TypeScript, you should have NodeJS installed first.  
Verify whether nodejs exists in your system.

In your terminal check **node -v**

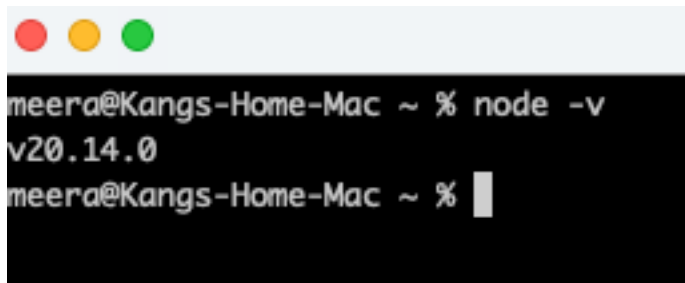
A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green). The terminal text shows the user 'meera' at the host 'Kangs-Home-Mac' in the directory '~'. The command 'node -v' has been entered, and the output is 'v20.14.0'. The prompt is ready for the next command.

```
meera@Kangs-Home-Mac ~ % node -v
v20.14.0
meera@Kangs-Home-Mac ~ %
```

Make sure if you have not already set in your PATH variable in your .bash\_profile add the below two lines at the end of your .bash\_profile file:

```
export LOCAL_BIN=/usr/local/bin
export PATH=./usr/bin:$LOCAL_BIN:$PATH
```

In your terminal check **node -v**

A screenshot of a macOS terminal window, identical to the one above. It shows the command 'node -v' being executed and the output 'v20.14.0' being displayed.

```
meera@Kangs-Home-Mac ~ % node -v
v20.14.0
meera@Kangs-Home-Mac ~ %
```

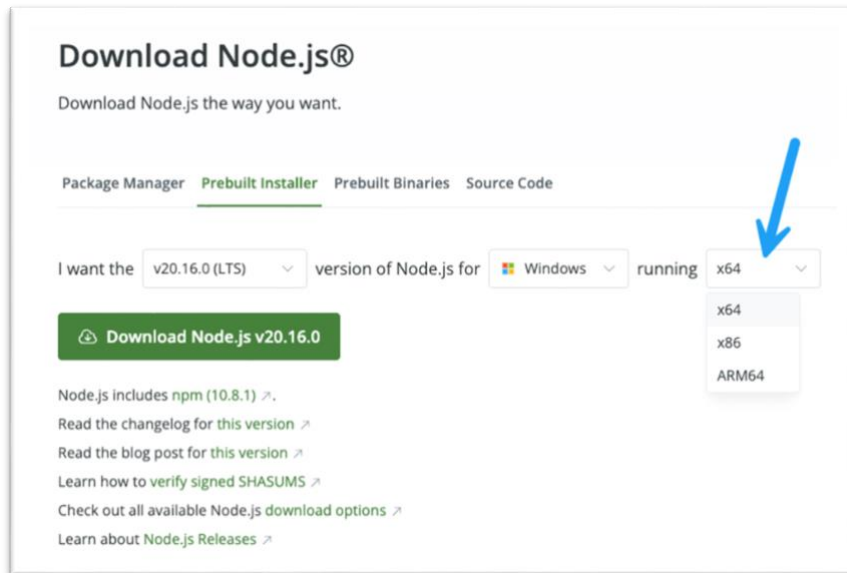
- *Install Node Using Prebuilt Installer for Windows*

Go to the below URL

<https://nodejs.org/en/download/prebuilt-installer>

Select from the list of available node versions, OS, and processor types, then download and install the prebuilt installer.



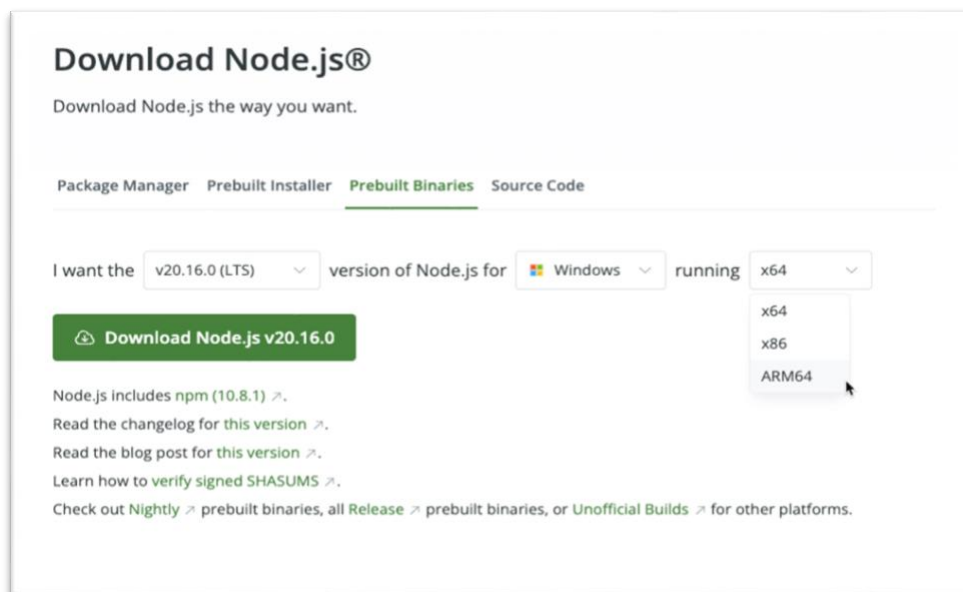


- *Installing Node using Prebuilt Binaries for Windows*

Go to the below URL:

<https://nodejs.org/en/download/prebuilt-binaries>

Select the node version, OS, and your processor type, download, and install.



# Project 1: Data Analysis Pipeline with Python and Pandas

## MacOS Install Guide

```
meera@Kangs-Home-Mac ~ % brew --version  
Homebrew 4.4.11
```

### Step 1: Install Python

Ensure Python 3 is installed (preferably version 3.8 or higher).

1. Install Python using Homebrew:

**brew install python**

Verify the Python installation:

**python3 --version**

```
meera ) python3 --version  
Python 3.9.6
```

**pip3 --version**

```
meera@Kangs-Home-Mac copilot % pip3 --version  
pip 24.3.1 from /Users/meera/Library/Python/3.9/lib/python/site-packages/pip (python 3.9)
```

### Step 2: Set Up a Virtual Environment

A virtual environment isolates dependencies for your project.

