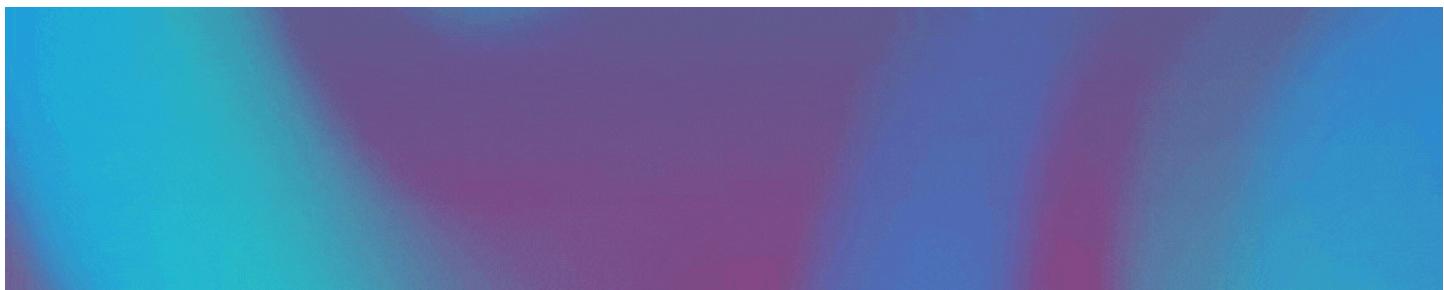


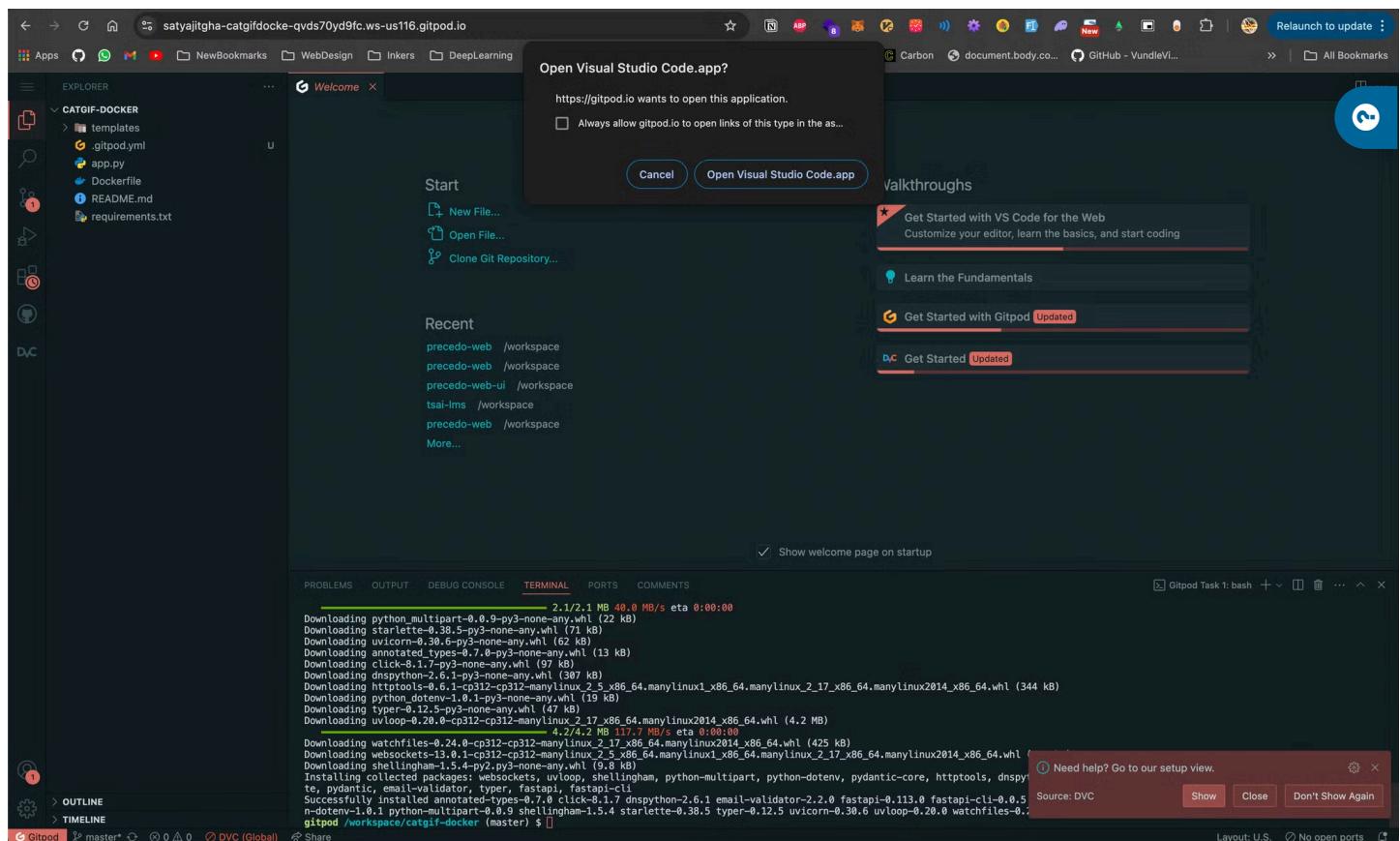
Session 03 - Docker - II

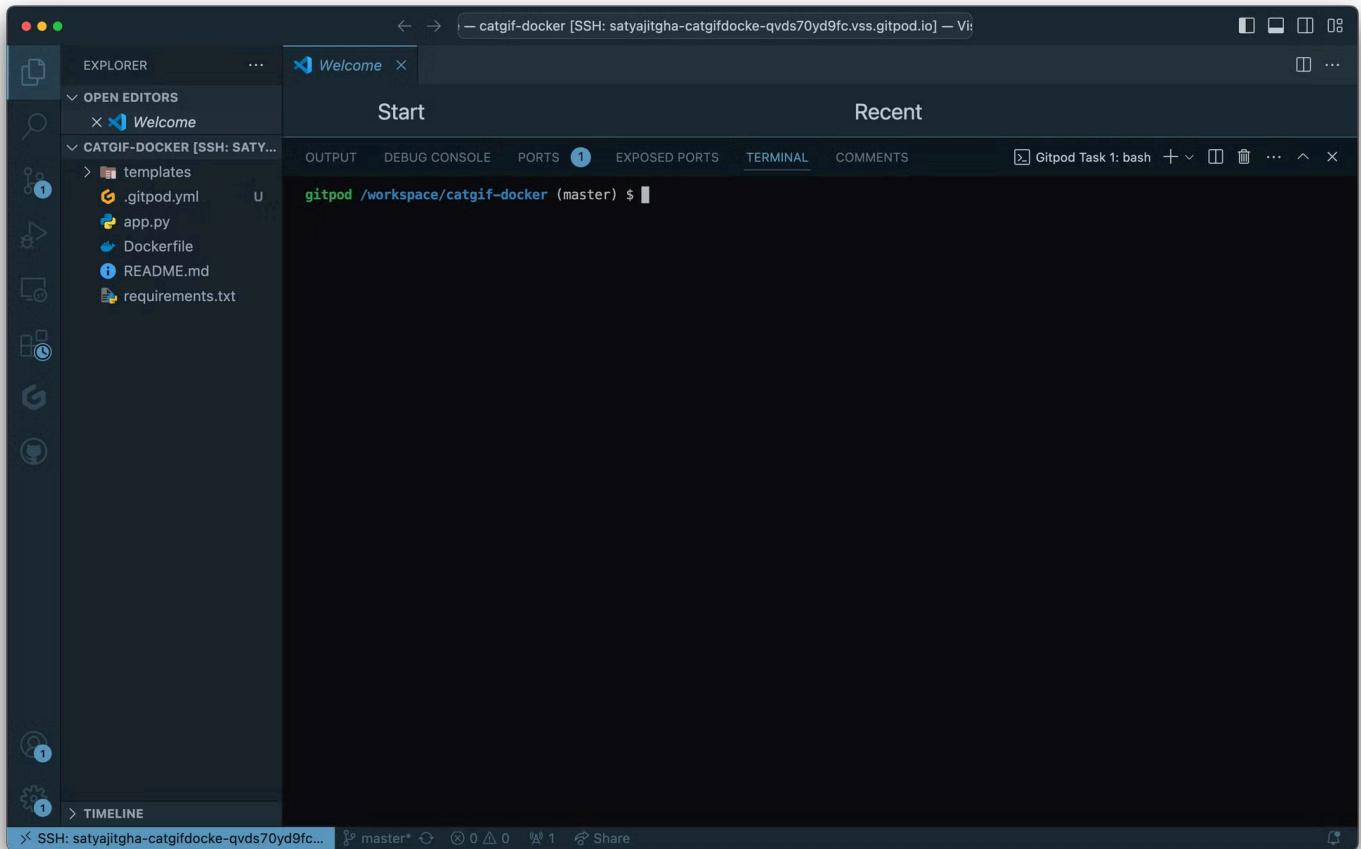
Start Assignment

- Due Saturday by 23:59
