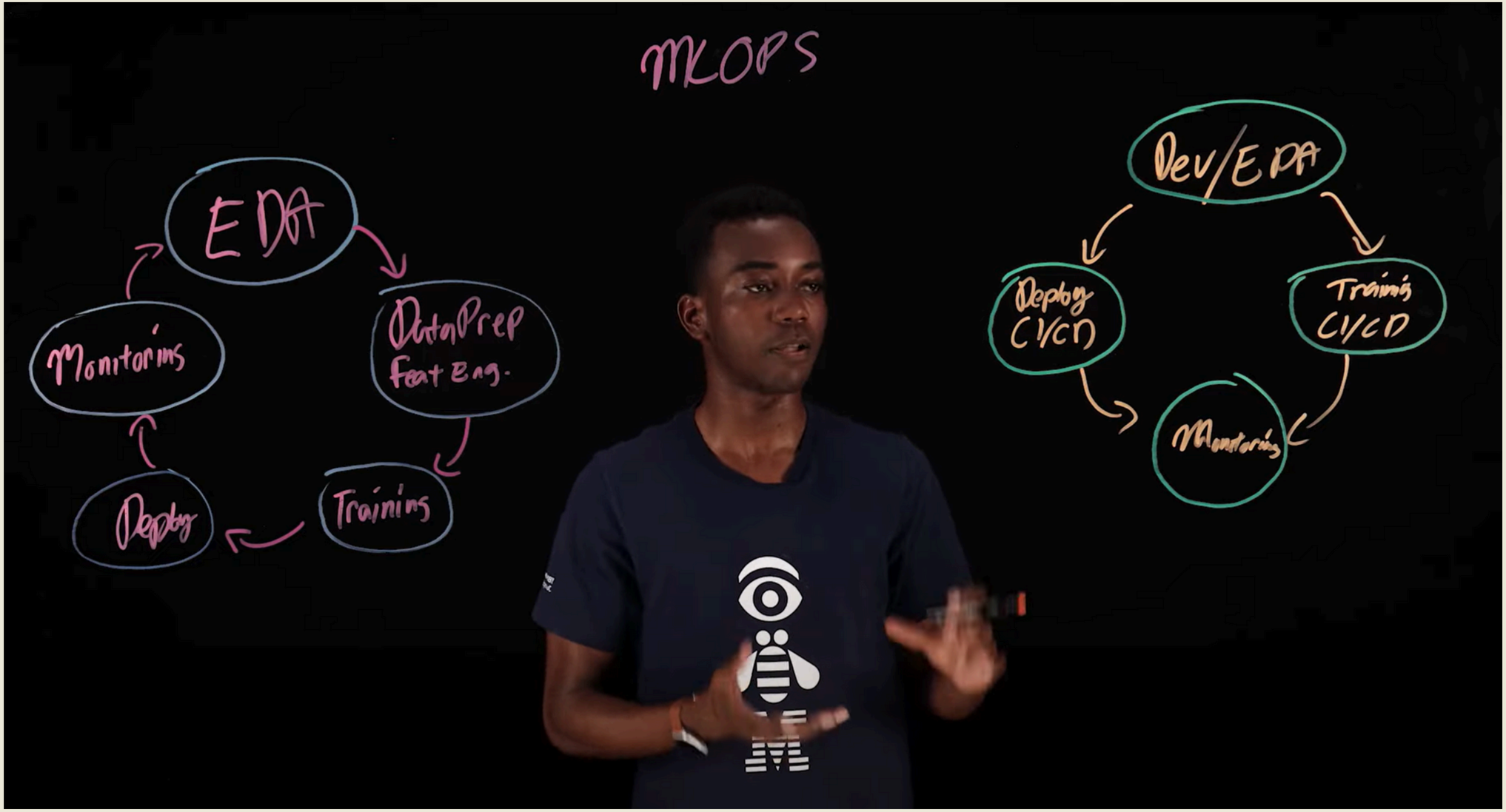
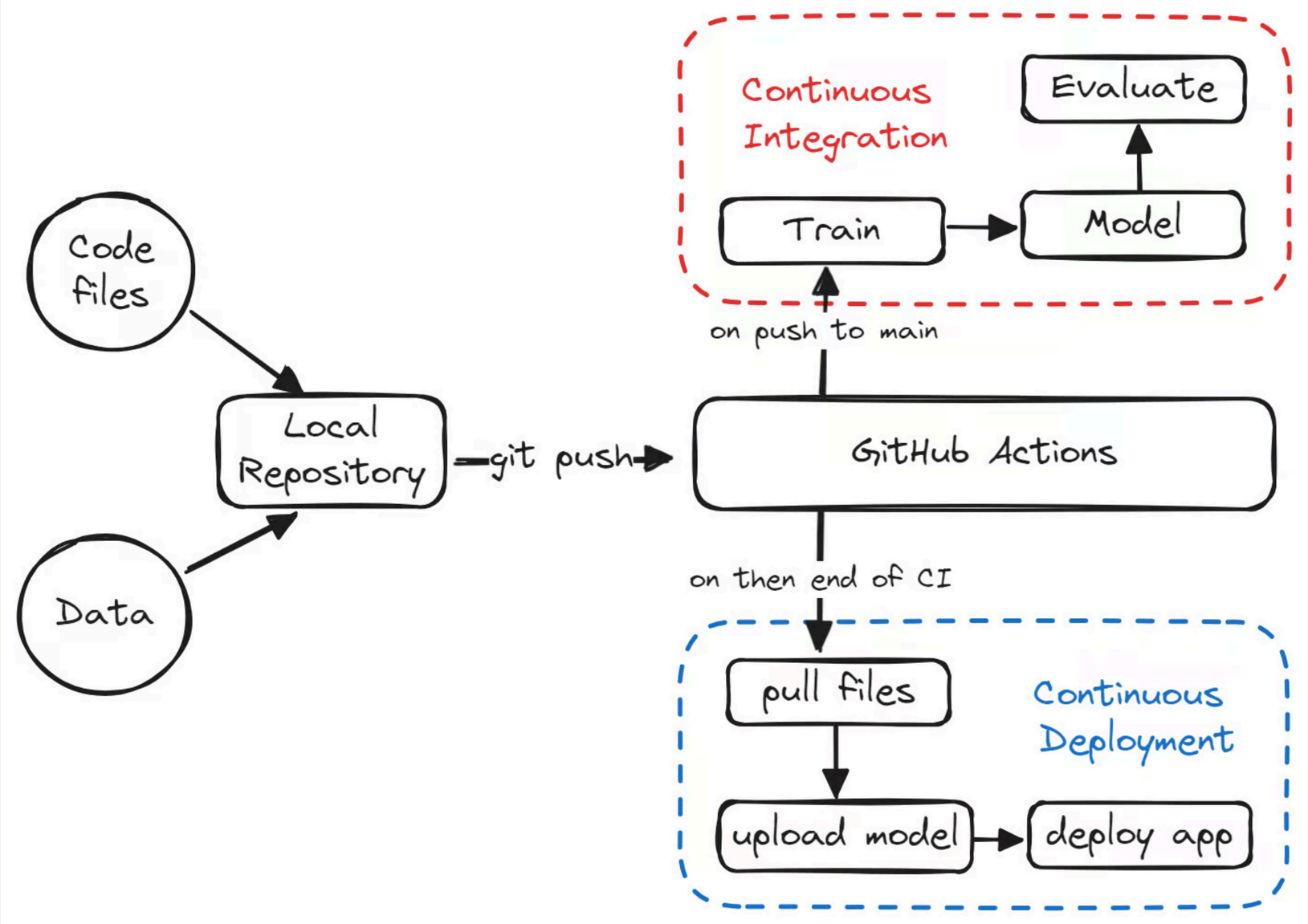
CI/CD

