Development Environment Setup

A Craftsman Must Sharpen His Tools to Do His Job

Pascal Sun

UWA NLP-TLP Group

The University of Western Australia

November 24, 2024

Table of Contents

1 Background

2 Terminal and Shell

3 WSL2 and Homebrew

4 IDE

5 Git and GitHub

6 Docker and Docker Compose

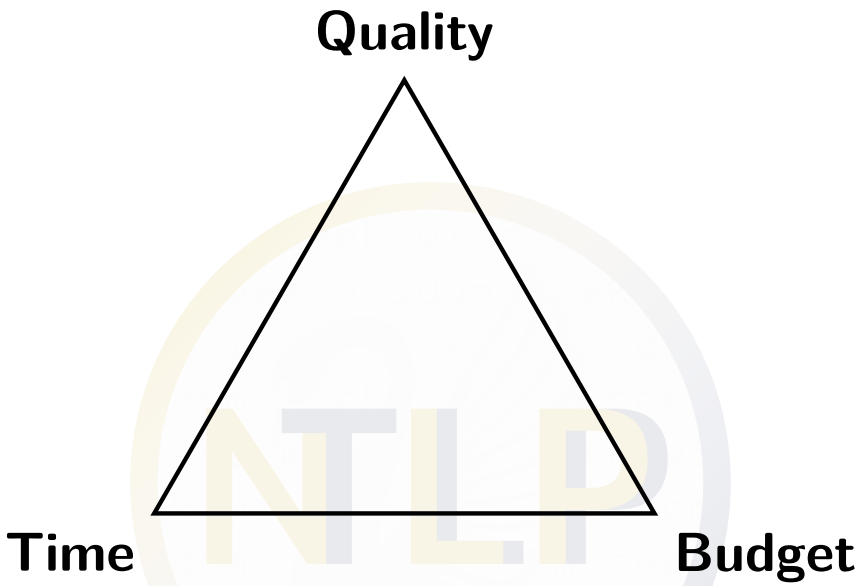
7 AI as your code buddy

8 Demo

9 Q&A

Background

The Impossible Triangle



Proper Developer Kits:

Reduce Time ↓ | Increase Quality ↑

Budget optimized through time efficiency

Learn from the Best Practice

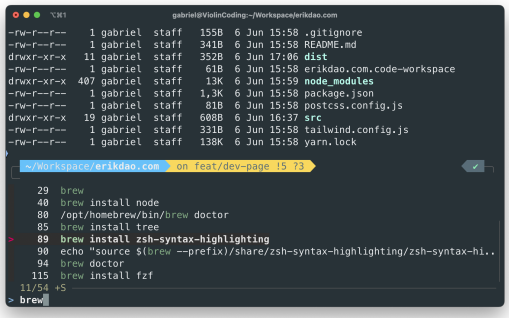
Required Kits

- Terminal and Shell
- WSL2 and Homebrew
- IDE
- Git and GitHub
- Docker and Docker Compose
- AI and capability to use them