- Points 500
- Submitting a text entry box



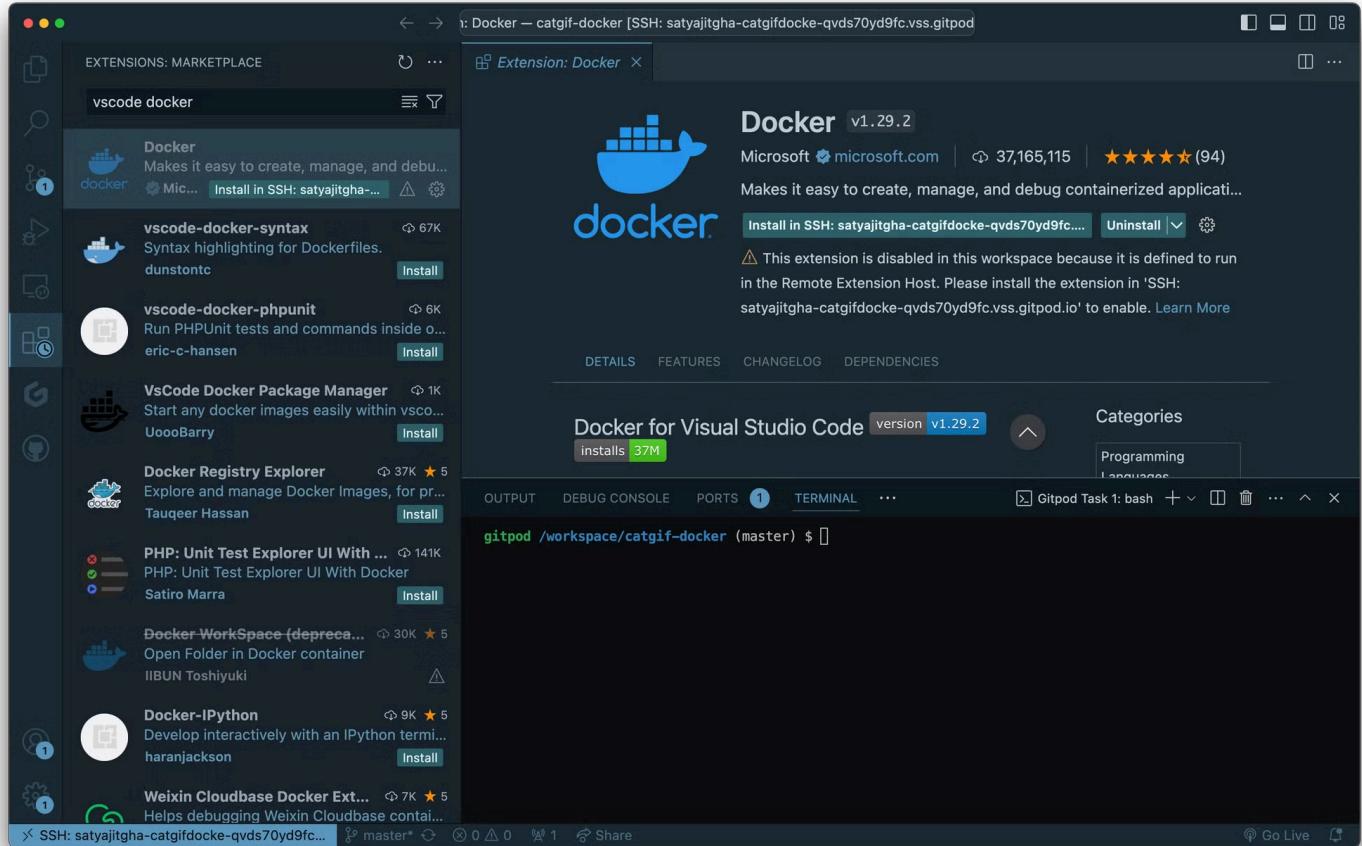
Docker VSCode





Install the docker extension in VSCode [ms-azuretools.vscode-docker](#)