1. Install `virtualenv`:

**pip3 install virtualenv**

```
copilot ) pip3 install virtualenv  
Defaulting to user installation because normal site-packages is not writeable  
Requirement already satisfied: virtualenv in /Users/meera/Library/Python/3.9/lib/python/site-packages (20.25.3)  
Requirement already satisfied: distlib<1.0,=>0.3.7 in /Users/meera/Library/Python/3.9/lib/python/site-packages (from virtualenv) (0.3.8)  
Requirement already satisfied: filelock4,>=3.12.2 in /Users/meera/Library/Python/3.9/lib/python/site-packages (from virtualenv) (3.13.4)  
Requirement already satisfied: platformdirs<5,>=3.9.1 in /Users/meera/Library/Python/3.9/lib/python/site-packages (from virtualenv) (4.2.0)  
[ INFO ] A new release of pip is available: 24.0 -> 24.3.1  
[ INFO ] To update, run: /Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip
```

Create a virtual environment for the project:

**mkdir -p copilot/project1**

**cd copilot/project1**

**python3 -m venv data\_pipeline\_env**

```
copilot ) python3 -m venv data_pipeline_env
copilot ) ls -ltr
total 0
drwxr-xr-x  6 meera  staff  192 Dec 14 17:31 data_pipeline_env/
```

Activate the virtual environment:

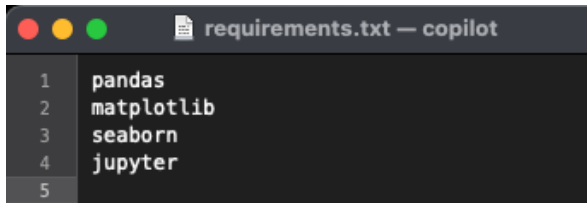
**source data\_pipeline\_env/bin/activate**

### Step 3: Install Required Python Libraries

Install the libraries needed for the project.

1. Create a `requirements.txt` file with the following content:

```
pandas
matplotlib
seaborn
jupyter
```



```
1 pandas
2 matplotlib
3 seaborn
4 jupyter
5
```

**pip3 install -r requirements.txt**

### Step 4: Install Jupyter Notebook

Jupyter Notebook will be used to write and present the summary report.

1. Install Jupyter Notebook:

**pip3 install notebook**

```
copilot ) pip install notebook
```

Collecting notebook  
Downloading notebook-7.3.1-py3-none-any.whl (13.2 MB)  
| ██ | 13.2 MB 5.2 MB/s

- ```
data_pipeline_env/bin/python3 -m pip install --upgrade pip
```

```
WARNING: You are using pip version 21.2.4; however, version 24.3.1 is available.
You should consider upgrading via the '/Users/meera/copilot/data_pipeline_env/bin/python3 -m pip install --upgrade pip' command.
copilot ) data_pipeline_env/bin/python3 -m pip install --upgrade pip
Requirement already satisfied: pip in ./data_pipeline_env/lib/python3.9/site-packages (21.2.4)
Collecting pip
  Downloading pip-24.3.1-py3-none-any.whl (1.8 MB)
    |██| 1.8 MB 6.7 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 21.2.4
    Uninstalling pip-21.2.4:
      Successfully uninstalled pip-21.2.4
Successfully installed pip-24.3.1
copilot )
```

- ## jupyter --version

```
(data_pipeline_env) meera@Kangs-Home-Mac copilot % jupyter --version

Selected Jupyter core packages...
IPython          : 8.18.1
ipykernel        : 6.29.5
ipywidgets       : 8.1.5
jupyter_client   : 8.6.3
jupyter_core     : 5.7.2
jupyter_server   : 2.14.2

/Users/meera/copilot/data_pipeline_env/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.
1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
jupyterlab       : 4.3.3
nbclient         : 0.10.1
nbconvert        : 7.16.4
nbformat         : 5.10.4
notebook         : 7.3.1
qtconsole        : not installed
traitlets        : 5.14.3
```

**pip3 list | grep jupyter**

```
(data_pipeline_env) meera@Kangs-Home-Mac copilot % pip3 list | grep jupyter
jupyter                1.1.1
jupyter_client         8.6.3
jupyter-console        6.6.3
jupyter_core           5.7.2
jupyter-events         0.10.0
jupyter-lsp            2.2.5
jupyter_server         2.14.2
jupyter_server_terminals 0.5.3
jupyterlab             4.3.3
jupyterlab_pygments    0.3.0
jupyterlab_server      2.27.3
jupyterlab_widgets     3.0.13
```

## Step 5: Test the Setup

1. Create a new Python script to test your environment:

### **touch test\_pipeline.py**

2. Open the script in VSCode using the terminal using the below command or directly from VSC:

### **code test\_pipeline.py**

3. Add the following lines of code to test Pandas and Matplotlib:

```
import pandas as pd
import matplotlib.pyplot as plt

print("Pandas and Matplotlib are working!")
```

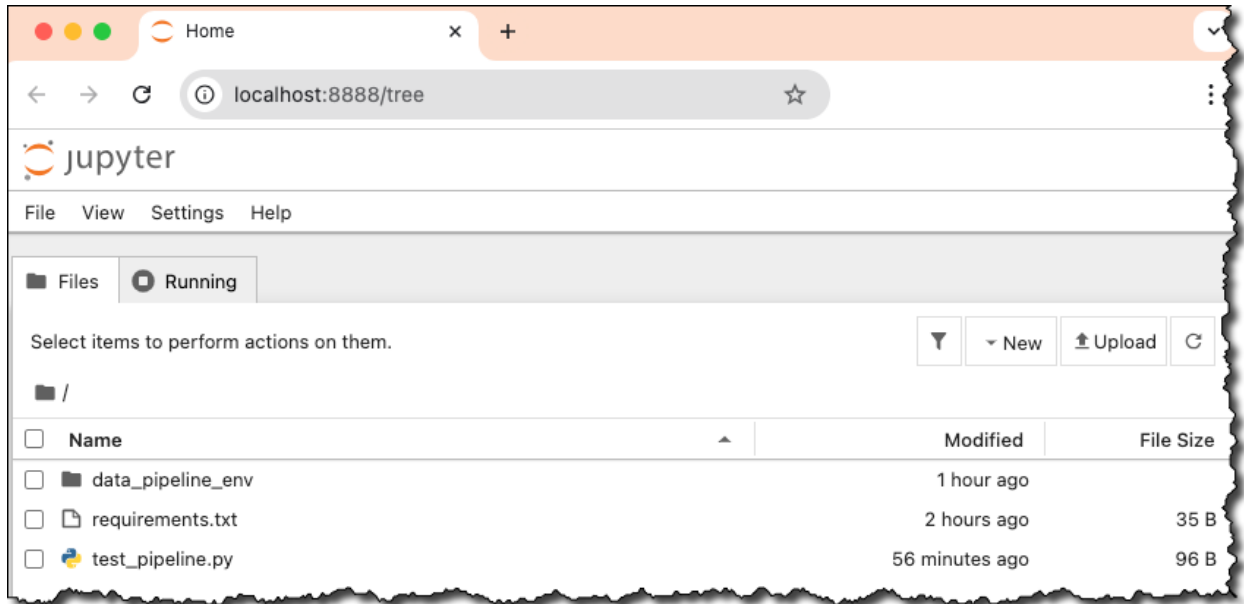
### **python test\_pipeline.py**

```
(data_pipeline_env) meera@Kangs-Home-Mac copilot % python ./test_pipeline.py
Pandas and Matplotlib are working!
```