For Machine Learning

Sumber : A Beginner's Guide to CI/CD for Machine Learning



Sumber : What is MLOps?



Sumber : A Beginner's Guide to CI/CD for Machine Learning

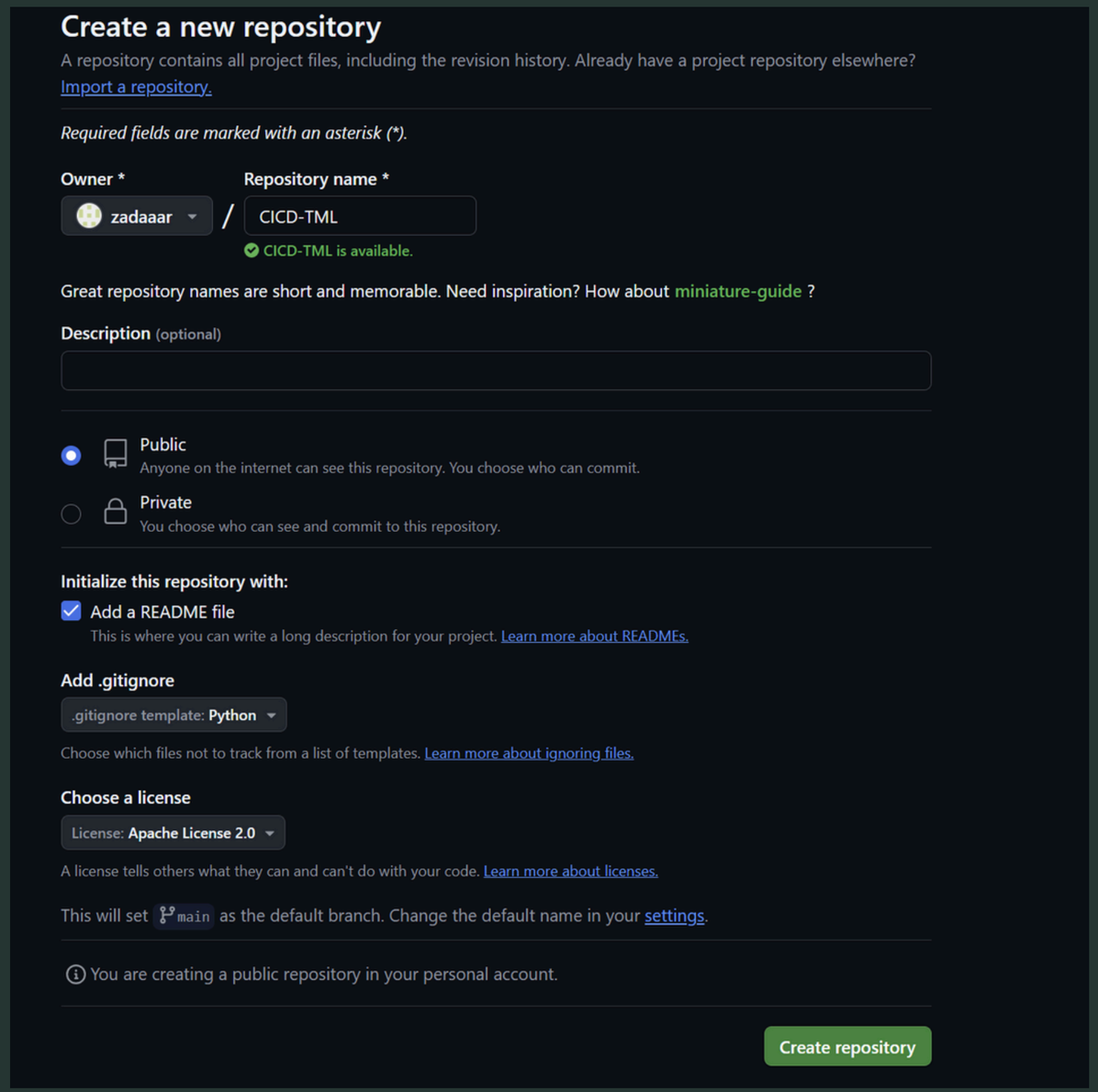
APA TUJUAN AKHIRNYA?

Tujuan akhir dari Praktikum ini adalah membuat sebuah aplikasi dengan interface pada Hugging Face, di mana sudah terotomatisasi sejak melakukan **push** ke **Github** hingga aplikasi ter-**deploy.** Proses otomatisasi akan dibantu dengan **Github Actions**