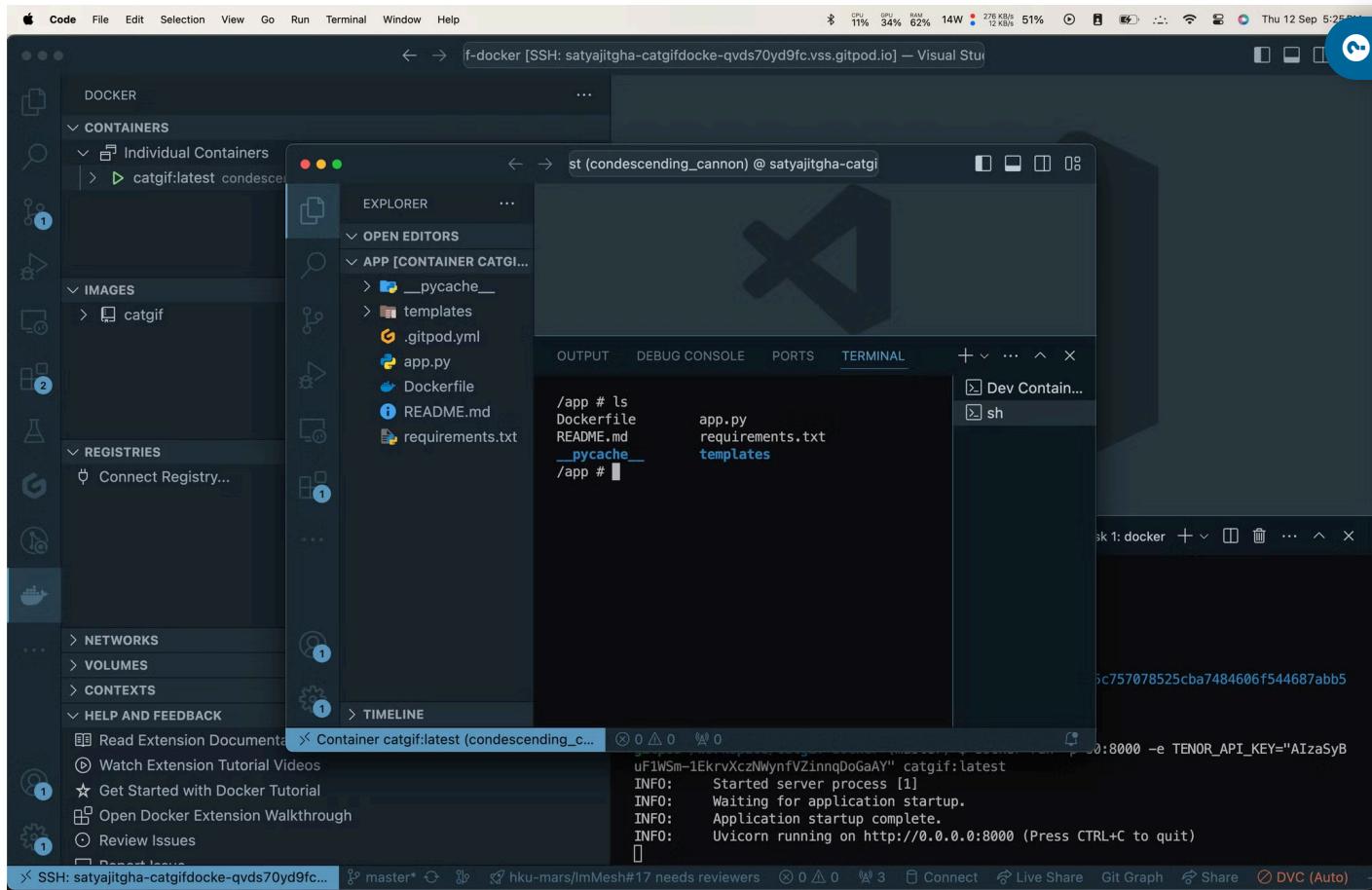
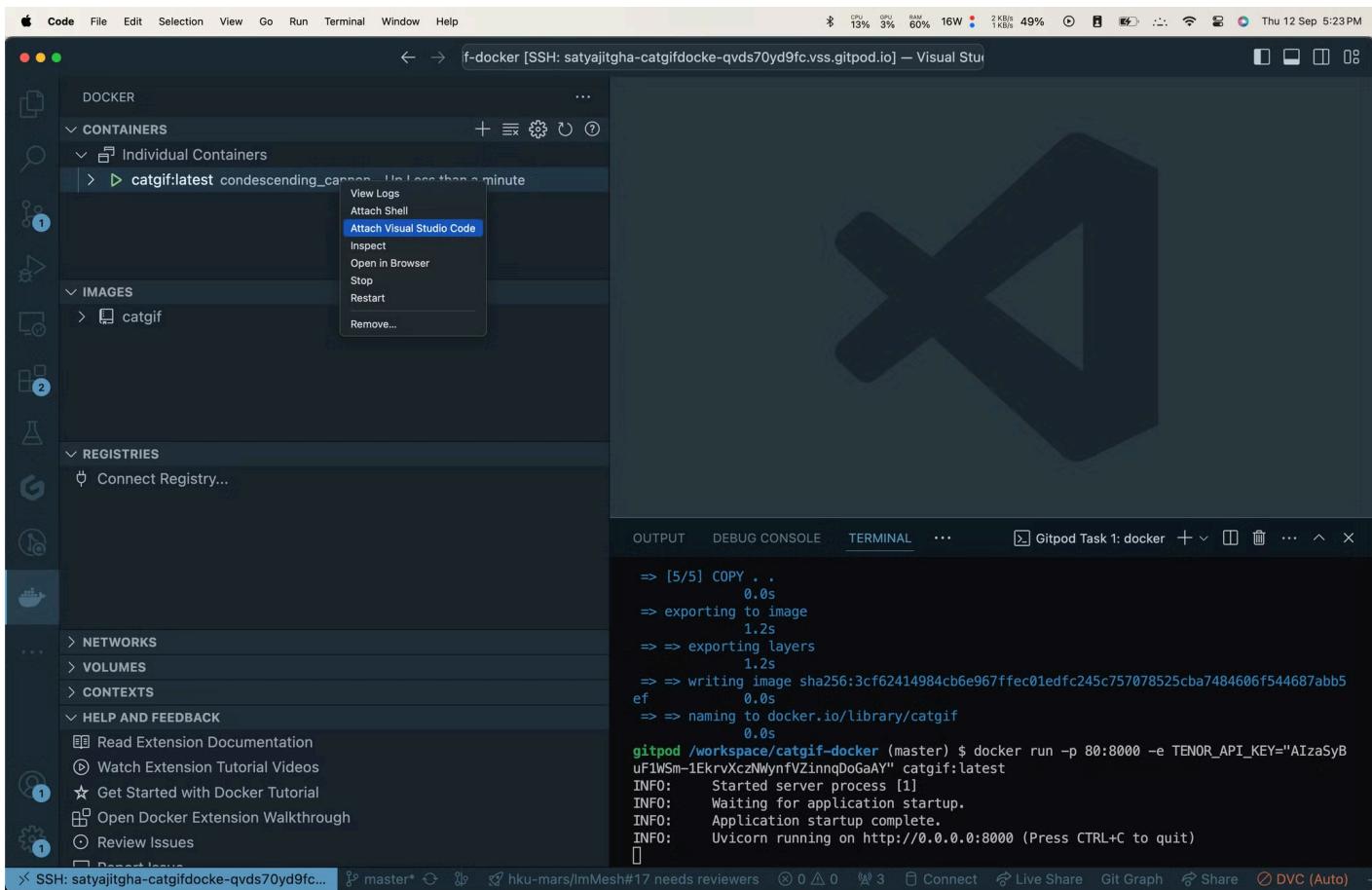




```
git clone <https://github.com/satyajitghana/catgif-docker>
```