## Step 6: Start Jupyter Notebook

1. Launch Jupyter Notebook:

### jupyter notebook



Press Ctrl+S

Shut down this Jupyter server (y/[n])? Y

## Step 7: Optional: Install Additional Tools

1. Install Pandas Profiling for Quick EDA

### **pip3 install pandas-profiling**

2. Install Seaborn for Advanced Visualizations

### **pip3 install seaborn**

# Project 2: Full-Stack Web Application with React and Node.js

## MacOS Install Guide

### Step 1: Install Homebrew

Provided the instructions under All Projects

### Step 2: Install Node.js and npm

Provided the instructions under All Projects

## Project Setup

### 1 Install Create React App

Create React App is a tool for bootstrapping React applications.

```
mkdir -p copilot/project2  
cd copilot/project2
```

1. Globally install `create-react-app`

```
npm install -g create-react-app  
create-react-app --version
```

```
project2 ) create-react-app --version  
5.0.1
```

### 2 Set Up the Frontend

1. Create a React project:

```
npx create-react-app frontend
```

```
project2 ) npx create-react-app frontend

Creating a new React app in /Users/meera/copilot/project2/frontend.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
```

2. Navigate into the project directory:

## cd frontend

```
project2 ) cd frontend/
frontend ) pwd
/Users/meera/copilot/project2/frontend
```

3. Install Axios in the React frontend:

```
npm install axios
```

4. Use Axios to make API calls to the backend in your React components.

```
backend ) pwd
/Users/meera/copilot/project2/backend
backend ) cd ../frontend
         npm install axios

added 4 packages, changed 1 package, and audited 1325 packages in 3s

265 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

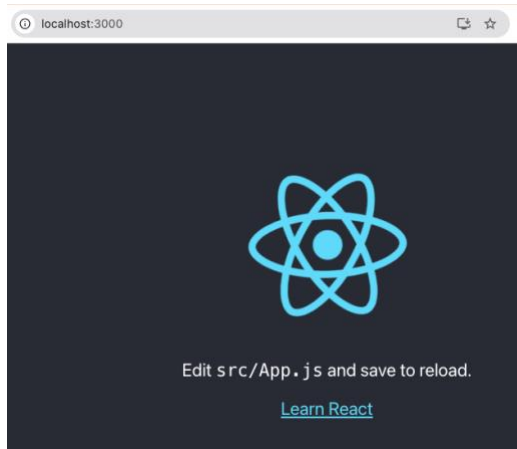
Run `npm audit` for details.
```

5. Start the development server:

## npm start

6. Confirm the React development server is running at <http://localhost:3000>





### 3 Set Up the Backend

1. Create a folder for the backend:

```
cd copilot/project2/  
mkdir backend  
cd backend
```

```
meera ) cd copilot/project2/  
project2 ) mkdir backend  
project2 ) cd backend/  
backend ) pwd  
/Users/meera/copilot/project2/backend  
backend )
```

2. Initialize a new Node.js project:

```
npm init -y
```

```
backend > npm init -y

Wrote to /Users/meera/copilot/project2/backend/package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

3. Install Express and other dependencies:

**npm install express body-parser cors jsonwebtoken bcryptjs dotenv**

```
backend > npm install express body-parser cors jsonwebtoken bcryptjs

added 89 packages, and audited 90 packages in 2s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

4. Install development dependencies:

**npm install --save-dev nodemon**

```
backend > npm install --save-dev nodemon

added 28 packages, and audited 118 packages in 2s

21 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

5. Create an entry file (e.g., index.js):

**touch index.js**

```
backend > touch index.js

backend > ls -ltr
total 120
drwxr-xr-x  116 meera  staff   3712 Dec 14 22:46 node_modules/
-rw-r--r--    1 meera  staff    431 Dec 14 22:46 package.json
-rw-r--r--    1 meera  staff  50072 Dec 14 22:46 package-lock.json
-rw-r--r--    1 meera  staff   330 Dec 14 22:51 index.js
```

## 4 Implement API Endpoints

1. Open `index.js` in a code editor (e.g., VSCode), Open the script in VSCode using the terminal using the below command or directly from VSC.

### code `index.js`

2. Add a basic Express server setup (`index.js`):

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
app.use(bodyParser.json());
app.use(cors());

app.get('/', (req, res) => {
  res.send('Backend is working!');
});

app.listen(5000, () => {
  console.log('Server is running on port 5000');
});
```

3. Run the backend server:

### node `index.js`

4. Confirm the server is running at <http://localhost:5000>.

```
backend ) node index.js
express:application set "x-powered-by" to true +0ms
express:application set "etag" to 'weak' +2ms
express:application set "etag fn" to [Function: generateETag] +0ms
express:application set "env" to 'develop' +0ms
express:application set "query parser" to 'extended' +1ms
express:application set "query parser fn" to [Function: parseExtendedQueryString] +0ms
express:application set "subdomain offset" to 2 +0ms
express:application set "trust proxy" to false +0ms
express:application set "trust proxy fn" to [Function: trustNone] +0ms
express:application booting in develop mode +0ms
express:application set "view" to [Function: View] +0ms
express:application set "views" to '/Users/meera/copilot/project2/backend/views' +0ms
express:application set "jsonp callback name" to 'callback' +0ms
express:router use '/' query +1ms
express:router:layer new '/' +0ms
express:router use '/' expressInit +0ms
express:router:layer new '/' +0ms
express:router use '/' jsonParser +0ms
express:router:layer new '/' +0ms
express:router use '/' corsMiddleware +1ms
express:router:layer new '/' +0ms
express:router:route new '/' +0ms
express:router:layer new '/' +0ms
express:router:route get '/' +0ms
express:router:layer new '/' +0ms
Server is running on port 5000
```

# Project 3: Machine Learning Model Deployment with Flask

## MacOS Install Guide

### Step 1: Install Homebrew

See installation instructions under All Projects

### Step 2: Install Python

Ensure Python 3 is installed (preferably version 3.8 or higher).

Install Python using Homebrew:

### **brew install python**

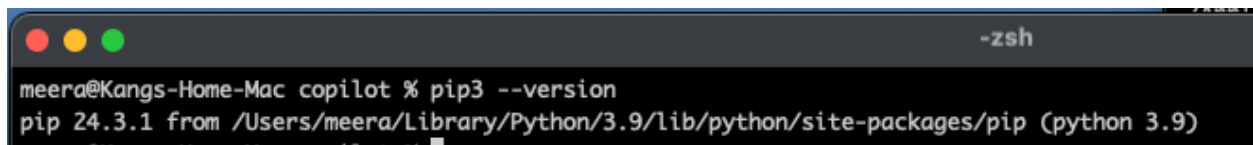
Verify the Python installation:

### **python3 --version**

A terminal window snippet showing the command 'python3 --version' being executed. The output is 'Python 3.9.6'. The prompt is 'meera >'.

```
meera > python3 --version
Python 3.9.6
```

### **pip3 --version**

A terminal window screenshot showing the command 'pip3 --version' being executed. The output is 'pip 24.3.1 from /Users/meera/Library/Python/3.9/lib/python/site-packages/pip (python 3.9)'. The prompt is 'meera@Kangs-Home-Mac copilot %'.

```
meera@Kangs-Home-Mac copilot % pip3 --version
pip 24.3.1 from /Users/meera/Library/Python/3.9/lib/python/site-packages/pip (python 3.9)
```