PERSIAPAN REPOSITORY

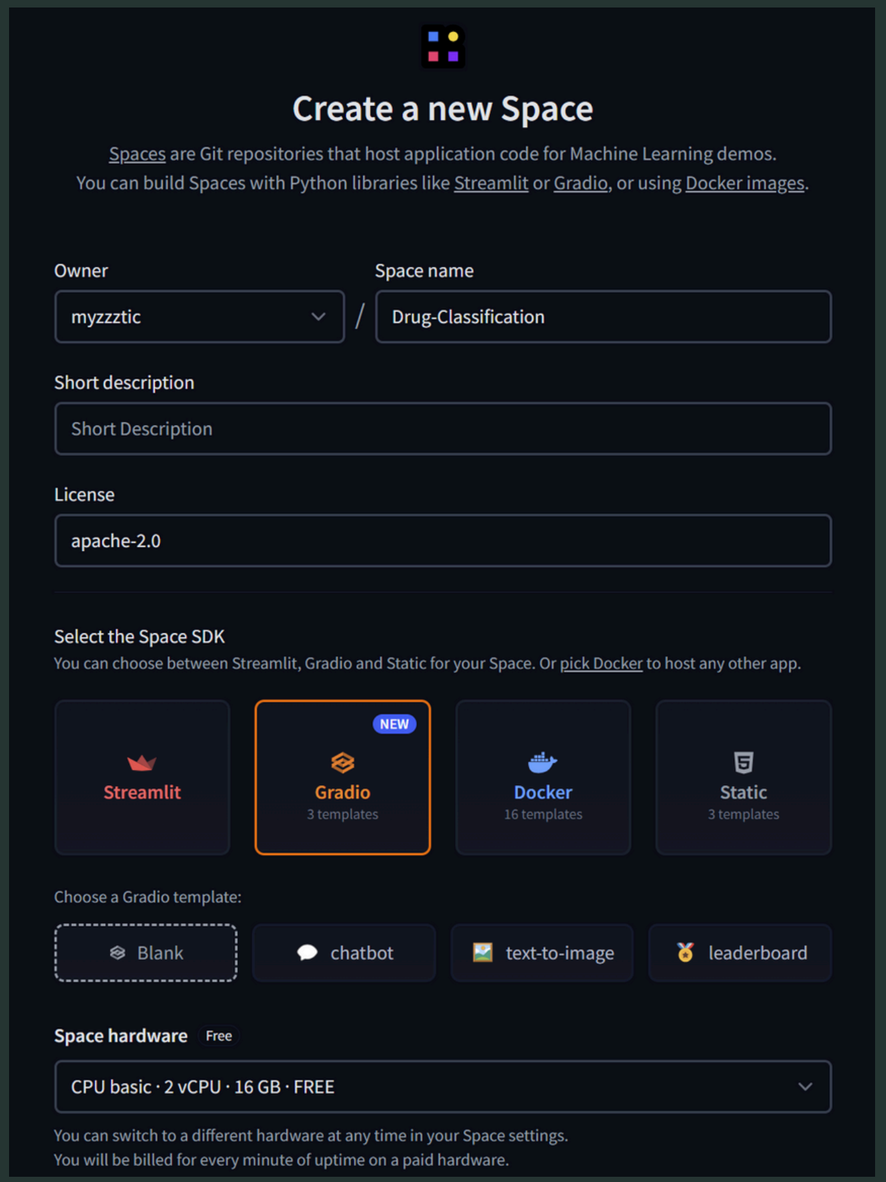
Buat new repository di github

1. Isi nama repository bebas
2. Beri deskripsi (opsional)
3. Buat dalam bentuk publik
4. Tambahkan README
5. Tambahkan .gitignore dengan template Python
6. Pilih license Apache License 2.0 (Paling atas)
7. Klik Create Repository



Buat new Space di Hugging Face:

1. Isi nama space “Drug-Classification”
2. Isi short description (optional)
3. Pilih License apache-2.0
4. Pilih Space SDK Gradio
5. Pilih space hardware yang FREE
6. Klik Create Space
7. Klik “Files” pada taskbar
8. Edit “README.md”



README.MD

---
title: Drug Classification
emoji: 💊 (Win + `.` atau FN + `E`)
colorFrom: yellow
colorTo: red
sdk: gradio
sdk_version: 5.24.0
app_file: drug_app.py
pinned: false
license: apache-2.0

```
title: Drug Classification
emoji: 💊
colorFrom: yellow
colorTo: red
sdk: gradio
sdk_version: 5.24.0
app_file: drug_app.py
pinned: false
license: apache-2.0
```

LAKUKAN CLONE REPOSITORY

1. Buka folder/directory yang akan digunakan untuk menyimpan repo.
2. Klik kanan, lalu klik “Open in Terminal”
3. Ketik “git clone <link repository github>”
4. Setelah proses selesai, ketik “code repo”

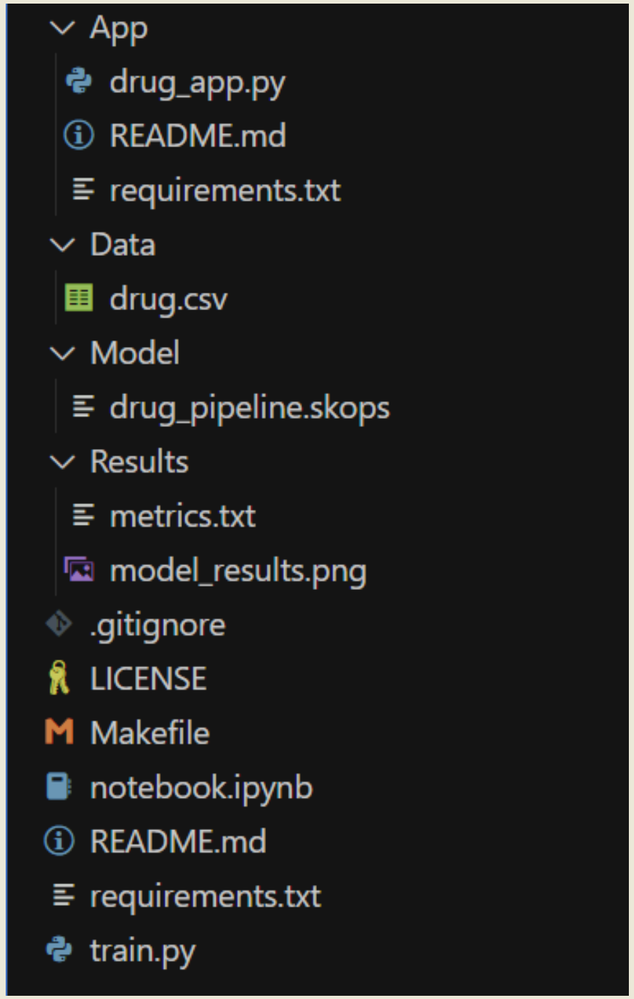
Tambahkan beberapa folder dan file, yaitu:

Folder:

1. App
2. Data
3. Model
4. Results

File:

1. App/drug_app.py
2. App/README.md
3. App/requirements.txt
4. Download [Dataset]() lalu masukkan ke dalam folder Data dengan nama drug.csv
5. Makefile
6. notebook.ipynb
7. requirements.txt
8. train.py



Masukkan pada README.md dan requirements.txt pada folder App

README.md

```
---
title: Drug Classification
emoji: 💊 (Win + `.` atau FN + `E`)
colorFrom: yellow
colorTo: red
sdk: gradio
sdk_version: 5.24.0
app_file: drug_app.py
pinned: false
license: apache-2.0
---
```

requirements.txt

```
# Core ML stack
scikit-learn==1.3.0
skops==0.10.0
numpy==1.24.0
pandas==1.5.0
gradio

# App dependencies
flask==2.3.0
black==23.7.0
```

LAKUKAN TRAINING DAN EVALUATING

Buka file notebook.ipynb

TRAINING & EVALUATION

Tahap ini merupakan tahap di mana kita akan **melakukan training dan evaluation** dari **model** yang nantinya akan kita gunakan pada CI/CD ini.