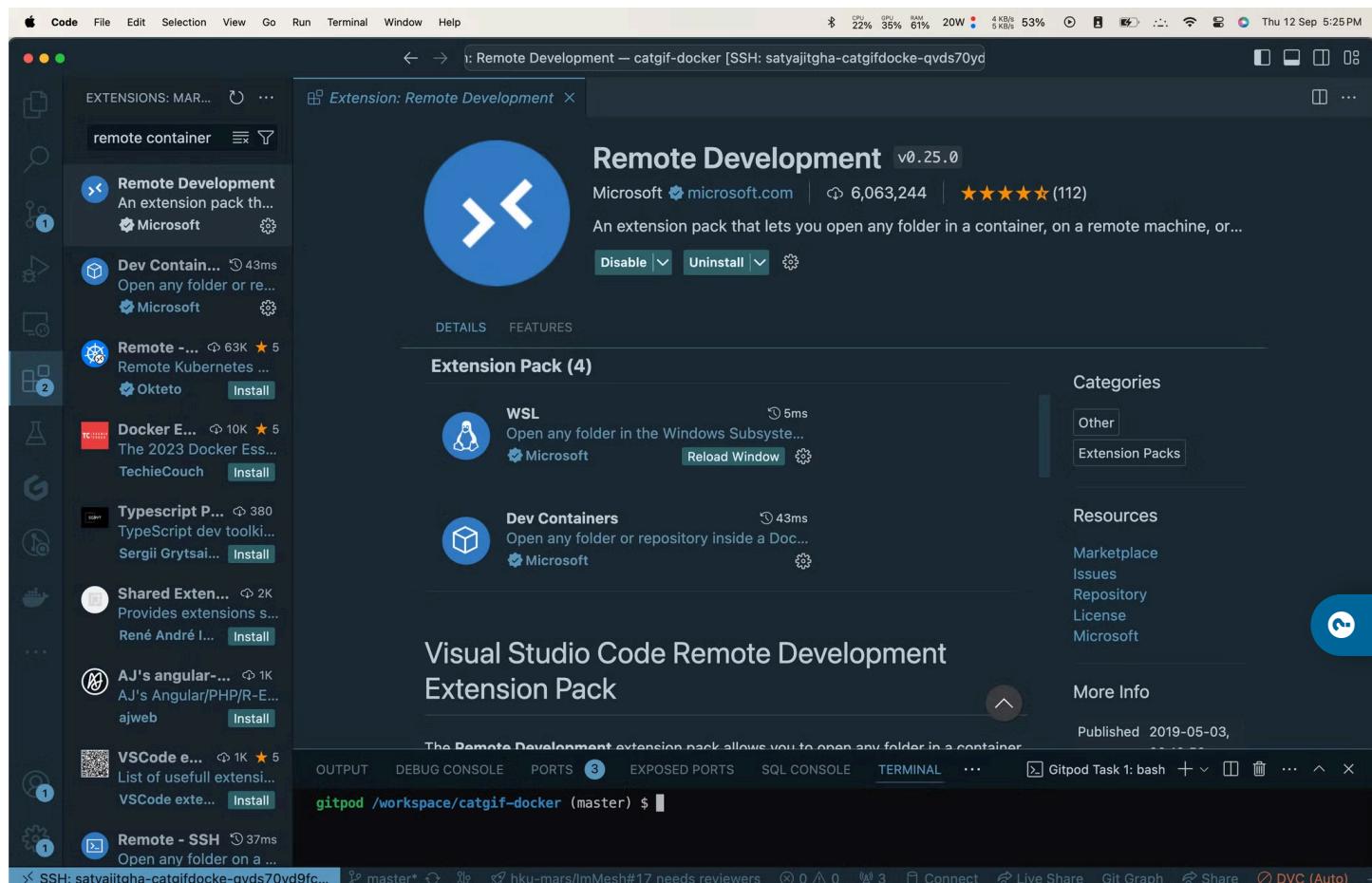
```
docker build -t catgif .
```

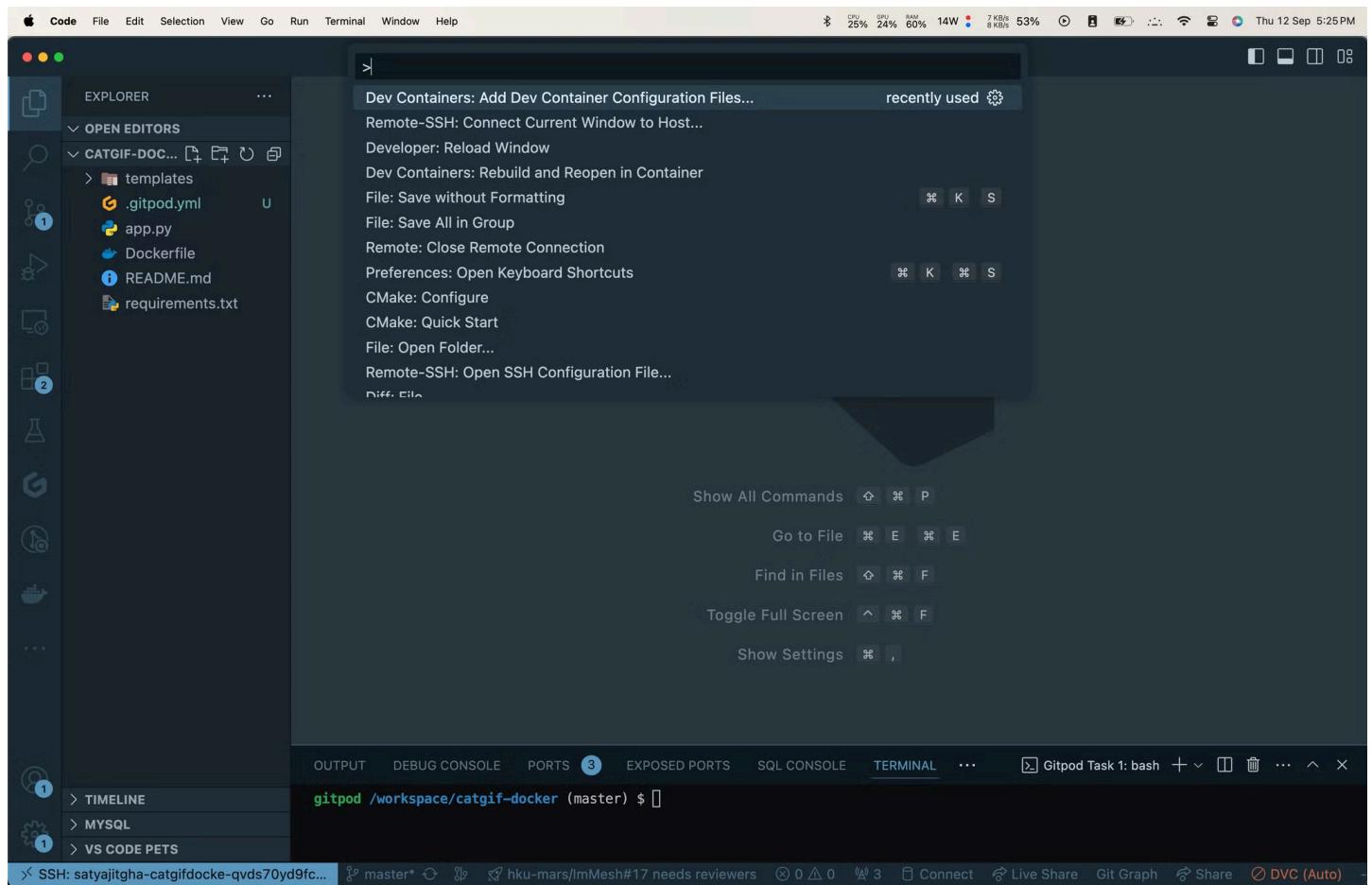
```
docker run -p 80:8000 -e TENOR_API_KEY="AIzaSyBuF1WSm-1EkrvXczNWynfVZinnqDoGaAY" catgif:latest
```