## Project Setup

### Step 1: Create a project folder

Open your terminal

### **mkdir -p copilot/project3**

### **cd copilot/project3**

```
meera ) cd copilot/project3
project3 ) ls -ltr
total 0
project3 ) pwd
/Users/meera/copilot/project3
```

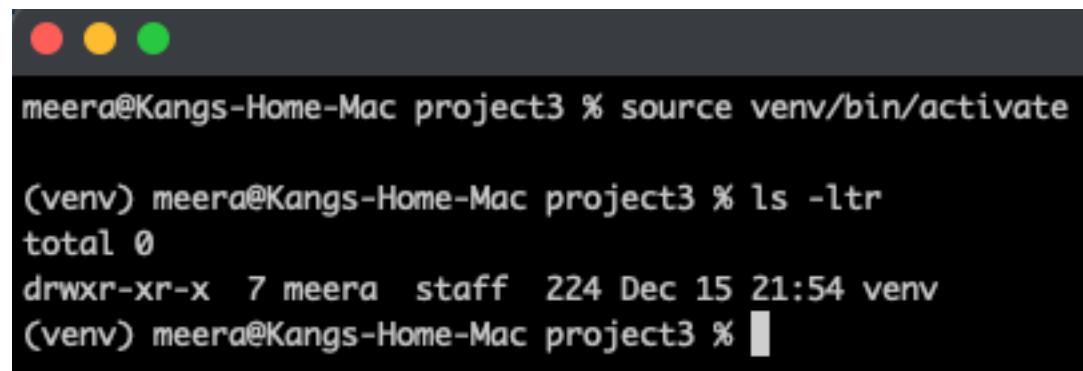
## Step 2: Create a Virtual Environment

Create a virtual environment in the project directory

**python3 -m venv venv**

Activate the virtual environment:

**source venv/bin/activate**

A terminal window with a dark background and light gray text. The window has a title bar with three colored circles (red, yellow, green) on the left. The text inside the terminal shows the user 'meera' at 'Kangs-Home-Mac' in the 'project3' directory. They run 'source venv/bin/activate', which changes the prompt to '(venv)'. Then they run 'ls -ltr', showing a directory listing for 'venv' with permissions 'drwxr-xr-x', size '7', owner 'meera', group 'staff', and date '224 Dec 15 21:54'. The prompt returns to '(venv) meera@Kangs-Home-Mac project3 %' with a cursor at the end.

```
meera@Kangs-Home-Mac project3 % source venv/bin/activate

(venv) meera@Kangs-Home-Mac project3 % ls -ltr
total 0
drwxr-xr-x  7 meera  staff  224 Dec 15 21:54 venv
(venv) meera@Kangs-Home-Mac project3 %
```

### Step 3: Create the requirements.txt File

Add the following dependencies in the requirements.txt

```
Flask==3.0.0  
pandas==2.0.3  
scikit-learn==1.3.2  
numpy==1.26.0  
joblib==1.3.2  
requests==2.31.0  
gunicorn==21.2.0
```

```
(venv) meera@Kangs-Home-Mac project3 % ls -ltr  
total 8  
drwxr-xr-x  7 meera  staff  224 Dec 15 21:54 venv  
-rw-r--r--  1 meera  staff  109 Dec 15 22:00 requirements.txt  
(venv) meera@Kangs-Home-Mac project3 %
```

### Step 4: Install Dependencies

Run the following command to install the dependencies:

**pip install -r requirements.txt**

## Step 5: Verify Installation

Check installed libraries:

### **pip freeze**

```
(venv) meera@Kangs-Home-Mac project3 % pip freeze

blinker==1.9.0
certifi==2024.12.14
charset-normalizer==3.4.0
click==8.1.7
Flask==3.0.0
gunicorn==21.2.0
idna==3.10
importlib_metadata==8.5.0
itsdangerous==2.2.0
Jinja2==3.1.4
joblib==1.3.2
MarkupSafe==3.0.2
numpy==1.26.0
packaging==24.2
pandas==2.0.3
python-dateutil==2.9.0.post0
pytz==2024.2
requests==2.31.0
scikit-learn==1.3.2
scipy==1.13.1
six==1.17.0
threadpoolctl==3.5.0
tzdata==2024.2
urllib3==2.2.3
Werkzeug==3.1.3
zipp==3.21.0
```

## Step 6: Add a Sample Dataset (sample\_data.csv)

Create a small CSV file to simulate training data:

```
feature1,feature2,target
1,2,0
2,3,1
3,4,0
4,5,1
```



## Step 7: Train the Model (`train_model.py`)

This script will preprocess the data, train a simple model, and save it.

```
import pandas as pd
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from joblib import dump

print(sklearn.__version__)

# Load data
data = pd.read_csv("sample_data.csv")
X = data[['feature1', 'feature2']]
y = data['target']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a simple model
model = LogisticRegression()
model.fit(X_train, y_train)

# Save the trained model
dump(model, 'model.pkl')
print("Model trained and saved as 'model.pkl'")
```

To train the model run in the terminal

```
python3 train_model.py
```

## Step 8: Create a Flask API (app.py)

This script loads the model and exposes a prediction endpoint.

```
from flask import Flask, request, jsonify
from joblib import load

app = Flask(__name__)

# Load the trained model
model = load("model.pkl")

@app.route("/")
def home():
    return "Flask API for Model Predictions!"

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    features = [data['feature1'], data['feature2']]
    prediction = model.predict([features])
    return jsonify({'prediction': int(prediction[0])})

if __name__ == '__main__':
    app.run(debug=True)
```

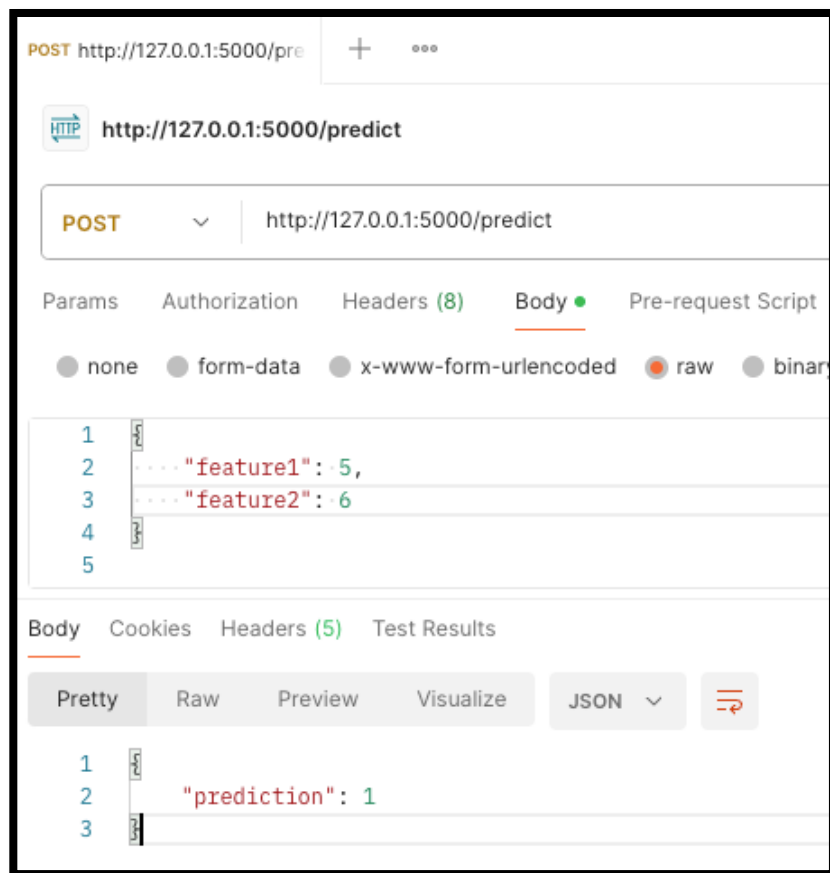
## Step 9: Test the API

Using cURL to test:

```
curl -X POST -H "Content-Type: application/json" -d '{"feature1":3, "feature2":4}'  
http://127.0.0.1:5000/predict
```

