GitHub Summary

What is GitHub?

GitHub is a cloud-based platform for **version control** and **collaboration**, built around **Git**, a system created by Linus Torvalds. It allows individuals and teams to manage changes to code, track versions, and collaborate on software projects efficiently.

How GitHub is Used

GitHub provides an online space to host repositories, track issues, review code, and collaborate through pull requests. It integrates directly with Git, so you can manage local code and sync it with a remote repository.

Typical Git workflow:

1. **Clone or create** a repository.
2. **Make changes** locally.
3. **Stage** the files you want to include.
4. **Commit** those changes with a message.
5. **Push** your commits to GitHub.

Example Git Workflow

```
# Step 1: Check repository status
git status

# Step 2: Stage specific files
git add simple_ml_script.py

# Step 3: Commit with a clear message
git commit -m "Updated model script and added output saving"

# Step 4: Push to the main branch
git push origin main
```

If you are setting up your repository for the first time:

```
# Link local repo to GitHub remote
git remote add origin git@github.com:username/repo-name.git

# Push initial commit
git push -u origin main
```

GitHub in Machine Learning (ML)

In ML workflows, GitHub is used to:

- **Version control** model training code, datasets, and experiment configurations.
- **Collaborate** on research and development with pull requests and reviews.
- **Integrate CI/CD pipelines** to automate training, testing, and deployment.
- **Track experiments** and reproducibility for model lifecycle management.

Many teams pair GitHub with tools like **DVC (Data Version Control)** and **GitHub Actions** for automating training or deployment workflows.

Docker Summary

What is Docker?

Docker is a **containerization platform** that allows you to package code, dependencies, and environment settings into lightweight, portable containers. These containers ensure consistent behavior across different systems.

Key Terms

Term	Description
Image	A snapshot of an environment (code, dependencies, OS) used to run containers.
Container	A running instance of an image; it's isolated and lightweight.
Dockerfile	A script defining how to build a Docker image (base image, dependencies, commands).
Volume	A way to persist data by linking local folders to the container's filesystem.
Registry	A repository where Docker images are stored (e.g., Docker Hub).

End-to-End ML Example

Project structure:

```
Simple_Docker_Practice/
│  Dockerfile
│  simple_ml_script.py
│  outputs/
```

simple_ml_script.py

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import joblib, pandas as pd, os

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
test_size=0.2, random_state=42)

model = LogisticRegression(max_iter=200, C=1.0)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"Model accuracy: {acc:.4f}")

output_dir = "/app/outputs"
os.makedirs(output_dir, exist_ok=True)

pd.DataFrame({"true": y_test, "predicted": y_pred}).to_csv(f"{output_dir}/
predictions.csv", index=False)
joblib.dump(model, f"{output_dir}/simple_logreg_model.joblib")
print("Outputs saved to /app/outputs")
```

Dockerfile

```
FROM python:3.10-slim
WORKDIR /app
COPY simple_ml_script.py .
RUN pip install scikit-learn pandas joblib
CMD ["python", "simple_ml_script.py"]
```

Commands to Run:

```
# Build image
docker build -t ml-docker-demo .

# Create output folder
mkdir -p outputs

# Run container with mounted output volume
docker run -v $(pwd)/outputs:/app/outputs ml-docker-demo
```

Expected Output:

```
Model accuracy: 1.0000
Predictions saved to '/app/outputs/predictions.csv'
Model saved to '/app/outputs/simple_logreg_model.joblib'
```

Key Docker Commands

Command	Description
`docker build -t name .`	Builds a Docker image from a Dockerfile.
`docker run name`	Runs a container from an image.
`docker ps`	Lists running containers.
`docker images`	Lists locally stored Docker images.
`docker exec -it <container> /bin/bash`	Opens a shell in a running container.
`docker rm` / `docker rmi`	Removes containers or images.
`docker push <repo>`	Pushes your image to Docker Hub or another registry.

Docker in ML

Docker ensures:

- **Reproducibility**: Identical environments for training, testing, and deployment.
- **Portability**: Run your model anywhere — cloud, local, or CI/CD.
- **Scalability**: Integrate easily with Kubernetes or cloud orchestration systems.

In ML pipelines, Docker is key to **MLOps**, enabling automated deployment and consistent model execution across development and production.

In summary:

- **GitHub** manages *code versions and collaboration.*
- **Docker** manages *environments and deployment consistency.*

Together, they form the backbone of a modern, reproducible ML workflow.