DevContainer

Extension [ms-vscode-remote.remote-containers](#)





```
// For format details, see <https://aka.ms/devcontainer.json>. For config options, see the
// README at: <https://github.com/devcontainers/templates/tree/main/src/python>
{
    "name": "Python 3",
    // Or use a Dockerfile or Docker Compose file. More info: <https://containers.dev/guide/dockerfiles>
    "image": "mcr.microsoft.com/devcontainers/python:1-3.11-bookworm"

    // Features to add to the dev container. More info: <https://containers.dev/features>.
    // "features": {},

    // Use 'forwardPorts' to make a list of ports inside the container available locally.
    // "forwardPorts": [],

    // Use 'postCreateCommand' to run commands after the container is created.
    // "postCreateCommand": "pip3 install --user -r requirements.txt",

    // Configure tool-specific properties.
    // "customizations": {},

    // Uncomment to connect as root instead. More info: <https://aka.ms/dev-containers-non-root>.
    // "remoteUser": "root"
}
```

```
{
    "name": "CatGIF",
    // Or use a Dockerfile or Docker Compose file. More info: <https://containers.dev/guide/dockerfiles>
```

```
// "image": "mcr.microsoft.com/devcontainers/python:0-3.11",
"build": {
  "dockerfile": "../Dockerfile",
  "context": ".."
},
// Features to add to the dev container. More info: <https://containers.dev/features>.
// "features": {},  

// Configure tool-specific properties.
"customizations": {
  // Configure properties specific to VS Code.
  "vscode": {
    "settings": {},
    "extensions": [
      "ms-python.python"
    ]
  }
},
// Use 'forwardPorts' to make a list of ports inside the container available locally.
// "forwardPorts": [9000],  

// Use 'portsAttributes' to set default properties for specific forwarded ports.
// More info: <https://containers.dev/implementors/json_reference/#port-attributes>
"portsAttributes": {
  "8000": {
    "label": "Flask Backend",
    "onAutoForward": "notify"
  }
},
// Use 'postCreateCommand' to run commands after the container is created.
// "postCreateCommand": "pip3 install -r requirements.txt"  

// Uncomment to connect as root instead. More info: <https://aka.ms/dev-containers-non-root>.
// "remoteUser": "root"
}
```

Code File Edit Selection View Go Run Terminal Window Help

CPU 8% GPU 11% RAM 60% 12W 28 KB/s 60% Thu 12 Sep 5:29 PM

EXPLORER OPEN EDITORS CATGIF-DOCKER [SSH: SATYA... .devcontainer devcontainer.json .devcontainer.json .gitpod.yml app.py Dockerfile README.md requirements.txt

Dev Containers: Rebuild and Reopen in Container .NET: Rebuild Azure Container Registry: Build Image in Azure... Bazel: Refresh Bazel Build Targets Dev Containers: Rebuild Without Cache and Reopen in Container rust-analyzer: Rebuild proc macros and build scripts

```
11 },
10   "forwardPorts": [
19     8000
18 ],
17   "customizations": {
16     "vscode": {
15       "settings": {},
14       "extensions": [
13         "ms-python.python"
12       ]
11     }
10   }
9 }
```

OUTPUT DEBUG CONSOLE PORTS EXPOSED PORTS SQL CONSOLE TERMINAL Gitpod Task 1: bash

gitpod /workspace/catgif-docker (master) \$

TIMELINE MYSQL VS CODE PETS

SSH: satyajitgtha-catgifdocke-qvds70yd9fc... master* hku-mars/lmMesh#17 needs reviewers 0 ▲ 0 Connect Live Share Git Graph Share DVC (Auto)

Code File Edit Selection View Go Run Terminal Window Help

CPU 30% GPU 30% RAM 60% 18W 28 KB/s 60% Thu 12 Sep 5:30 PM

EXPLORER OPEN EDITORS CATGIF-DOCKER [DEV CONTAINER: CatGIF Service @ satyajitgtha-catgifdocke-qvds70yd9fc.vss]

Dev Containers

```
[16652 ms]
[16652 ms]
[16652 ms] Start: Run in container: sed -i -E 's/((^|\s)PATH=)([^$]*$)/\1${PATH:-\3}/g' /etc/profile || true
[16653 ms]
[16653 ms]
[16696 ms] Start: Run: docker inspect --type container 5695b7457ac1bbf1dfcc66490e01d1e2a21aa29cc76e133a33ba10e547982d83
[17378 ms] Start: Run: docker exec -i -u root 5695b7457ac1bbf1dfcc66490e01d1e2a21aa29cc76e133a33ba10e547982d83 /bin/sh -c echo "New
container started. Keep-alive process started." ; export VSCODE_REMOTE_CONTAINERS_SESSION=bee87eab-1bfe-4b1c-a8f4-8a22b46b68651726
142415022 ; /bin/sh
[17378 ms] Start: Inspecting container
[17379 ms] Start: Run: docker inspect --type container 5695b7457ac1bbf1dfcc66490e01d1e2a21aa29cc76e133a33ba10e547982d83
[18063 ms] Start: Run in container: /bin/sh
[18123 ms] New container started. Keep-alive process started.
```