Send a POST request to the `/predict` endpoint with JSON data

Using Postman:



# Project 4: Automated Testing for Web Application using Selenium

## Step 1: Install Java Development Kit (JDK)

Open the terminal and install OpenJDK (any version above openjdk11+ is good, but keep 21 or latest 23)

### **brew install openjdk@21**

```
meera ) brew install openjdk@21
=> Downloading https://ghcr.io/v2/homebrew/core/openjdk/21/manifests/21.0.5
#####
=> Fetching openjdk@21
=> Downloading https://ghcr.io/v2/homebrew/core/openjdk/21/blobs/sha256:1f6b28a4ce616c847f4b6360ccc4632a3862ca8e
#####
=> Pouring openjdk@21--21.0.5.sonoma.bottle.tar.gz
=> Caveats
For the system Java wrappers to find this JDK, symlink it with
  sudo ln -sf /usr/local/opt/openjdk@21/libexec/openjdk.jdk /Library/Java/JavaVirtualMachines/openjdk-21.jdk

openjdk@21 is keg-only, which means it was not symlinked into /usr/local,
because this is an alternate version of another formula.

If you need to have openjdk@21 first in your PATH, run:
  echo 'export PATH="/usr/local/opt/openjdk@21/bin:$PATH"' >> ~/.zshrc

For compilers to find openjdk@21 you may need to set:
  export CPPFLAGS="-I/usr/local/opt/openjdk@21/include"
=> Summary
📦 /usr/local/Cellar/openjdk@21/21.0.5: 600 files, 329.5MB
=> Running `brew cleanup openjdk@21`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
```

**sudo ln -sf /usr/local/opt/openjdk@21/libexec/openjdk.jdk  
/Library/Java/JavaVirtualMachines/openjdk-21.jdk**

If you need to have openjdk@21 first in your PATH, run:  
**echo 'export PATH="/usr/local/opt/openjdk@21/bin:\$PATH"' >> ~/.zshrc**

For compilers to find openjdk@21 you may need to set:  
**export CPPFLAGS="-I/usr/local/opt/openjdk@21/include"**

Set JAVA\_HOME in your ~/.zshrc

**export JAVA\_HOME="/usr/libexec/java\_home -v 21"**

**source ~/.zshrc**

Validate java installed correctly

## java -version

```
meera@Kangs-Home-Mac ~ % java -version
openjdk version "21.0.5" 2024-10-15
OpenJDK Runtime Environment Homebrew (build 21.0.5)
OpenJDK 64-Bit Server VM Homebrew (build 21.0.5, mixed mode, sharing)
```

## Step 2: Install Maven

Install Maven using Homebrew:

## brew install maven

```
meera@Kangs-Home-Mac ~ % brew install maven
=> Downloading https://formulae.brew.sh/api/formula.jws.json
=> Downloading https://formulae.brew.sh/api/cask.jws.json
=> Downloading https://ghcr.io/v2/homebrew/core/maven/manifests/3.9.9
#####
=> Fetching maven
=> Downloading https://ghcr.io/v2/homebrew/core/maven/blobs/sha256:019b91415dce288368bd462ebbf009a262f7d9a4eb05f1bf64a4d
#####
=> Pouring maven--3.9.9.sonomac.bottle.tar.gz
  🍺  /usr/local/Cellar/maven/3.9.9: 94 files, 10.2MB
=> Running 'brew cleanup maven'...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see 'man brew').
meera@Kangs-Home-Mac ~ % mvn -v
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: /Users/meera/tools/apache-maven-3.8.5
Java version: 21.0.5, vendor: Homebrew, runtime: /usr/local/Cellar/openjdk@21/21.0.5/libexec/openjdk.jdk/Contents/Home
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "15.2", arch: "x86_64", family: "mac"
```

Pay attention which version is installed and add the below lines into your ~/.zshrc

**export M2\_HOME=/usr/local/Cellar/maven/3.9.9**

**export M2=\$M2\_HOME/bin**

**export PATH=\$M2:\$PATH**

Validate Maven installed correctly

## mvn -v

```
1 meera@Kangs-Home-Mac ~ % mvn -v
2 Apache Maven 3.9.9 (8e8579a9e76f7d015eeSec7bfc97d260186937)
Maven home: /usr/local/Cellar/maven/3.9.9/libexec
Java version: 21.0.5, vendor: Homebrew, runtime: /usr/local/Cellar/openjdk@21/21.0.5/libexec/openjdk.jdk/Contents/Home
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "15.2", arch: "x86_64", family: "mac"
```

## Project Setup

Open your terminal

```
mkdir -p copilot/project4
cd copilot/project4
```

Create a maven default project using the below command

```
mvn archetype:generate -DgroupId=com.amgen.app -DartifactId=ui-tests -
-DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.5 -
-DinteractiveMode=false
```

```
meera@Kangs-Home-Mac project4 % mvn archetype:generate -DgroupId=com.amgen.app -DartifactId=ui-tests -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.5 -DinteractiveMode=false

[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/3.1.2/maven-install-plugin-3.1.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/3.1.2/maven-install-plugin-3.1.2.pom (8.5 kB at 17 kB/s)
```

```
[INFO] -----
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.5
[INFO] -----
[INFO] Parameter: groupId, Value: com.amgen.app
[INFO] Parameter: artifactId, Value: ui-tests
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.amgen.app
[INFO] Parameter: packageInPathFormat, Value: com/amgen/app
[INFO] Parameter: junitVersion, Value: 5.11.0
[INFO] Parameter: package, Value: com.amgen.app
[INFO] Parameter: groupId, Value: com.amgen.app
[INFO] Parameter: artifactId, Value: ui-tests
[INFO] Parameter: javaCompilerVersion, Value: 17
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[WARNING] Don't override file /Users/meera/copilot/project4/ui-tests/src/main/java/com/amgen/app
[WARNING] Don't override file /Users/meera/copilot/project4/ui-tests/src/test/java/com/amgen/app
[WARNING] CP Don't override file /Users/meera/copilot/project4/ui-tests/.mvn
[INFO] Project created from Archetype in dir: /Users/meera/copilot/project4/ui-tests
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.675 s
[INFO] Finished at: 2024-12-16T22:50:07-05:00
[INFO] -----
meera@Kangs-Home-Mac project4 %
```