Terminal and Shell

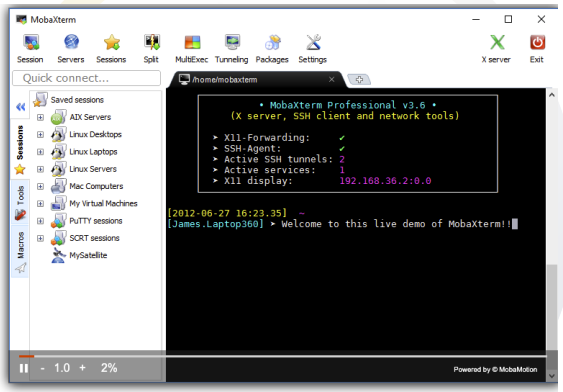
Terminal

Terminal: Essential developer interface that stands the test of time



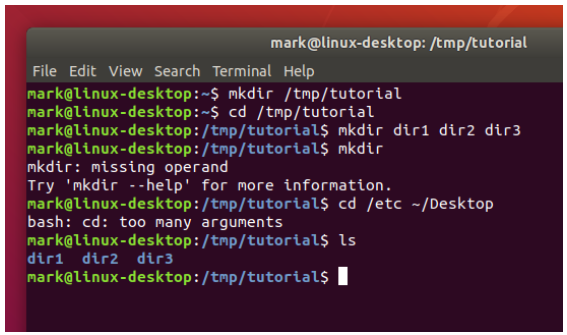
Mac: iTerm2

https://iterm2.com/



Windows: MobaXterm

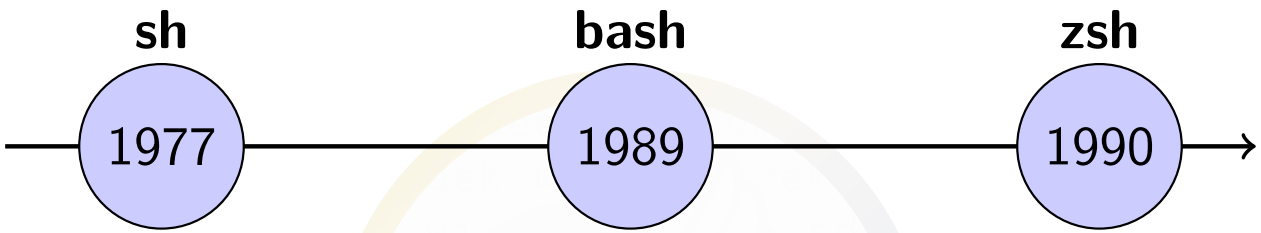
https://mobaxterm.mobatek.net/



Linux: Default

Built-in terminal

Evolution of Shells



Terminal

- Text Input/Output interface
- Handles display & keyboard
- Examples: iTerm2, MobaXterm

Shell

- Command interpreter
- Processes commands
- Examples: bash, zsh

Terminal is like a **PC**, Shell is the **Program** inside

sh (Bourne Shell)

Core Details

- Created by Stephen Bourne (Bell Labs, 1976)
- Original Unix Version 7 shell
- Command-line interpreter & scripting

Key Contributions:

- Introduced shell scripting
- ALGOL-inspired syntax
- Base compatibility layer

Significance
Foundation for all modern shells

bash (Bourne Again Shell)

Created by Brian Fox (GNU Project, 1989)

Enhanced Features

- Command history
- Job control
- Aliases & functions

Significance

- Default in Linux
- Backward compatible
- Balanced usability

Fun Fact: Name is a pun on "Bourne Again"

zsh (Z Shell)

Created by Paul Falstad (1990)

Advanced Features

- Intelligent autocompletion
- Customizable themes
- Plugin support
- Global aliases
- Advanced scripting

Modern Status
Default shell in
MacOS and Kali Linux
since 2019 and 2020

Perfect for power users who value productivity and customization

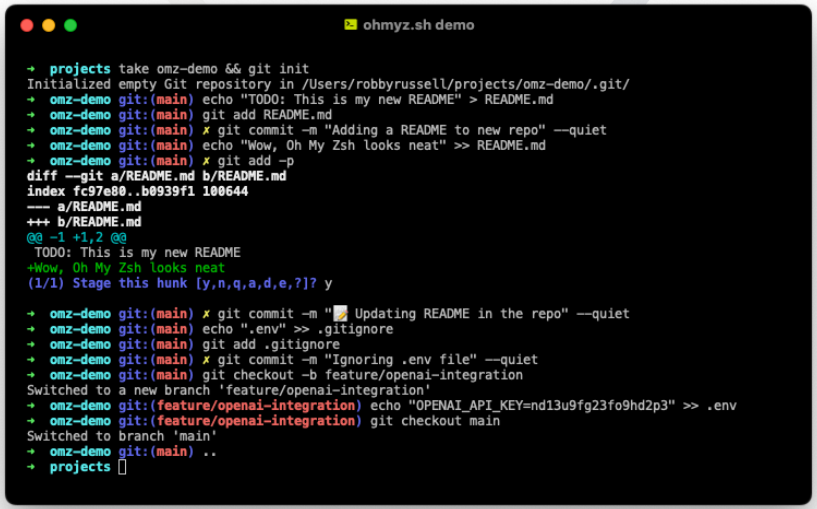
zsh and oh-my-zsh

Shell Evolution

- **sh** (1976)
  - Original Unix shell
- **bash** (1989)
  - Enhanced features
  - Default in most Linux
- **zsh** (1990)
  - More powerful
  - Default in macOS