Opening Remote... 0 ▲ 0 0

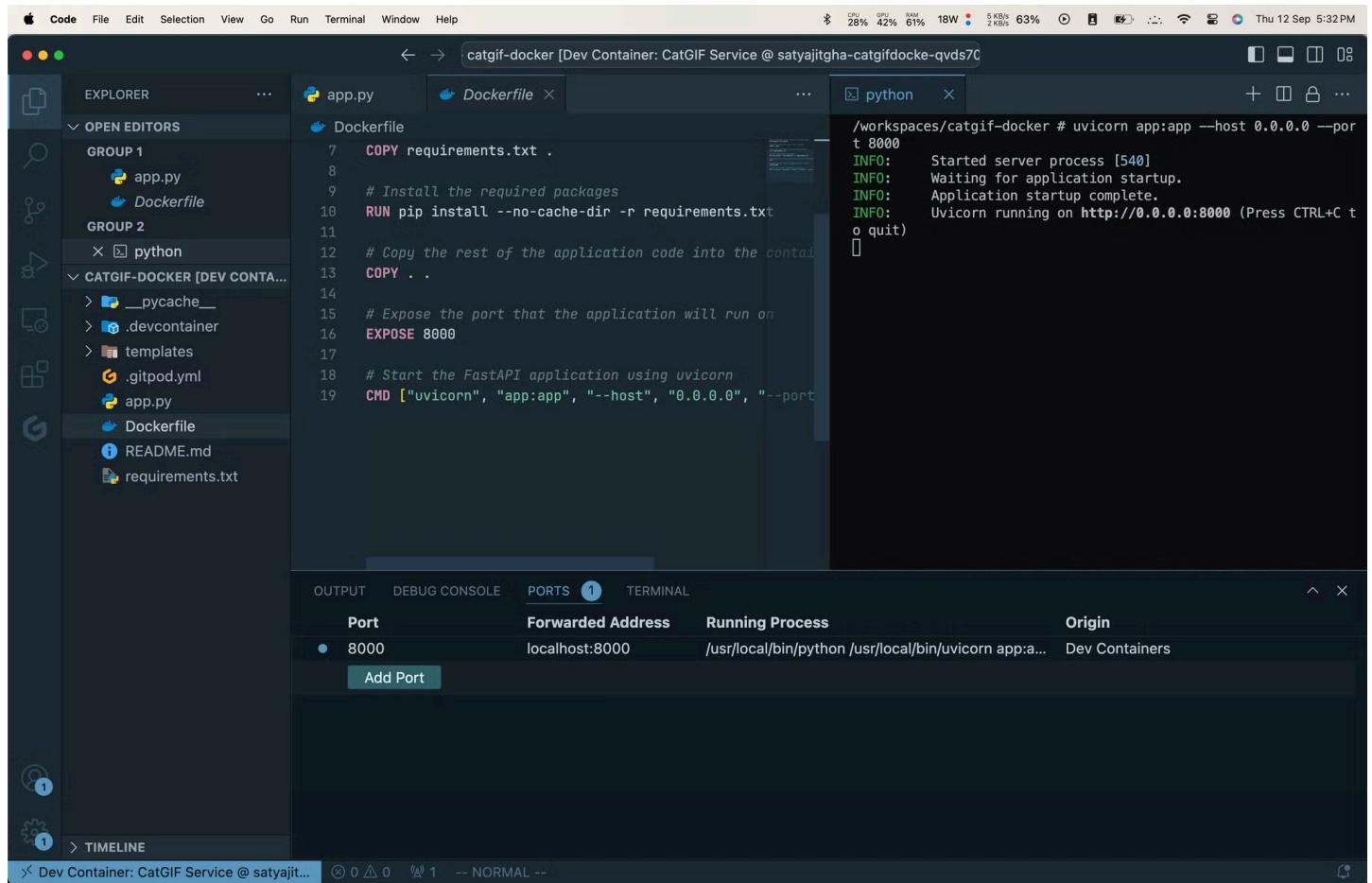
The screenshot shows the VS Code interface with a Python FastAPI application running in a Docker container. The Explorer sidebar on the left lists files and folders, including `app.py`, `.devcontainer`, `templates`, `.gitpod.yml`, `Dockerfile`, `README.md`, and `requirements.txt`. The `app.py` file is open in the editor, displaying code for a FastAPI application that searches for cat GIFs using the Tenor API. A tooltip in the bottom right corner asks if the user wants to install the recommended 'Python' extension from Microsoft. The status bar at the bottom shows the current workspace, file statistics, and terminal settings.

```
import os
from fastapi import FastAPI, Request
from fastapi.templating import Jinja2Templates
from fastapi.responses import HTMLResponse
import httpx
import random

app = FastAPI()
templates = Jinja2Templates(directory="templates")

TENOR_API_KEY = os.getenv("TENOR_API_KEY")
TENOR_API_URL = "https://tenor.googleapis.com/v2/search"

async def get_cat_gif_url():
    async with httpx.AsyncClient() as client:
        params = {
            "q": "cat",
            "key": TENOR_API_KEY,
            "client_key": "my_test_app",
            "limit": 50, # Fetch 50 results
            "media_filter": "gif",
        }
```



<https://github.com/satyajitghana/catgif-docker/tree/devcontainer> ↗
(<https://github.com/satyajitghana/catgif-docker/tree/devcontainer>)

Installing packages in Dockerfile

```
RUN apt-get update && export DEBIAN_FRONTEND=noninteractive \
&& apt-get -y install git
```

Introduction to Docker Compose

Docker Compose is a tool that allows you to define and manage multi-container Docker applications. It uses YAML files to configure the application's services and with a single command, you can create and start all the services specified in your configuration.

Understanding Docker Compose

Before we delve deeper into Docker Compose, it's crucial to understand the basic components of Docker Compose, which include:

- Dockerfile:** This is a text file that contains all commands needed to build a Docker image.
- Docker Images:** These are read-only templates used to create containers. Images are created from Dockerfiles and are stored in Docker registries.
- Docker Containers:** These are runnable instances of Docker images.
- Docker Compose File (docker-compose.yml):** This YAML file describes your app's services and defines which software, services, and tools are needed to run them.

Key Features of Docker Compose

- Multiple isolated environments on a single host
- Preserve volume data when containers are created
- Only recreate containers that have changed
- Variables and moving a composition between environments

Docker Compose Workflow

The Docker Compose tool follows a simple three-step process:

- Define your app's environment with a Dockerfile:** This file helps Docker to build the image of your application.
- Define the services that make up your app in docker-compose.yml:** This file specifies all the services that your application needs, such as databases, queues, caches, etc.
- Run docker-compose up and Compose starts and runs your entire app:** This command starts the whole system. If the images are not yet built or up-to-date, it builds them using the Dockerfile. It then (re)creates the containers for the services defined in the docker-compose.yml file.