The above command creates bunch of folders and the default maven project

Open the project4 in VSCode, edit the pom.xml and add the below dependencies in the existing pom.xml before the closing </dependencies>

```
.gitignore project1 pom.xml U AppTest.java U
project4 > ui-tests > pom.xml > project > dependencies
32
33 <dependencies>
34 <dependency>
35 <groupId>org.junit.jupiter</groupId>
36 <artifactId>junit-jupiter-api</artifactId>
37 <scope>test</scope>
38 </dependency>
39 <!-- Optionally: parameterized tests support -->
40 <dependency>
41 <groupId>org.junit.jupiter</groupId>
42 <artifactId>junit-jupiter-params</artifactId>
43 <scope>test</scope>
44 </dependency>
45
46 <!-- Selenium WebDriver -->
47 <dependency>
48 <groupId>org.seleniumhq.selenium</groupId>
49 <artifactId>selenium-java</artifactId>
50 <version>4.14.0</version>
51 </dependency>
52
```

```
<!-- Selenium WebDriver -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.14.0</version>

</dependency>

<!-- TestNG for testing -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.8.0</version>

<scope>test</scope>

</dependency>

<!-- WebDriver Manager to simplify driver management -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.7.0</version>

</dependency>

<!-- Jackson Databind (for JSON handling if needed) -->

<dependency>
```

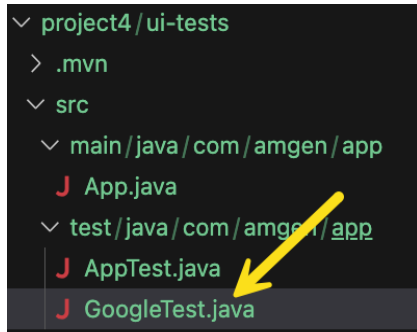
```

<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.15.2</version>
</dependency>

<!-- Logging (SLF4J and Logback) -->
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-api</artifactId>
<version>2.0.7</version>
</dependency>
<dependency>
<groupId>ch.qos.logback</groupId>
<artifactId>logback-classic</artifactId>
<version>1.4.11</version>
</dependency>

```

In the VSCode goto the folder project4/ui-tests/src/test/java/com/amgen/app directory  
Create GoogleTest.java



Add the below code in the GoogleTest.java

```

package com.amgen.app;

import org.testng.annotations.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;

import io.github.bonigarcia.wdm.WebDriverManager;

```



```
public class GoogleTest {  
    @Test  
    public void validatePageTitle() {  
        System.out.println("validatePageTitle");  
        WebDriverManager.chromedriver().setup();  
        WebDriver driver = new ChromeDriver();  
        driver.get("https://google.com");  
        System.out.println("Page title is: " + driver.getTitle());  
        Assert.assertEquals(driver.getTitle(), "Google");  
        driver.quit();  
    }  
}
```

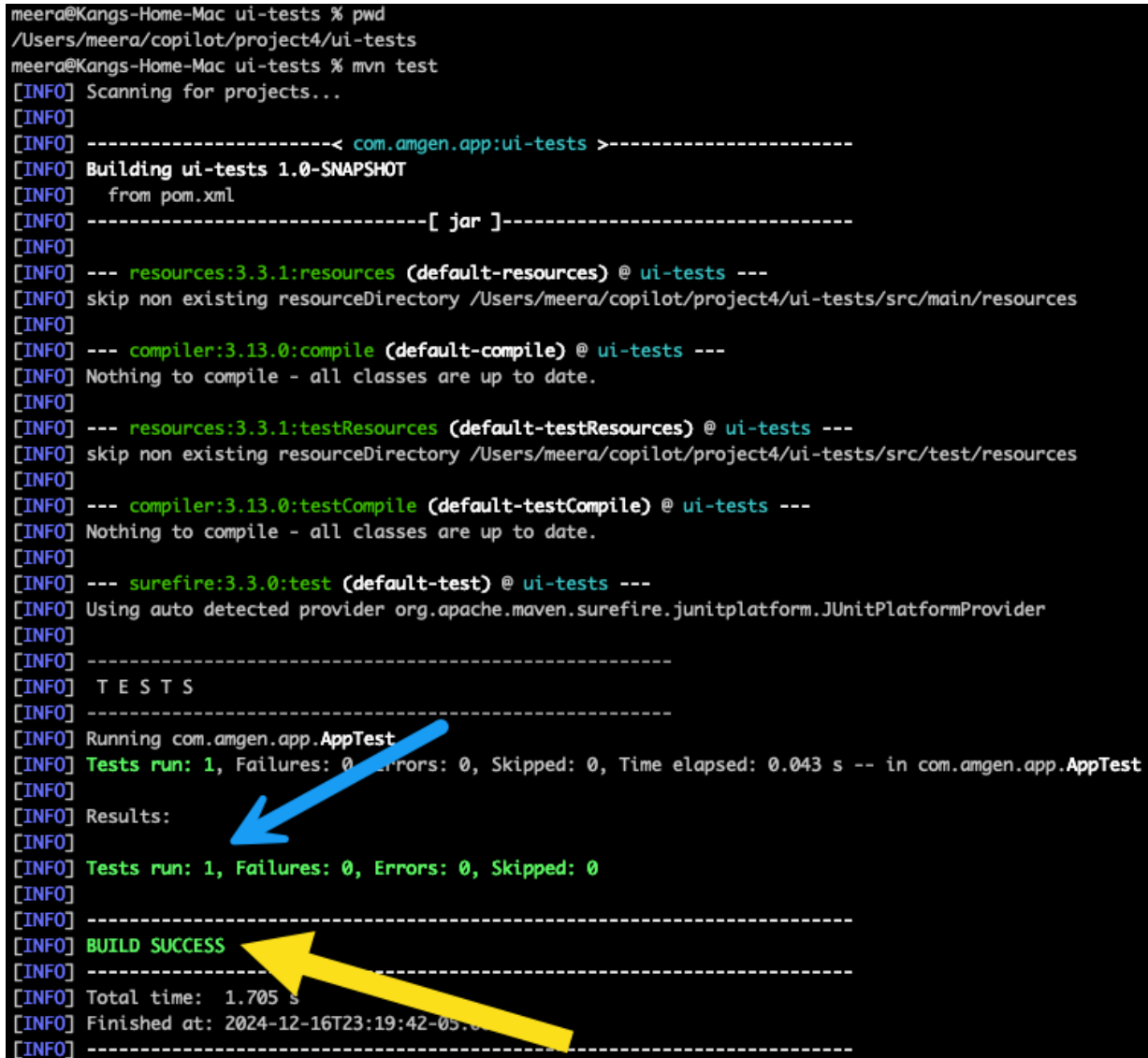
In the terminal run the below mvn command

## **cd project4/ui-tests**

## **mvn test**

All the dependencies will get downloaded and then the build will be success as shown below

```
meera@Kangs-Home-Mac ui-tests % pwd
/Users/meera/copilot/project4/ui-tests
meera@Kangs-Home-Mac ui-tests % mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.amgen.app:ui-tests >-----
[INFO] Building ui-tests 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ ui-tests ---
[INFO] skip non existing resourceDirectory /Users/meera/copilot/project4/ui-tests/src/main/resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ ui-tests ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ ui-tests ---
[INFO] skip non existing resourceDirectory /Users/meera/copilot/project4/ui-tests/src/test/resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ ui-tests ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- surefire:3.3.0:test (default-test) @ ui-tests ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.amgen.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.043 s -- in com.amgen.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.705 s
[INFO] Finished at: 2024-12-16T23:19:42-0500
[INFO]
[INFO] -----
```



Note: In your organization if you need **settings.xml** to download the artifact, get that file and copy into /Users/meera/.m2

/Users/\$USER/.m2

Check with your DevOps team for more information.