Load Dataset

```
import pandas as pd

drug_df = pd.read_csv("Data/drug.csv")
drug_df = drug_df.sample(frac=1)
drug_df.head(3)
```

Train Test Split

```
from sklearn.model_selection import train_test_split

X = drug_df.drop("Drug", axis=1).values
y = drug_df.Drug.values

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=125
)
```

Machine Learning Pipeline

```
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OrdinalEncoder, StandardScaler

cat_col = [1,2,3]
num_col = [0,4]

transform = ColumnTransformer(
    [
        ("encoder", OrdinalEncoder(), cat_col),
        ("num_imputer", SimpleImputer(strategy="median"), num_col),
        ("num_scaler", StandardScaler(), num_col),
    ]
)
pipe = Pipeline(
    steps=[
        ("preprocessing", transform),
        ("model", RandomForestClassifier(n_estimators=100,
random_state=125)),
    ]
)
pipe.fit(X_train, y_train)
```

Model Evaluation

```
from sklearn.metrics import accuracy_score, f1_score

predictions = pipe.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
f1 = f1_score(y_test, predictions, average="macro")

print("Accuracy:", str(round(accuracy, 2) * 100) + "%", "F1:", round(f1,
2))
```

Save Metrics

```
with open("Results/metrics.txt", "w") as outfile:
    outfile.write(f"\nAccuracy = {accuracy.round(2)}, F1 Score = {f1.round(2)}.")
```

Create Confusion Matrix

```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix

cm = confusion_matrix(y_test, predictions, labels=pipe.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipe.classes_)
disp.plot()
plt.savefig("Results/model_results.png", dpi=120)
```

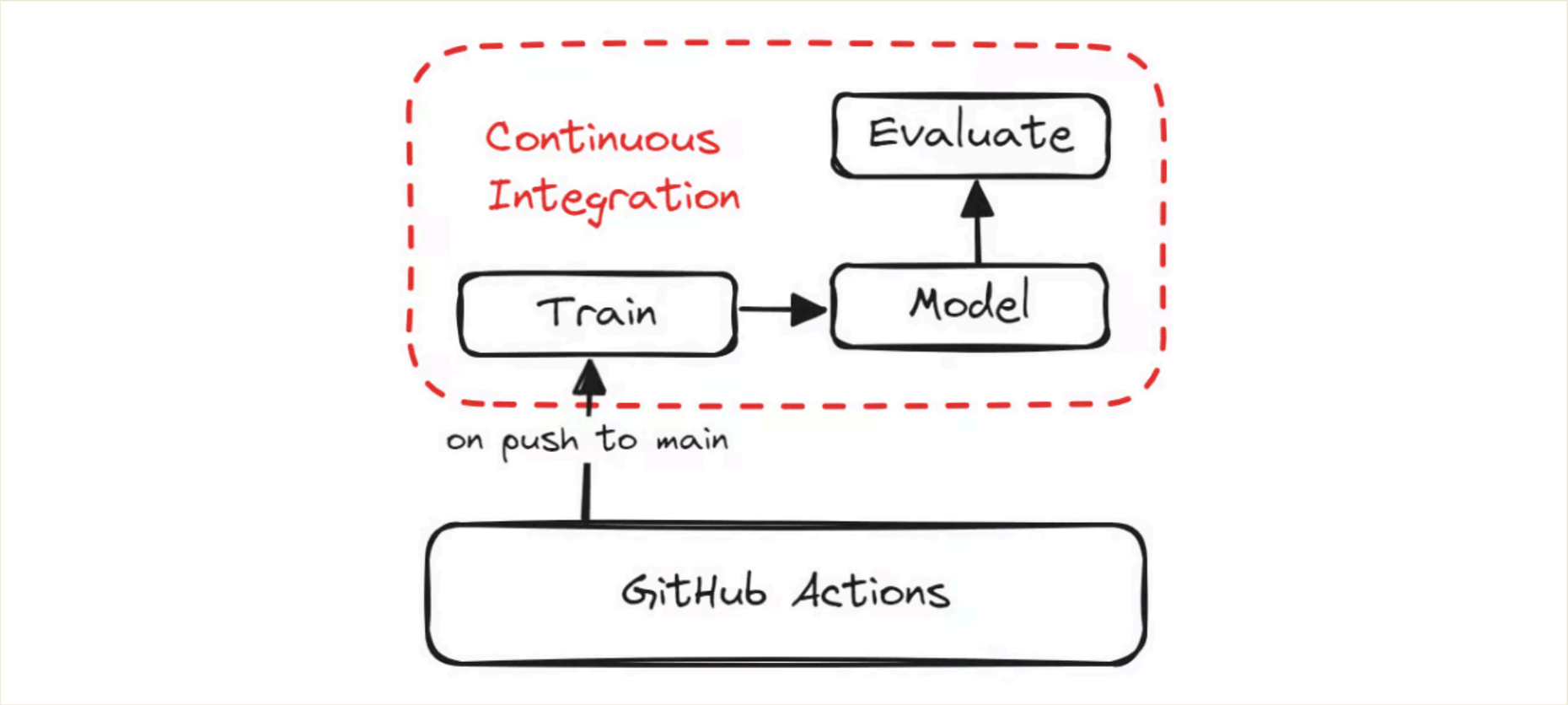
Save Model

```
import skops.io as sio

sio.dump(pipe, "Model/drug_pipeline.skops")
```

Masukkan pada file train.py

Kode ini akan digunakan pada saat CI

BUAT PIPELINE CONTINUOUS INTEGRATION



Sumber : A Beginner's Guide to CI/CD for Machine Learning

CONTINUOUS INTEGRATION

Pada tahap ini, kita ingin saat kita melakukan **push** ke dalam Github akan **otomatis** melakukan Training, Menyimpan Model, dan Mengevaluasi Model tersebut. Dengan itu model akan selalu **terintegrasi.** Kita akan menggunakan Continuous Machine Learning (**CML**) untuk membantu proses integrasi.

