

Pre-Workshop Setup Guide: Computational Reproducibility in Machine Learning

Waheed U. Bajwa (Rutgers University–New Brunswick)

February 25, 2025

Welcome to the **Computational Reproducibility in Machine Learning** workshop! To ensure a smooth experience, please follow the setup instructions below before attending.

Our default production environment will be **Windows 10/11 with Windows Subsystem for Linux (WSL2)**, along with **VS Code**. The same setup works on Linux natively. However, **Mac users** will need to use a **Docker-enabled Linux virtual machine (VM) via Canonical's Multipass** in order to build a Docker image.

This guide provides setup instructions for both **Windows and macOS**.

1. Create Essential Accounts

If you do not have these accounts already, please create them:

- **GitHub** (for versioning): Sign up [here](#)
- **DockerHub** (for publishing Docker images; can be linked with GitHub): Sign up [here](#)
- **Zenodo** (for publishing research outputs; can be linked with GitHub): Sign up [here](#)

2. Setup Instructions for Windows 10/11

2.1 Enable and Install WSL2

1. Open **PowerShell as Administrator** and run:

```
wsl --install
```

This installs WSL2 along with a default Linux distribution.

2. If WSL is already installed, ensure it is set to version 2:

```
wsl --set-default-version 2
```

3. Install a Linux distribution (e.g., **Ubuntu 24.04 LTS**):

```
wsl --install -d Ubuntu
```

4. Launch the Linux terminal from the **Start Menu**, and when prompted, create a **username** and **password**.

2.2 Install Miniconda

Inside the **Ubuntu terminal**, run:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
```

When prompted, say **“yes”** to the question:

Do you wish to update your shell profile to automatically initialize conda? (yes/no)

To ensure Miniconda is active, restart the shell:

```
source ~/.bashrc
```

(Optional) To **disable auto-activation of Conda’s base environment**, run:

```
conda config --set auto_activate_base false
source ~/.bashrc
```

2.3 Install VS Code

Download and install **VS Code**: Download here

- **For Windows**, select the **User Installer** version.

Inside VS Code:

- Install the **Remote Development** extension pack from Microsoft to develop inside WSL.
 - Install the following additional extensions:
 - **Docker** from Microsoft
 - **GitHub Repositories** from GitHub
 - **GitHub Pull Requests** from GitHub
 - **Jupyter** from Microsoft
 - **Python** from Microsoft
 - **Remote Repositories** from Microsoft
 - Additional extensions for **R**, **MATLAB**, etc., if needed.
-

2.4 Install Git

Download and install **Git for Windows**: Download here

Verify installation:

```
git --version
```

2.5 Install Docker

1. Download and install **Docker Desktop**: Download here
2. During installation, **select WSL2 as the backend**.
3. Enable the **WSL integration** in Docker settings.
4. Verify installation:

```
docker --version
```

3. Setup Instructions for macOS

3.1 Install Homebrew

Install **Homebrew** (Mac package manager) by running:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Verify installation:

```
brew --version
```

3.2 Install VS Code

Download and install **VS Code for macOS**: Download here

Inside VS Code:

- Install the following extensions:
 - **Docker** from Microsoft
 - **GitHub Repositories** from GitHub
 - **GitHub Pull Requests** from GitHub
 - **Jupyter** from Microsoft
 - **Python** from Microsoft
 - **Remote Development** from Microsoft
 - **Remote Repositories** from Microsoft
 - Additional extensions for **R**, **MATLAB**, etc., if needed.
-

3.3 Install Git

Install Git via Homebrew:

```
brew install git
```

Verify installation:

```
git --version
```

3.4 Install Docker

Download **Docker for macOS**: Download here

- Ensure you select the correct version for your processor (Intel or Apple Silicon).
 - Verify installation:

```
docker --version
```
-

3.5 Configure SSH for Remote Access to the Linux VM

Generate an SSH key:

```
ssh-keygen -t rsa -b 4096
```

Copy the public key from `id_rsa.pub`:

```
cat ~/.ssh/id_rsa.pub
```

Create a **cloud-config.yaml** file for SSH access to the Multipass Linux VM (save it where your public key is stored):

```
#cloud-config
users:
  - name: ubuntu
    sudo: ['ALL=(ALL) NOPASSWD:ALL']
    ssh-authorized-keys:
      - ssh-rsa <PASTE YOUR PUBLIC KEY HERE>
```

3.6 Install Multipass and Launch a Docker-enabled Linux VM

Download and install **Multipass for macOS**: Download [here](#)

Verify installation:

```
multipass version
```

Launch a Docker-enabled Multipass instance with SSH support:

```
multipass launch docker --name vscode-docker --memory 8G --disk 80G --cpus 2
                        --cloud-init ~/.ssh/cloud-config.yaml
```

Check the instance's IP address:

```
multipass list
```

In **VS Code**, go to **Remote Development**, open SSH settings, and add:

```
Host multipass-vscode-docker
  HostName <MULTIPASS_INSTANCE_IP_ADDRESS>
  User ubuntu
```

Click on the host name **multipass-vscode-docker** and connect.

3.7 Install Miniconda within the Multipass Linux VM

Once inside the **Multipass shell**, install Miniconda:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
```

Follow the same **Conda activation steps** as in the Windows setup.

(Optional Note): If you are unable to connect to your Multipass Linux VM using VS Code, you can still perform **Docker-based activities** within the Linux VM (with a mounted folder inside your system). Meanwhile, the rest of the activities can be carried out using **Miniconda installed on macOS** (installation guide) along with **VS Code on the host macOS**.

3.8 Managing the Multipass Instance

To stop (hibernate) the instance:

```
multipass stop vscode-docker
```

To delete the instance:

```
multipass delete --purge vscode-docker
```

4. Final Steps Before the Workshop

- Ensure you can create files using VS Code inside your WSL (Windows) or Multipass Linux VM (Mac).
- Play around with Conda, Git, and Docker before the workshop, to the extent possible.
- Test that Jupyter Notebooks work inside your environment.

This guide ensures that your environment is properly set up before the workshop. Please complete all steps and reach out if you encounter issues. See you at the workshop!