# Project 5: Serverless ETL Pipeline

## Step 1: Install Python

Ensure Python 3 is installed (preferably version 3.8 or higher).

Install Python using Homebrew (check All Projects section how to install Homebrew):

### brew install python

Verify the Python installation:

### python3 --version

```
meera ) python3 --version
Python 3.9.6
```

### pip3 --version

```
meera@Kangs-Home-Mac copilot % pip3 --version
pip 24.3.1 from /Users/meera/Library/Python/3.9/lib/python/site-packages/pip (python 3.9)
```

## Step 2: Install AWS CLI

Use the below link to install

<https://docs.aws.amazon.com/cli/v1/userguide/install-macos.html>

### pip3 install awscli --upgrade --user

```
meera@Kangs-Home-Mac ~ % pip3 install awscli --upgrade --user
Collecting awscli
  Using cached awscli-1.36.23-py3-none-any.whl.metadata (1.3 kB)
Requirement already satisfied: botocore==1.35.82 in /Library/Python/3.9/lib/python/site-packages (from awscli) (1.35.82)
Requirement already satisfied: docutils<0.17,>=0.10 in /Library/Python/3.9/lib/python/site-packages (from awscli) (0.16)
Requirement already satisfied: s3transfer<0.11.0,>=0.10.0 in /Library/Python/3.9/lib/python/site-packages (from awscli) (0.10.4)
Requirement already satisfied: PyYAML<6.1,>=3.10 in /Library/Python/3.9/lib/python/site-packages (from awscli) (6.0.1)
Requirement already satisfied: colorama<0.4.7,>=0.2.5 in /Library/Python/3.9/lib/python/site-packages (from awscli) (0.4.6)
Requirement already satisfied: rsa<4.8,>=3.1.2 in /Library/Python/3.9/lib/python/site-packages (from awscli) (4.7.2)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /Library/Python/3.9/lib/python/site-packages (from botocore==1.35.82->awscli) (1.0.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /Library/Python/3.9/lib/python/site-packages (from botocore==1.35.82->awscli) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /Library/Python/3.9/lib/python/site-packages (from botocore==1.35.82->awscli) (1.26.15)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.2 in /Library/Python/3.9/lib/python/site-packages (from rsa<4.8,>=3.1.2->awscli) (0.6.0)
Requirement already satisfied: six<1.17.0,>=1.10 in /Library/Developer/CommandLineTools/Library/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/site-packages (from urllib3<1.27,>=1.25.4->awscli) (1.15.0)
Using cached awscli-1.36.23-py3-none-any.whl (4.5 MB)
Installing collected packages: awscli
Successfully installed awscli-1.36.23
```

## Project Setup

Create the project folder

```
mkdir -R copilot/project5  
cd copilot/project5
```

Initialize node project

```
npm init -y
```

```
meera@Kangs-Home-Mac ~ % cd copilot  
meera@Kangs-Home-Mac copilot % cd project5  
meera@Kangs-Home-Mac project5 % pwd  
/Users/meera/copilot/project5  
meera@Kangs-Home-Mac project5 % npm init -y  
Wrote to /Users/meera/copilot/project5/package.json:  
  
{  
  "name": "project5",  
  "version": "1.0.0",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "description": ""  
}
```

Install necessary packages

```
npm install aws-sdk @prisma/client prisma dotenv
```

```
meera@Kangs-Home-Mac project5 % npm install aws-sdk @prisma/client prisma dotenv  
npm warn deprecated querystring@0.2.0: The querystring API is considered Legacy.  
  
added 49 packages, and audited 50 packages in 10s  
  
17 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

Install Prisma CLI globally

```
npm install -g prisma
```

```
meera@Kangs-Home-Mac project5 % npm install -g prisma  
  
added 6 packages in 4s  
meera@Kangs-Home-Mac project5 %
```

# Project 6: Developing Microservices Architecture with Docker and K8s

## MacOS Install Guide

### Step 1: Install Homebrew

See install instructions provided under All Projects section

### Step 2: Install Node.js and npm

See Install instructions provided under All Projects section

### Step 3: Install Docker and Docker Compose

1. Download Docker Desktop from the official Docker website.

<https://docs.docker.com/desktop/setup/install/mac-install/>

<https://docs.docker.com/desktop/setup/install/windows-install/>

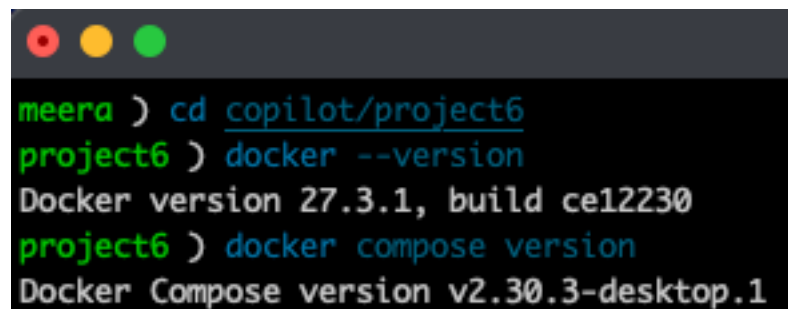
2. Follow the installation steps based on your OS.

`docker --version`

`docker compose version`

(for older version use `docker-compose --version`)

**Note:** Docker Desktop includes Docker Compose

A terminal window with a dark background and light green text. The prompt is 'meera >'. The user enters 'cd copilot/project6'. The prompt changes to 'project6 >'. The user enters 'docker --version'. The output is 'Docker version 27.3.1, build ce12230'. The user enters 'docker compose version'. The output is 'Docker Compose version v2.30.3-desktop.1'.