Buka Makefile

```
install:
  pip install --upgrade pip &&\
  pip install -r requirements.txt

format:
  black *.py

train:
  python train.py

eval:
  echo "## Model Metrics" > report.md
  cat ./Results/metrics.txt >> report.md

  echo '\n## Confusion Matrix Plot' >> report.md
  echo '![Confusion Matrix](./Results/model_results.png)' >> report.md

  cml comment create report.md
```

Tambah requirements.txt

```
pandas
scikit-learn
numpy
matplotlib
skops
black
```

Lakukan commit dan push pada cmd/terminal (wajib diingat)

```
git add .
git commit -am “new changes”
git push origin main
```

Jika melihat lampu ini maka lakukan commit setelah proses selesai

Buka link github masing-masing

Klik pada bagian Actions lalu klik “set up a workflow yourself”
Rename menjadi “ci.yml”

```yaml
name: Continuous Integration
on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]
  workflow_dispatch:

permissions: write-all
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: iterative/setup-cml@v2
      - name: Install Packages
        run: make install
      - name: Format
        run: make format
      - name: Train
        run: make train
      - name: Evaluation
        env:
          REPO_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: make eval
```

CI.YML

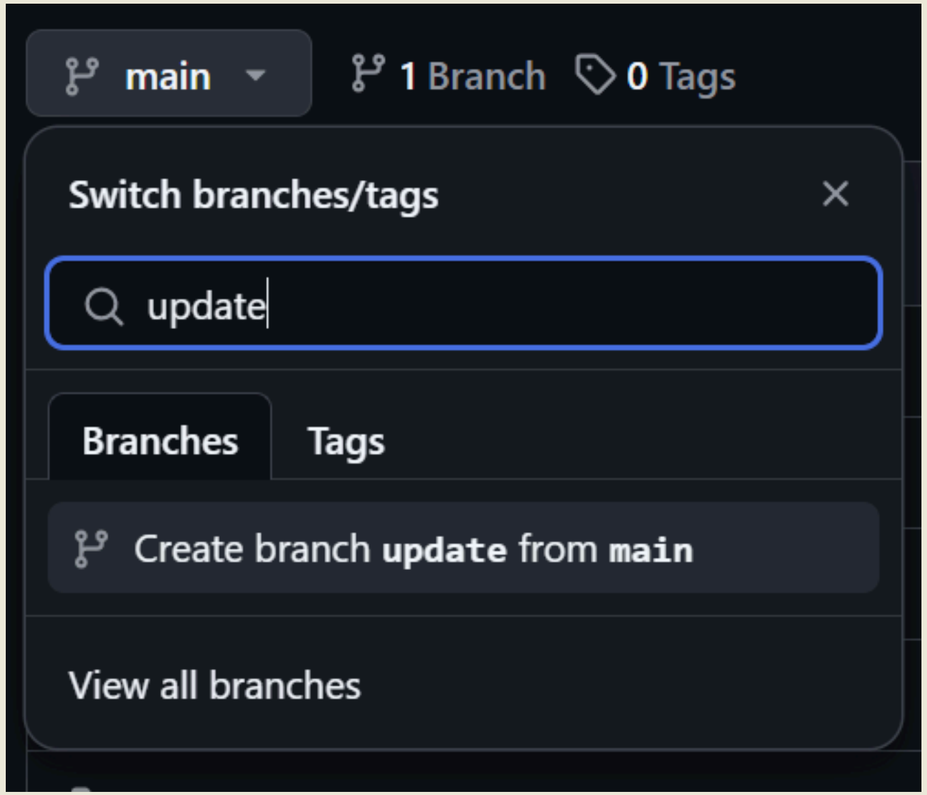
Pada slide sebelumnya terdapat tulisan “on:” yang merupakan trigger dari actions yang kita buat, yaitu saat kita melakukan **Push**.

Selanjutnya adalah menuliskan apa saja yang akan dilakukan pada “Actions” ini, mulai dari Install Packages, Format, Train, dan Evaluation.

NEW BRANCH “UPDATE”

Branch Update berfungsi agar model dan hasil yang telah kita buat akan tersimpan secara jelas versi versinya dan tidak mengubah main.

Buat Branch baru bernama “update”

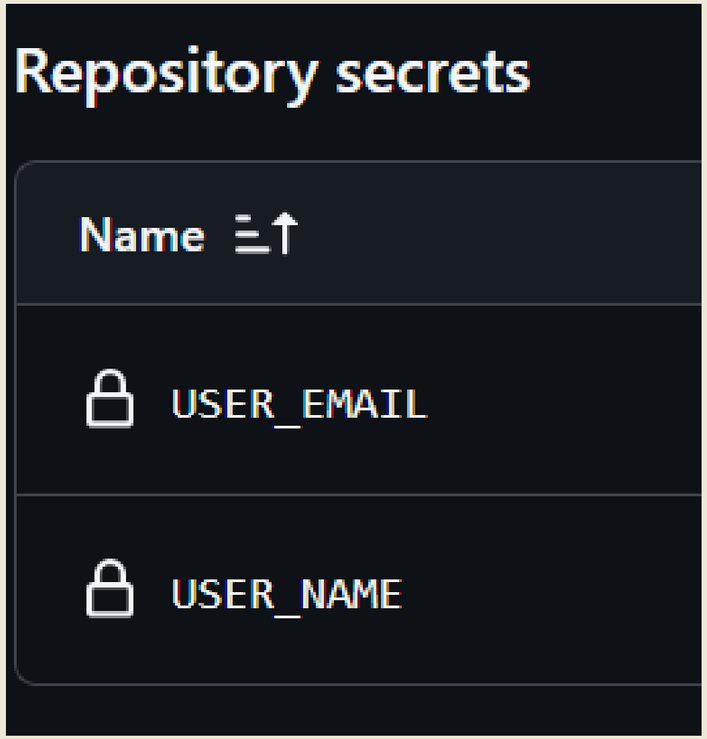


REPOSITORY SECRET

Agar dapat melakukan **command Git** untuk **mengupdate Branch**, kita membutuhkan email dan username dari GitHub, namun hal tersebut kadang bersifat **sensitif/rahasia**. Oleh karena itu kita gunakan fitur **“Repository Secret”**

Buat repository secrets

1. Klik Settings dan klik “Secrets and variables”
2. Pilih “Actions” dan klik “New repository secret”
3. Tambahkan seperti gambar di samping sesuai dengan email github dan usernam github



