**🛢️ Oil Volume Prediction Project - Full Documentation**

This document provides a complete, beginner-friendly walkthrough of the **Oil Volume Prediction ML App** — from EC2 setup to GitHub version control to deployment on Render using Streamlit. Every command, action, and tool is explained clearly.

**📌 Project Overview**

* **Goal**: Predict oil production volume (BORE\_OIL\_VOL) using well sensor data.
* **Technologies Used**:
  + Python, Pandas, Scikit-learn
  + Jupyter Notebook on EC2
  + Streamlit for frontend
  + GitHub for version control
  + Render for deployment
  + Git Bash for command-line interface

**🧠 EC2 + Git Bash Setup (Minute-by-Minute)**

**🔹 Why Git Bash?**

Git Bash provides a Unix-style shell on Windows, which allows the use of Linux commands like ssh, scp, and chmod. It's required for interacting with AWS EC2 securely and is preferred over cmd/PowerShell due to better compatibility with cloud environments.

**🔹 EC2 Instance Setup**

1. **Service**: AWS EC2 (Ubuntu 22.04)
2. **Instance Type**: t2.micro (free tier)
3. **Key Pair**: Downloaded .pem file (e.g., aws.pem)

**🔹 Configure Security Group (Inbound Rules)**

Inbound rules added:

| **Type** | **Protocol** | **Port Range** | **Source** | **Why?** |
| --- | --- | --- | --- | --- |
| SSH | TCP | 22 | 0.0.0.0/0 | Allows connection to EC2 terminal via Git Bash |
| Custom TCP | TCP | 8888 | 0.0.0.0/0 | Required to open access to Jupyter Notebook on browser |

📌 **Port Explanation**:

* Port 22: SSH (secure shell) — used to remotely control the server
* Port 8888: Jupyter Notebook runs on this port — must be opened to access via browser
* 0.0.0.0/0: Means open to all IPs — fine for dev, restrict to your IP for production security

**🔐 Connecting to EC2 using Git Bash**

Navigate to the folder containing your .pem file:

cd ~/Downloads

Set correct permissions so your private key isn’t publicly viewable:

chmod 400 aws.pem

Then connect:

ssh -i aws.pem ubuntu@<your-ec2-public-ip>

**Explanation**:

* -i aws.pem: specifies your private key
* ubuntu@: default user for Ubuntu
* <your-ec2-public-ip>: visible in AWS EC2 dashboard

**🧪 JupyterLab Setup on EC2**

Install Python, pip, and venv:

sudo apt update && sudo apt install python3-pip python3-venv -y

Create a virtual environment:

python3 -m venv dsenv

* -m venv: tells Python to run the venv module

Activate the virtual environment:

source dsenv/bin/activate

Install required packages:

pip install jupyter pandas numpy seaborn matplotlib scikit-learn boto3 joblib

Start Jupyter server:

jupyter notebook --ip=0.0.0.0 --port=8888 --no-browser

* --ip=0.0.0.0: allow access from any IP
* --port=8888: set port (matches inbound rule)
* --no-browser: prevents it from opening a browser in EC2

Use the token URL and replace localhost with your **EC2 public IP**:

http://<your-ec2-ip>:8888/?token=abc123

**📤 Transfer Files From EC2 to Local Machine (for GitHub Upload)**

On your local Git Bash:

scp -i aws.pem ubuntu@<ec2-ip>:~/dsenv/app.py .

scp -i aws.pem ubuntu@<ec2-ip>:~/dsenv/oil\_volume\_model.pkl .

**Explanation**:

* scp: Secure Copy Protocol
* -i: Use identity file
* : separates remote path from command
* . means copy to current local directory

**🗂️ GitHub Setup with Git Bash**

**Global Config**

git config --global user.name "YourName"

git config --global user.email "you@example.com"

**Initialize Git + Push to GitHub**

git init # start a new git repo

git add . # stage all files for commit

git commit -m "Initial commit" # -m for message inside quotes

git branch -M main # -M: force rename current branch to 'main'

git remote add origin https://github.com/<username>/<repo>.git

# add your remote repo

git push -u origin main # -u sets upstream for future pushes

**🌐 Render Deployment**

**Why Render?**

* Free hosting for Streamlit
* Integrates with GitHub
* Deploys in 1 click

**Deployment Steps:**

1. Login to [Render](https://render.com/)
2. Click **New Web Service**
3. Connect GitHub repo
4. Set:
   * **Build Command**: pip install -r requirements.txt
   * **Start Command**: streamlit run app.py --server.headless true
   * **Instance Type**: Free
5. Click Deploy

App will be publicly available in 1-2 minutes 🎉

**✅ Recap of All Bash Commands Used**

| **Command** | **Explanation** |
| --- | --- |
| chmod 400 aws.pem | Restricts file access (400 = owner read only) |
| ssh -i aws.pem ubuntu@<IP> | Connect to EC2 securely using SSH protocol |
| scp -i aws.pem ... | Copy files between local machine and EC2 |
| sudo apt install ... | Install system packages (requires root) |
| python3 -m venv dsenv | Create isolated Python environment using the venv module |
| source dsenv/bin/activate | Activate virtual environment (changes shell prompt) |
| pip install ... | Installs Python libraries in the active environment |
| jupyter notebook --ip=0.0.0.0 | Launch Jupyter and allow public IP access |
| git init | Start a new Git repository |
| git add . | Add all files to staging area |
| git commit -m "..." | Commit staged changes with a message (-m) |
| git branch -M main | Rename current branch to main, forcefully (-M) |
| git remote add origin ... | Add GitHub repo as remote |
| git push -u origin main | Push to GitHub and set default upstream for future pushes |

This completes your end-to-end deployment documentation. It includes every detail about commands, port usage, inbound rules, Git usage, IP connection logic, and tool choices for a fully reproducible ML deployment workflow.