```
meera > cd copilot/project6
project6 > docker --version
Docker version 27.3.1, build ce12230
project6 > docker compose version
Docker Compose version v2.30.3-desktop.1
```

### Step 4: Install Kubernetes CLI (kubectl)

Use the below link and follow the installation for your OS

<https://kubernetes.io/docs/tasks/tools/>

**brew install kubectl**

```
project6 ) brew install kubectl
==> Auto-updating Homebrew...
Adjust how often this is run with HOMEBREW_
HOMEBREW_NO_AUTO_UPDATE. Hide these hints w
==> Auto-updated Homebrew!
Updated 2 taps (homebrew/core and homebrew/
==> New Formulae
bender          glaze          gurk
==> New Casks
soundsource@test
```

Verify kubectl installation

**kubectl version --client**

```
meera@Kangs-Home-Mac ~ % kubectl version --client
Client Version: v1.32.0
Kustomize Version: v5.5.0
```

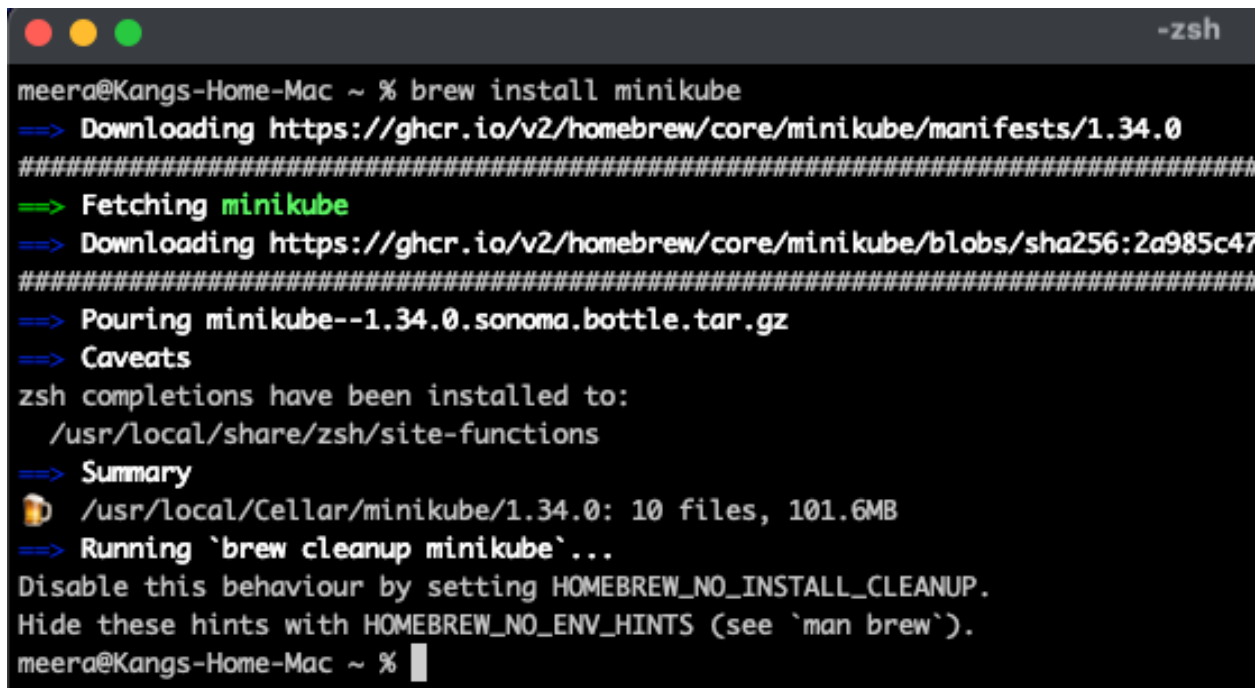
## Step 5: Install Minikube (for Local Kubernetes Cluster)

Follow the below link to install MiniKube for your OS

<https://minikube.sigs.k8s.io/docs/start/?arch=%2Fmacos%2Fx86-64%2Fstable%2Fbinary+download>

For macOS:

**brew install minikube**



```
meera@Kangs-Home-Mac ~ % brew install minikube
=> Downloading https://ghcr.io/v2/homebrew/core/minikube/manifests/1.34.0
#####
=> Fetching minikube
=> Downloading https://ghcr.io/v2/homebrew/core/minikube/blobs/sha256:2a985c47
#####
=> Pouring minikube--1.34.0.sonoma.bottle.tar.gz
=> Caveats
zsh completions have been installed to:
  /usr/local/share/zsh/site-functions
=> Summary
📦 /usr/local/Cellar/minikube/1.34.0: 10 files, 101.6MB
=> Running `brew cleanup minikube`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
meera@Kangs-Home-Mac ~ %
```

Start minikube:

## minikube start

```
meera@Kangs-Home-Mac ~ % brew install minikube
--> Downloading https://ghcr.io/v2/homebrew/core/minikube/manifests/1.34.0
--> Fetching minikube
--> Downloading https://ghcr.io/v2/homebrew/core/minikube/manifests/1.34.0
--> Pouring minikube--1.34.0
--> Caveats
zsh completions have been installed to: /usr/local/share/zsh/site-functions
--> Summary
📦 /usr/local/Cellar/minikube/1.34.0
--> Running 'brew cleanup minikube'
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_HINTS_IN_TERM.
meera@Kangs-Home-Mac ~ % minikube start
🐳 minikube v1.34.0 on Darwin 15.2
🔧 Automatically selected the docker driver
🚀 Using Docker Desktop driver with root privileges
👉 Starting "minikube" primary control-plane node in "minikube" cluster
📦 Pulling base image v0.0.45 ...
📦 Downloading Kubernetes v1.31.0 preload ...
> gcr.io/k8s-minikube/kicbase...: 187.56 MiB / 487.90 MiB 22.05% 16.09 MiB
```

```
meera@Kangs-Home-Mac ~ % minikube start
🐳 minikube v1.34.0 on Darwin 15.2
🔧 Automatically selected the docker driver
🚀 Using Docker Desktop driver with root privileges
👉 Starting "minikube" primary control-plane node in "minikube" cluster
📦 Pulling base image v0.0.45 ...
📦 Downloading Kubernetes v1.31.0 preload ...
> gcr.io/k8s-minikube/kicbase...: 487.90 MiB / 487.90 MiB 100.00% 26.22 M
> preloaded-images-k8s-v18-v1...: 326.69 MiB / 326.69 MiB 100.00% 16.39 M
🔥 Creating docker container (CPUs=2, Memory=7890MB) ...
📦 Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
  ▪ Generating certificates and keys ...
  ▪ Booting up control plane ...
  ▪ Configuring RBAC rules ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔍 Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: default-storageclass, storage-provisioner
👉 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
meera@Kangs-Home-Mac ~ %
```

## minikube status

```
meera@Kangs-Home-Mac ~ % minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```



## Project Setup

Create the project folder

**mkdir -p copilot/project6**

**cd copilot/project6**

Initialize a new Node.js project

**npm init -y**

```
fish /Users/meera/copilot/project6
project6 ) pwd
/Users/meera/copilot/project6
project6 ) npm init -y
Wrote to /Users/meera/copilot/project6/package.json:

{
  "name": "project6",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Install Fastify CLI globally

**npm install --global fastify-cli**

```
project6 ) npm i -g fastify-cli
npm warn deprecated inflight@1.0.6:
d way to coalesce async requests by
npm warn deprecated glob@8.1.0: Glo
added 110 packages in 3s
```

**fastify version**

```
project6 ) fastify version
7.2.0
```

Install fastify locally for your project

`npm i -D fastify`

```
project6 > npm i -D fastify
added 46 packages, and audited 47 packages in 3s
2 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

Verify Fastify installation

```
project6 > npm list fastify
project6@1.0.0 /Users/meera/copilot/project6
└─ fastify@5.2.0
```