Update Makefile

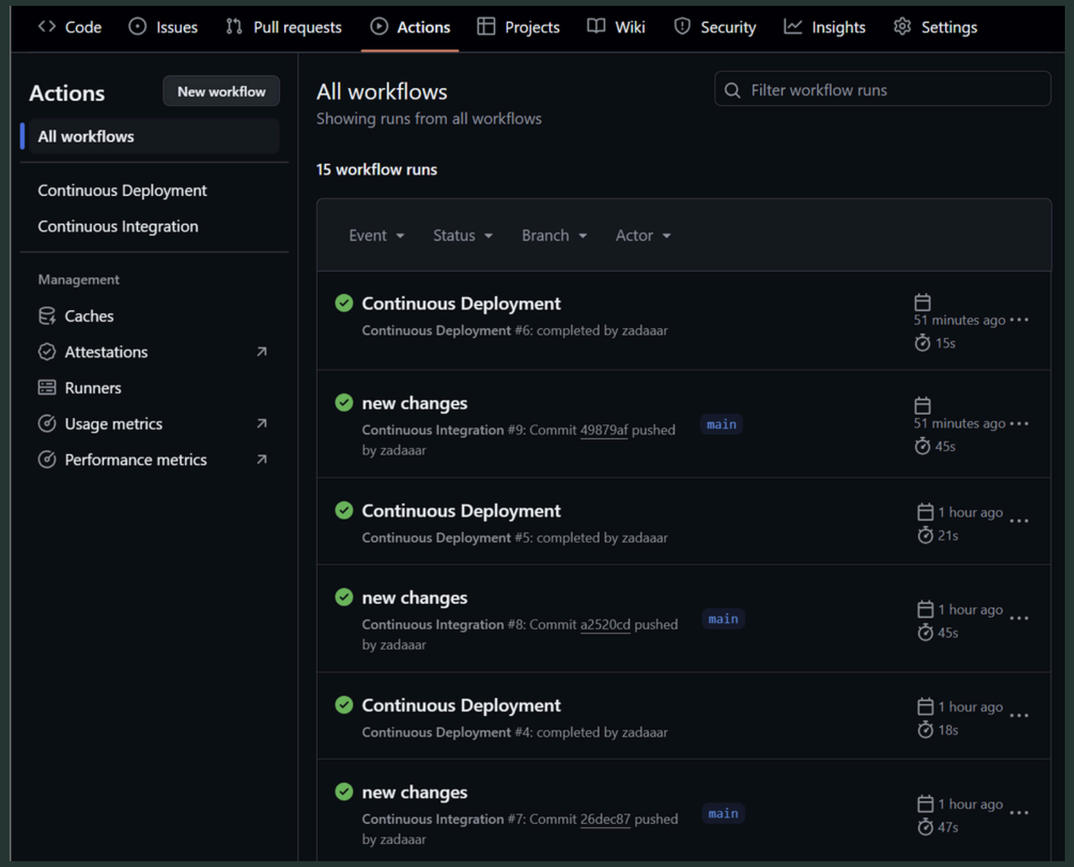
```
update-branch:
 git config --global user.name $(USER_NAME)
 git config --global user.email $(USER_EMAIL)
 git commit -am "Update with new results"
 git push --force origin HEAD:update
```

Update ci.yml

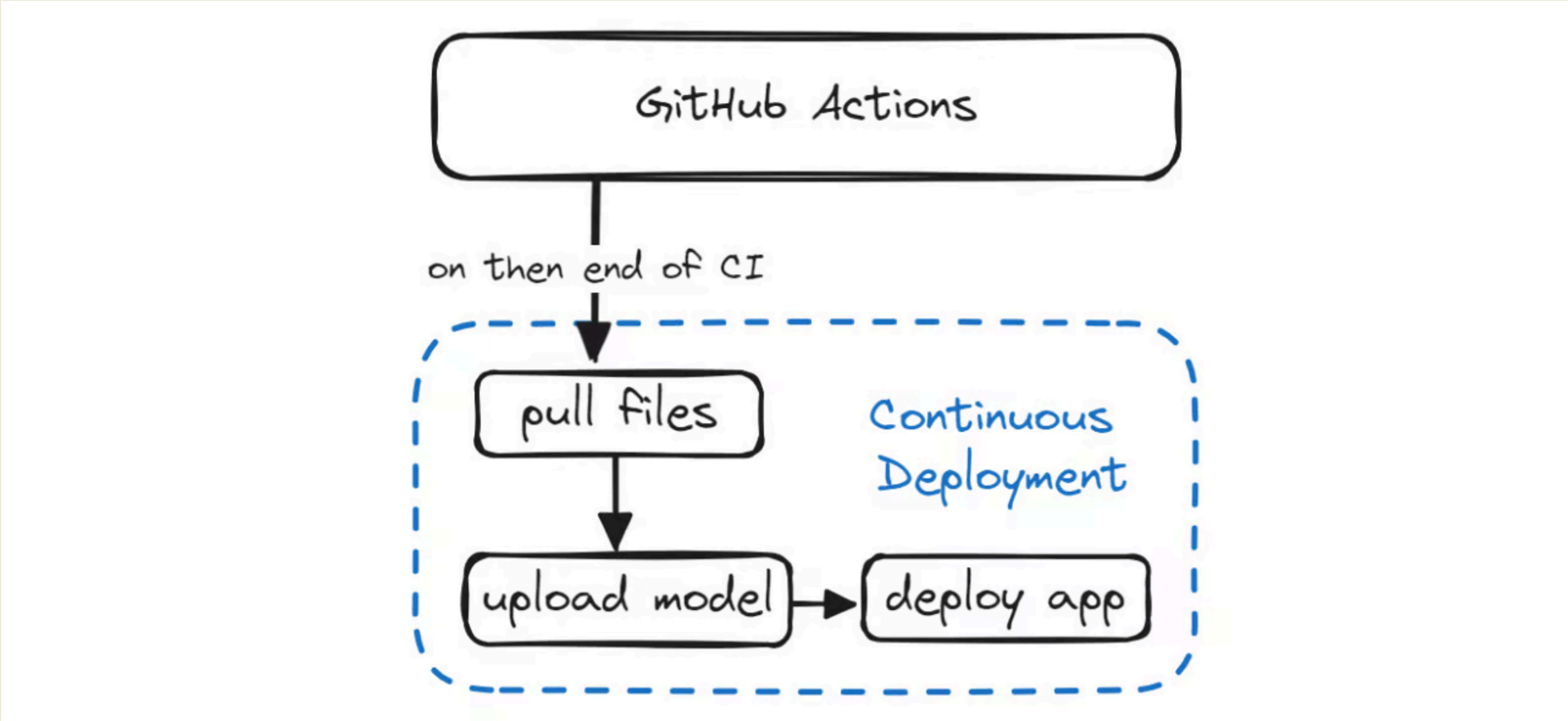
```
- name: Update Branch
    env:
      NAME: ${{ secrets.USER_NAME }}
      EMAIL: ${{ secrets.USER_EMAIL }}
    run: make update-branch USER_NAME=$NAME USER_EMAIL=$EMAIL
```