Migrate to Docker Compose

```
git clone https://github.com/satyajitghana/catgif-docker  (https://github.com/satyajitghana/catgif-docker)
cd catgif-docker
git checkout origin/compose
```

docker-compose.yml

```
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 3000:8000
    depends_on:
      - redis
```

```
redis:
  image: redis:latest
  volumes:
    - redis-data:/data
volumes:
  redis-data:
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files in the project: `__pycache__`, `.devcontainer`, `templates`, `.gitpod.yml`, `app.py`, `docker-compose.yml`, `Dockerfile`, `README.md`, and `requirements.txt`.
- EDITOR:** Displays the `docker-compose.yml` file content:

```
version: '3'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 3000:8000
    depends_on:
      - redis
  redis:
    image: redis:latest
    volumes:
      - redis-data:/data
volumes:
  redis-data:
```

- OUTPUT:** Shows the logs from the `redis` service:

```
redis-1 | 1:C 12 Sep 2024 12:08:54.914 * o000o000o000 Redis is starting o000o000c0000
redis-1 | 1:C 12 Sep 2024 12:08:54.914 * Redis version=7.4.0, bits=64, commit=00000000, modified=0, pid=1, just
started
redis-1 | 1:C 12 Sep 2024 12:08:54.914 # Warning: no config file specified, using the default config. In order
to specify a config file use redis-server /path/to/redis.conf
redis-1 | 1:M 12 Sep 2024 12:08:54.915 * monotonic clock: POSIX clock_gettime
redis-1 | 1:M 12 Sep 2024 12:08:54.916 * Running mode=standalone, port=6379.
redis-1 | 1:M 12 Sep 2024 12:08:54.916 * Server initialized
redis-1 | 1:M 12 Sep 2024 12:08:54.916 * Ready to accept connections tcp
web-1 | INFO: Started server process [1]
web-1 | INFO: Waiting for application startup.
web-1 | INFO: Application startup complete.
web-1 | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

- TERMINAL:** Shows the command `redis-server` running.

```
docker compose build
```

```
docker compose up
```

Docker Volumes

```
docker volume create hello
```

```
docker run -it --rm -v hello:/data ubuntu bash
```

Let's see what is inside the redis volume

```
gitpod /workspace/catgif-docker (devcontainer) $ docker volume ls
DRIVER      VOLUME NAME
local       catgif-docker_redis-data
local       vscode
```

```
docker run -it --rm -v catgif-docker_redis-data:/data ubuntu bash
```

DevContainer with Docker Compose

[https://containers.dev/guide/dockerfile#docker-compose-dockerfile ↗](https://containers.dev/guide/dockerfile#docker-compose-dockerfile)
(<https://containers.dev/guide/dockerfile#docker-compose-dockerfile>)

Training, Eval & Deploy Example

[https://github.com/satyajitghana/mnist-fastapi-compose ↗](https://github.com/satyajitghana/mnist-fastapi-compose)
(<https://github.com/satyajitghana/mnist-fastapi-compose>)

```
git clone https://github.com/satyajitghana/mnist-fastapi-compose ↗ (https://github.com/satyajitghana/mnist-fastapi-compose)
```

```
version: '3.8'

services:
  train:
    build:
      context: model-train
      dockerfile: Dockerfile.train
    volumes:
      - ./model:/workspace/model
      - ./data:/workspace/data
    restart: on-failure

  evaluate:
    build:
      context: model-eval
      dockerfile: Dockerfile.eval
    volumes:
      - ./model:/workspace/model
      - ./data:/workspace/data
    restart: on-failure

  server:
    build:
      context: model-deploy
      dockerfile: Dockerfile.serve
    volumes:
      - ./model:/app/model
    ports:
      - "8000:80"
    restart: unless-stopped

  infer:
```

```
build:
  context: model-inference
  dockerfile: Dockerfile.infer
volumes:
  - ./data:/workspace/data
  - ./requests:/workspace/requests
network_mode: "host"
restart: no
```

```
docker compose run train
```

```
SSH: satyajitgha-mnistfastap-m0okk5unjsa... master 0% hku-mars/lmMesh#17 needs reviewers 0 0 0 0 9 2 Connect Live Share Git Graph Share DVC (Auto)
```

The screenshot shows a terminal window with the following output:

```
2023-09-12T20:08:54.000Z [train.py:1] INFO: Starting training loop...
2023-09-12T20:08:54.000Z [train.py:1] INFO: Model: CNN
2023-09-12T20:08:54.000Z [train.py:1] INFO: Device: CPU
2023-09-12T20:08:54.000Z [train.py:1] INFO: Optimizer: Adam
2023-09-12T20:08:54.000Z [train.py:1] INFO: Learning Rate: 0.001
2023-09-12T20:08:54.000Z [train.py:1] INFO: Training Data: MNIST
2023-09-12T20:08:54.000Z [train.py:1] INFO: Validation Data: MNIST
2023-09-12T20:08:54.000Z [train.py:1] INFO: Batch Size: 64
2023-09-12T20:08:54.000Z [train.py:1] INFO: Epochs: 10
2023-09-12T20:08:54.000Z [train.py:1] INFO: Save Interval: 1 epoch
2023-09-12T20:08:54.000Z [train.py:1] INFO: Dry Run: False
2023-09-12T20:08:54.000Z [train.py:1] INFO: Loss Function: CrossEntropyLoss
2023-09-12T20:08:54.000Z [train.py:1] INFO: Metric: Accuracy
2023-09-12T20:08:54.000Z [train.py:1] INFO: Model Checkpoint: model_checkpoint.pth
2023-09-12T20:08:54.000Z [train.py:1] INFO: Training Start Time: 2023-09-12T20:08:54.000Z
2023-09-12T20:08:54.000Z [train.py:1] INFO: Training End Time: 2023-09-12T20:08:54.000Z
2023-09-12T20:08:54.000Z [train.py:1] INFO: Training Duration: 0 seconds
2023-09-12T20:08:54.000Z [train.py:1] INFO: Training Loss: 2.314632
2023-09-12T20:08:54.000Z [train.py:1] INFO: Training Accuracy: 0.000000
2023-09-12T20:08:54.000Z [train.py:1] INFO: Validation Loss: 1.155138
2023-09-12T20:08:54.000Z [train.py:1] INFO: Validation Accuracy: 0.000000
```

```
docker compose run evaluate
```

A screenshot of the VS Code interface. The top bar shows the title "md — mnist-fastapi-compose [SSH: satyajitgha-mnistfastap-m0okk5unjsa.vss.git]" and system status icons. The left sidebar (EXPLORER) lists project files: docker-compose.yml, README.md, train.py, Dockerfile.train, data, MNIST, .gitkeep, docker, Dockerfile.devel, model, .gitkeep, eval_results.json, mnist_cnn.pt, model-deploy, Dockerfile.serve, serve.py, model-eval, check_eval.py, Dockerfile.eval, evaluate.py, model-inference, Dockerfile.infer, infer.py, model-train, check_train.py, and a timeline entry for MySQL. The right side features a terminal window with a dark theme, showing a session in progress:

```
# mnist-fastapi-compose
# mnist-fastapi-compose
An example of MNIST training, eval and deployment with FastAPI and PyTorch and Docker Compose (Not for production usage)
```
... bash
docker compose run train
docker compose run evaluate
docker compose run --service-ports server
...
... bash
docker compose run infer
```
The terminal also displays a log of Docker build steps and a gitpod command.
```

