Here's a **comprehensive 4-hour session workflow and step-by-step practical guide** tailored for an Ubuntu setup on VirtualBox. The flow balances **hands-on activities**, **concept explanations**, and **integration of tools** across embedded systems, cloud, version control, and CI/CD.

🧭 Session Structure Overview (4 Hours Total)

Module	Topic	Duration
1	Embedded Systems Simulation	45 min
2	AWS EC2 & S3 Cloud Concepts	45 min
3	Git & GitHub Version Control	30 min
4	Docker & CI/CD Workflow	45 min
5	Integration Project: Edge-to-Cloud Dashboard	45 min
6	Wrap-Up, Deliverables & Resources	30 min

✅ Pre-Session Setup on Ubuntu VirtualBox

1. **Ubuntu Desktop 22.04 LTS** – Ensure VirtualBox Guest Additions are installed.
2. Update and install tools:

```
sudo apt update && sudo apt upgrade -y
sudo apt install python3 python3-pip git docker.io docker-compose -y
```

3. Add current user to Docker group:

```
sudo usermod -aG docker $USER
newgrp docker
```

4. Install Node.js (if needed for frontend dashboard):

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs
```

1️⃣ Embedded Systems Concepts & Simulation (45 min)

✅ Key Concepts:

- What is an Embedded System?
- Sensors, actuators, microcontrollers (simulate on Ubuntu).

✅ Practical Steps:

🟢 Simulate Sensor Data in Python:

```
# sensor_simulator.py
import time, random, json

def generate_data():
    return {
        "temperature": round(random.uniform(20.0, 35.0), 2),
        "humidity": round(random.uniform(40.0, 70.0), 2),
        "timestamp": time.strftime('%Y-%m-%d %H:%M:%S')
    }

with open("sensor_log.json", "a") as f:
    while True:
        data = generate_data()
        print(data)
        f.write(json.dumps(data) + "\n")
        time.sleep(2)
```

Run:

```
python3 sensor_simulator.py
```

2️⃣ AWS Cloud: EC2 & S3 (45 min)

✅ Setup:

- Guide students to create a free AWS account
- Use AWS Console or AWS CLI for actions.

✅ EC2 Practical:

1. Launch Ubuntu EC2 (t2.micro) in **us-east-1**.
2. Open port 22, 80 in Security Group.
3. SSH from Ubuntu VM:

```
ssh -i your-key.pem ubuntu@your-ec2-ip
```

4. Install Apache & Host Page:

```
sudo apt update
sudo apt install apache2 -y
echo "Hello from EC2 Web Server" | sudo tee /var/www/html/index.html
```

✅ S3 Practical:

1. Create S3 bucket via Console.
2. Upload a sample file:

```
echo "Test from Ubuntu VM" > testfile.txt
aws s3 cp testfile.txt s3://your-bucket-name/
```

3️⃣ Git & GitHub Version Control (30 min)

✅ Git Setup:

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

✅ Practical:

1. Initialize repo:

```
mkdir embedded_project && cd embedded_project
git init
```

2. Create and push to GitHub:

```
echo "# Embedded Cloud Project" > README.md
git add . && git commit -m "Initial commit"
# Link to GitHub
git remote add origin https://github.com/yourusername/yourrepo.git
git push -u origin master
```

3. **Branching & Conflict Demo:**

```
git checkout -b feature1
echo "Line from feature1" >> test.txt
git add . && git commit -m "Feature1"
git checkout master
git checkout -b feature2
echo "Line from feature2" >> test.txt
git add . && git commit -m "Feature2"
git checkout master
git merge feature1
git merge feature2  # Simulate and resolve conflict
```

4️⃣ Docker & CI/CD Workflow (45 min)

✅ Dockerize Flask App:

1. Create simple app:

```
# app.py
```

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def home():
    return "Embedded Cloud App"
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

2. Create Dockerfile:

```
FROM python:3.10
WORKDIR /app
COPY . .
RUN pip install flask
CMD ["python", "app.py"]
```

3. Build and run:

```
docker build -t embedded-app .
docker run -p 5000:5000 embedded-app
```

✅ Push to Docker Hub:

```
docker login
docker tag embedded-app yourdockerhubusername/embedded-app
docker push yourdockerhubusername/embedded-app
```

✅ GitHub Actions:

Create `.github/workflows/docker.yml` in repo:

```
name: Docker CI

on:
  push:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest
```

```
    steps:
      - uses: actions/checkout@v2
      - name: Build Docker Image
        run: docker build -t embedded-app .
```

5️⃣ Final Project: Edge-to-Cloud Dashboard (45 min)

✅ Objective:

- Sensor → Flask API → EC2 → Dashboard on web.

✅ Flask API on EC2:

1. Use same Flask app from above.

2. Modify to accept POST:

```
@app.route("/data", methods=["POST"])
def receive_data():
    content = request.json
    with open("data.json", "a") as f:
        f.write(json.dumps(content) + "\n")
    return "Data received", 200
```

3. Send sensor data from Ubuntu VM:

```
import requests
while True:
    data = generate_data()
    requests.post("http://<ec2-ip>:5000/data", json=data)
    time.sleep(5)
```

✅ Frontend Display (Optional with Chart.js):

Use Node.js, Flask templates or simple HTML:

```
<!-- display.html -->
```

```
<html><body><h2>Sensor Dashboard</h2><div id="output"></div></body></html>
```

6️⃣ Wrap-Up & Submission (30 min)

✅ Deliverables:

- GitHub repo with code and Dockerfile
- Screenshot of running EC2 Flask app
- Docker Hub link
- GitHub Actions CI badge (optional)

✅ Resources:

- Docker Playground
- AWS Educate
- Flask Docs
- GitHub Docs
- CI/CD with GitHub Actions

Would you like a printable PDF version or a GitHub README-based handout as well?