SETELAH PUSH

Kita bisa memantau workflow dengan cara membuka Repository pada Github dan klik “Actions” pada taskbar



BUAT PIPELINE CONTINUOUS DEVELOPMENT



Sumber : A Beginner's Guide to CI/CD for Machine Learning

TAMPILAN APLIKASI

Selanjutnya kita akan membuat tampilan aplikasi yang akan kita deploy dengan model yang sudah kita buat. Aplikasi tersebut akan memuat:

1. **Model** yang telah kita training
2. Terdapat **input** berupa form dan slider untuk melakukan testing sebagai User
3. Tampilan **output** dari hasil analisis yang dilakukan oleh model yang telah dibuat.
4. Tentunya dengan tampilan yang tidak membosankan

Buat drug_app.py

```
import gradio as gr
import skops.io as sio

pipe = sio.load("./Model/drug_pipeline.skops",
trusted=sio.get_untrusted_types(file="./Model/drug_pipeline.skops"))


def predict_drug(age, sex, blood_pressure, cholesterol, na_to_k_ratio):
  """Predict drugs based on patient features.

  Args:
    age (int): Age of patient
    sex (str): Sex of patient
    blood_pressure (str): Blood pressure level
    cholesterol (str): Cholesterol level
    na_to_k_ratio (float): Ratio of sodium to potassium in blood

  Returns:
    str: Predicted drug label
  """
```

Buat drug_app.py

```
features = [age, sex, blood_pressure, cholesterol, na_to_k_ratio]
 predicted_drug = pipe.predict([features])[0]

 label = f"Predicted Drug: {predicted_drug}"
 return label


inputs = [
 gr.Slider(15, 74, step=1, label="Age"),
 gr.Radio(["M", "F"], label="Sex"),
 gr.Radio(["HIGH", "LOW", "NORMAL"], label="Blood Pressure"),
 gr.Radio(["HIGH", "NORMAL"], label="Cholesterol"),
 gr.Slider(6.2, 38.2, step=0.1, label="Na_to_K"),
]
outputs = [gr.Label(num_top_classes=5)]

examples = [
 [30, "M", "HIGH", "NORMAL", 15.4],
 [35, "F", "LOW", "NORMAL", 8],
 [50, "M", "HIGH", "HIGH", 34],
]
```

Buat drug_app.py

```
title = "Drug Classification"
description = "Enter the details to correctly identify Drug type?"
article = "This app is a part of the Beginner's Guide to CI/CD for Machine Learning. It teaches how to
automate training, evaluation, and deployment of models to Hugging Face using GitHub Actions."


gr.Interface(
 fn=predict_drug,
 inputs=inputs,
 outputs=outputs,
 examples=examples,
 title=title,
 description=description,
 article=article,
 theme=gr.themes.Soft(),
).launch()
```

COBA SECARA LOKAL

Setelah ini kita akan mencoba untuk menjalankan aplikasi **secara lokal terlebih dahulu** sebelum kita lakukan deployment, fungsinya agar **lebih mudah** saat melakukan **troubleshooting**.

COBA SECARA LOCAL

Buat virtual environment

```
python -m venv .venv
venv\Scripts\activate
pip install -r .\App\requirements.txt
python .\App\drug_app.py
```

Contoh Tampilan Lokal:

Drug Classification

Enter the details to correctly identify Drug type?

Age 40

15 — 74

Sex

m | F

Blood Pressure

HIGH | LOW | NORMAL

Cholesterol

HIGH | NORMAL

Na_to_K 9.6

6.2 — 38.2

output

Predicted Drug: drugA

Flag

Clear **Submit**

Examples

Age	Sex	Blood Pressure	Cholesterol	Na_to_K
30	m	HIGH	NORMAL	15.4
35	F	LOW	NORMAL	8
50	m	HIGH	HIGH	34

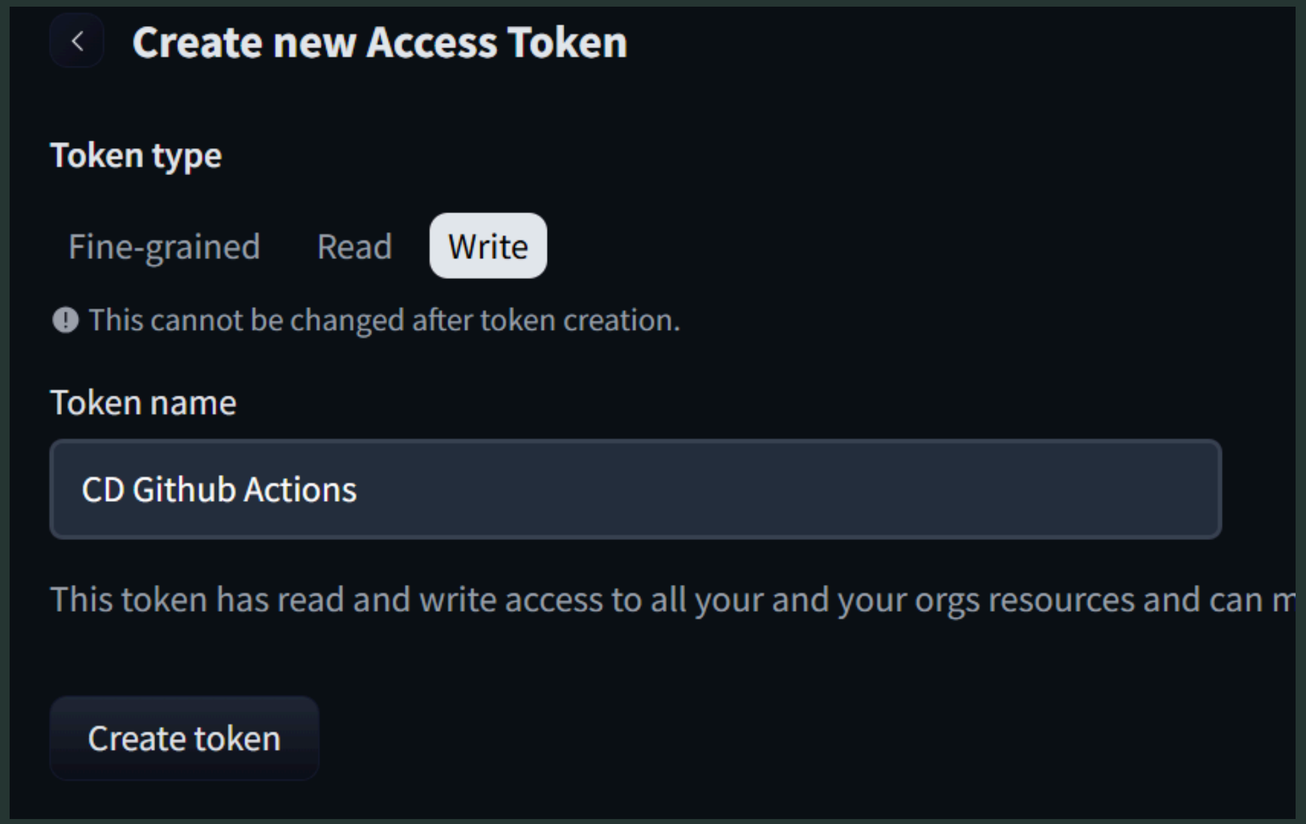
This app is a part of the Beginner's Guide to CI/CD for Machine Learning. It teaches how to automate training, evaluation, and deployment of models to Hugging Face using GitHub Actions.