docker compose run --service-ports server

docker compose run infer

```
INFO: 172.18.0.1:38060 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38066 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38080 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38088 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38094 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38108 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38124 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38134 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38140 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38112 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38132 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:37982 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38166 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38184 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38146 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38196 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38198 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38192 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38210 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38002 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38144 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38154 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38202 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38182 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38020 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38230 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38240 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38254 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38226 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38268 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38276 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38292 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38306 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38314 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38328 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38334 - "POST /predict/ HTTP/1.1" 200 OK
INFO: 172.18.0.1:38336 - "POST /predict/ HTTP/1.1" 200 OK
Completed request 2158
Completed request 9929
Completed request 1695
Completed request 5159
Completed request 6528
Completed request 9493
Completed request 6671
Completed request 9025
Completed request 7973
Completed request 542
Completed request 101
Completed request 7986
Completed request 4888
Completed request 4407
Completed request 725
Completed request 8111
Completed request 9041
Completed request 2396
Completed request 4624
Completed request 849
Completed request 4746
Completed request 5120
Completed request 5613
Completed request 2177
Completed request 8984
Completed request 2809
Completed request 7844
Completed request 7774
Completed request 5157
Completed request 9701
Completed request 3194
Completed request 3958
Completed request 7536
Completed request 28
Completed request 5616
Completed 1000 in 8.748724222183228 seconds
114.30237993608306 requests per second
```

Multiple Compose Files

<https://docs.docker.com/compose/extends/> ↗ (<https://docs.docker.com/compose/extends/>)

Load Balancing and Scaling with Docker Compose

<https://medium.com/@vinodkrane/microservices-scaling-and-load-balancing-using-docker-compose-78bf8dc04da9> ↗ (<https://medium.com/@vinodkrane/microservices-scaling-and-load-balancing-using-docker-compose-78bf8dc04da9>)

Recordings

Studio



Assignment

Docker Compose for MNIST Training, Evaluation, and Inference

In this assignment, you will create a Docker Compose configuration to perform training, evaluation, and inference on the MNIST dataset.

Requirements:

1. You'll need to use this model and training technique (MNIST Hogwild):
https://github.com/pytorch/examples/tree/main/mnist_hogwild ↗

(https://github.com/pytorch/examples/tree/main/mnist_hogwild).

2. Set Num Processes to 2 for MNIST HogWild
3. Create three services in the Docker Compose file: `train`, `evaluate`, and `infer`.
4. Use a shared volume called `mnist` for sharing data between the services.
5. The `train` service should:
 - o Look for a checkpoint file in the volume. If found, resume training from that checkpoint. Train for **ONLY 1 epoch** and save the final checkpoint. Once done, exit.
6. The `evaluate` service should:
 - o Look for the final checkpoint file in the volume. Evaluate the model using the checkpoint and save the evaluation metrics in a json file. Once done, exit.
 - o Share the model code by importing the model instead of copy-pasting it in `eval.py` ↗ (<http://eval.py>)
7. The `infer` service should:
 - o Run inference on any 5 random MNIST images and save the results (images with file name as predicted number) in the `results` folder in the volume. Then exit.
8. After running all the services, ensure that the model, and results are available in the `mnist` volume.

Detailed Instructions:

1. Build all the Docker images using `docker compose build`.
2. Run the Docker Compose services using `docker compose run train`, `docker compose run evaluate`, and `docker compose run infer`. Verify that all services have completed successfully.
3. Check if the checkpoint file (`mnist_cnn.pt`) is saved in the `mnist` volume. If found, display "Checkpoint file found." If not found, display "Checkpoint file not found!" and exit with an error.
4. Check if the evaluation results file (`eval_results.json`) is saved in the `mnist` volume.
 1. Example: `{"Test loss": 0.0890245330810547, "Accuracy": 97.12}`
5. Check the contents of the `results` folder in the `mnist` volume see if the inference results are saved.
6. You can always comment/uncomment parts of the `grading.sh` ↗ (<http://grading.sh>) to skip some tests, this would reduce time while debugging and remove the need to run all the tests everytime you run `grading.sh` ↗ (<http://grading.sh>). But make sure you don't commit changes in `grading.sh` ↗ (<http://grading.sh>) to repo

The provided grading script will run the Docker Compose configuration, check for the required files, display the results, and perform size and version checks.

You can run it yourself before pushing the code to your repo

```
bash tests/grading.sh
```

Also, You can use ChatGPT o1, Claude, Cursor for the Assignment 😊

Github Classroom:

<https://classroom.github.com/a/H1dh0F7f> ↗

(<https://classroom.github.com/a/H1dh0F7f>)

The screenshot shows a GitHub Classroom run summary for a job named "run-autograding-tests". The job succeeded 1 minute ago in 3m 32s. The summary includes logs for Docker Compose training and inference, and an Autograding Reporter section with a table of test results.

Logs (Docker Compose):

```
700 ✓ Train Epoch: 1 [59520/60000 (99%)] Loss: 0.028100
707 9 Train Epoch: 1 [59520/60000 (99%)] Loss: 0.020851
708 {'Test loss': 0.09178855972290038, 'Accuracy': 97.12}
709 Inference completed. Results saved in the 'results' folder.
710 ✅ All services have completed.
711 ❌ Checking for checkpoint file...
712 ✅ Checkpoint file found.
713 ❌ Checking for eval_results.json file...
714 ✅ eval_results.json file found.
715 📄 Printing the content of eval_results.json file...
716 {"Test loss": 0.09178855972290038, "Accuracy": 97.12} ❌ Checking for inference results...
717 ✅ 5 inference result images found.
718 ➔ All checks passed successfully!
```

Autograding Reporter:

```
1 ▶Run classroom-resources/autograding-grading-reporter@v1
2   with:
3     runners: test-docker-compose
4     token: ***
5     env:
6       TEST_DOCKER_COMPOSE_RESULTS:
7         eyJ2ZXJzaW9uIjoxCzdf0dXMIo1jVXNzIiwibWF4X3Njb3JlIjoiMDAsInRlc3RzIjpbeyJuYll1jo1VGVzdCBEB2NrZXIgQ29tcG9zZSIisInN0YXRicyI6InBhc3MiLCJzY29yZSI6NTAwLCJ0ZXN0X2NGU1Oj1YXNoIC4vdGVzdMwvZ3jhZGU1Yz5zciStimZpbGvUyWlljoiIiwibGluzV9ubylGMGwiZhVyyXRpb24i0jIwNzQzMn1dfQ==
8   📄 Processing: test-docker-compose
9   ✅ Test Docker Compose
10  Test code:
11    bash ./tests/grading.sh
12  Total points for test-docker-compose: 500.00/500
13
14  Test runner summary
15
16  ┌─────────┐ ┌─────────┐ ┌─────────┐
17  │ Test Runner Name | Test Score | Max Score |
18  └─────────┘ └─────────┘ └─────────┘
19  test-docker-compo... | 500 | 500
20  Total: | 500 | 500
21
22  ⚠ Grand total tests passed: 1/1
23
24 Workflow Run Response: https://api.github.com/repos/The-School-of-AI/emlo4-session-03-satyajitghana/check-suites/28340407286
```