HUGGING FACE TOKEN

Setelah berhasil mencoba secara lokal, kita membutuhkan token untuk menyambungkan github ke hugging face dan bisa menambah isi dari Space yang ada di Hugging Face.

Buat token Hugging Face:

1. Klik foto profil di taskbar
2. Klik “Acces Tokens”
3. Klik “Create New Token”
4. Klik Token type “Write”
5. Beri nama CD Github Actions
6. Klik “Create Token”
7. Copy token tersebut dan masukkan ke repository secret baru bernama “HF”



Update Makefile

```
hf-login:
  git pull origin update
  git switch update
  pip install -U "huggingface_hub[cli]"
  huggingface-cli login --token $(HF) --add-to-git-credential

push-hub:
  huggingface-cli upload <UserNameHF>/Drug-Classification ./App --repo-type=space --commit-message="Sync App files"
  huggingface-cli upload <UserNameHF>/Drug-Classification ./Model /Model --repo-type=space --commit-message="Sync Model"
  huggingface-cli upload <UserNameHF>/Drug-Classification ./Results /Metrics --repo-type=space --commit-message="Sync
Model"

deploy: hf-login push-hub
```

Ganti yang **merah** dengan Username Hugging Face kalian.

Buat cd.yml

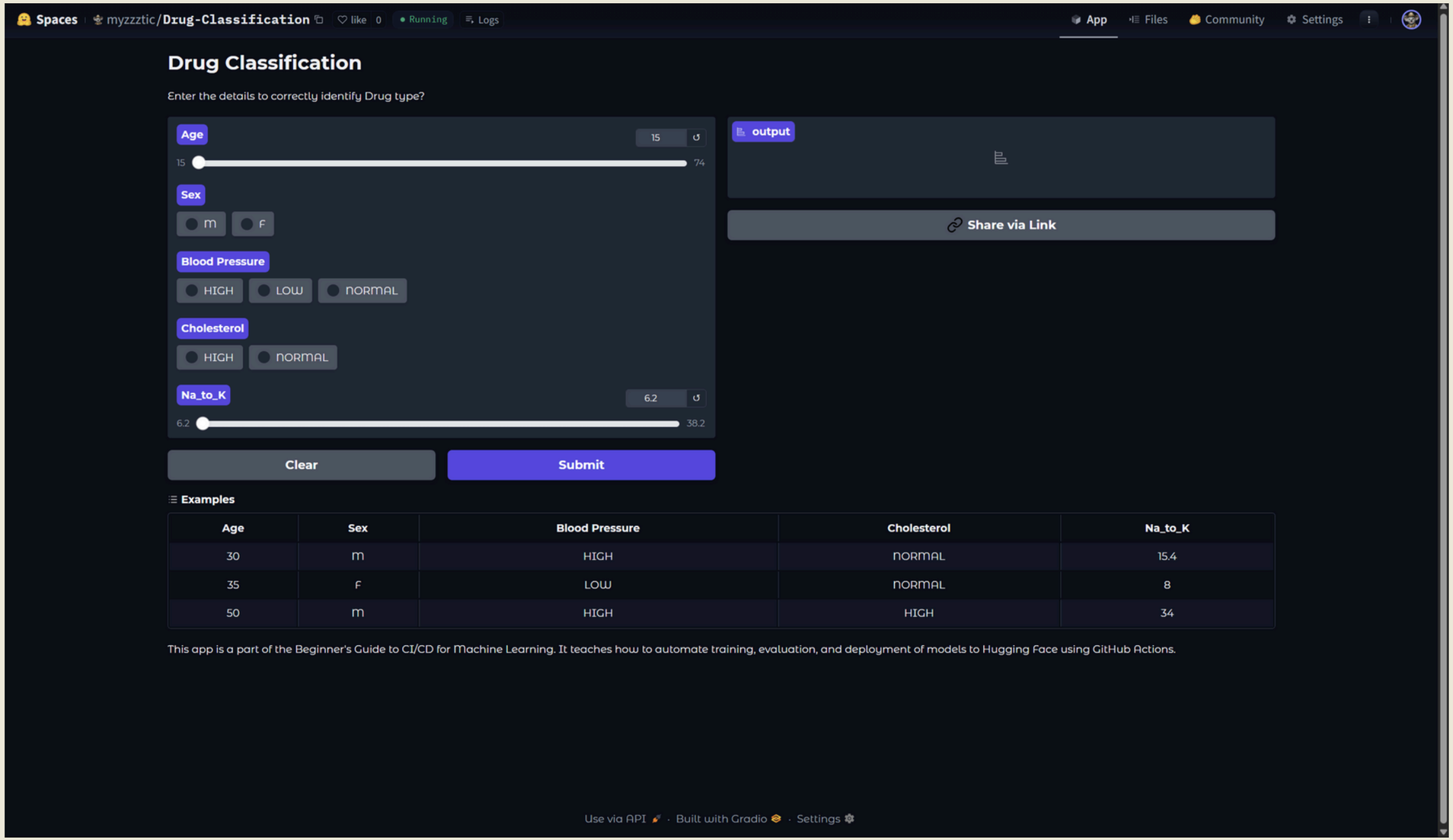
```yaml
name: Continuous Deployment
on:
 workflow_run:
  workflows: ["Continuous Integration"]
  types:
   - completed

 workflow_dispatch:

jobs:
 build:
  runs-on: ubuntu-latest
  steps:
   - uses: actions/checkout@v3

   - name: Deployment To Hugging Face
    env:
     HF: ${{ secrets.HF }}
    run: make deploy HF=$HF
```

Update requirements.txt pada Folder App

```
# Core ML stack
scikit-learn==1.6.1
skops>=0.11.0
numpy==1.24.0
pandas==1.5.0
gradio
# App dependencies
flask==2.3.0
black==23.7.0
```

Warna merah menunjukkan baris yang terdapat perubahan.

Tampilan Hugging Face



TERIMA KASIH