Oh-My-Zsh

- Framework for zsh
- **Key Features:**
  - 300+ plugins
  - Custom themes
  - Auto-completion
  - Git integration



Homework

Your Path to a Better Terminal Experience[1]

1. **Choose Your Terminal**
   - Mac → iTerm2 (`https://iterm2.com/`)
   - Windows → MobaXterm (`https://mobaxterm.mobatek.net/`)
   - Linux → Default Terminal
2. **Setup Shell Environment**
   - Install zsh
   - Install oh-my-zsh
3. **Enhance with Powerlevel10k**
   - Install theme

Result: A powerful, customized, and efficient development environment

[1] `https://tutorial.nlp-tlp.org/`

WSL2 and Homebrew

Attention! Windows users

Why Unix-based Systems?

- Most servers run on Linux
- Much better developer experience and efficiency

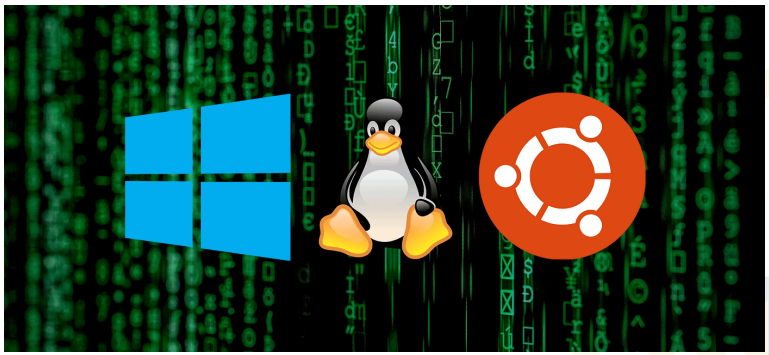
Using Windows for Development?

- Windows Subsystem for Linux (WSL2)
- Provides full Linux kernel
- Native Linux experience on Windows

Recommendation:

- Mac/Linux: Ready for development
- Windows: Install WSL2 (No need to replace your laptop!)

Install WSL2 and Ubuntu22.04 on Windows



Windows + Ubuntu:
Best of both worlds

Open PowerShell as Administrator
for all commands

1. Install WSL2

- Run command:
  `wsl --install`
- Restart computer

2. Install Ubuntu22.04

- Run command:
  `wsl --install -d Ubuntu-22.04`
- Set username & password

3. Setup in WSL Terminal

- Open Ubuntu terminal
- Do the work mentioned above

Some references: [2]

[2] `https://linuxbeast.com/blog/install-ubuntu-20-04-or-22-04-in-wsl-2-on-windows-10/`

Package Managers: Developer's App Store

What is a Package Manager?

- Like App Store for developers
- Install, update, and manage software
- Command-line based

APT (Advanced Package Tool)

- Ubuntu's native package manager
- Example: `apt install git`
- Used for system packages

Homebrew

- Originally for macOS
- Now available for Linux too
- Example: `brew install git`

Homework

1. Install WSL2 and Ubuntu 22.04 if you use Windows[3]
2. Install Homebrew (Mac, Linux, WSL2 for Windows)[4]

[3] https://tutorial.nlp-tlp.org/#wsl2--ubuntu22.04
[4] https://tutorial.nlp-tlp.org/#brew-on-linux

IDE

IDE (Integrated Development Environment)

What is IDE?

- More than just text editor
- Auto completion of code
- Syntax highlighting
- Debugging tools
- Version control integration

Popular IDEs

- Visual Studio Code - Free, open-source by Microsoft
- JetBrains Series - Paid, free for students:
  - PyCharm - Specialized for Python development
  - WebStorm - For JavaScript/Web development

Which one should we use?

For Students

Use **JetBrains Series** - Free student license available[a]

[a] https://www.jetbrains.com/lp/leaflets-gdc/students/

Why JetBrains over VSCode?

- VSCode requires setup & extensions knowledge
- PyCharm provides built-in best practices
- Better learning environment for beginners
- "Ready-to-code" experience

VSCode vs Jetbrain PyCharm

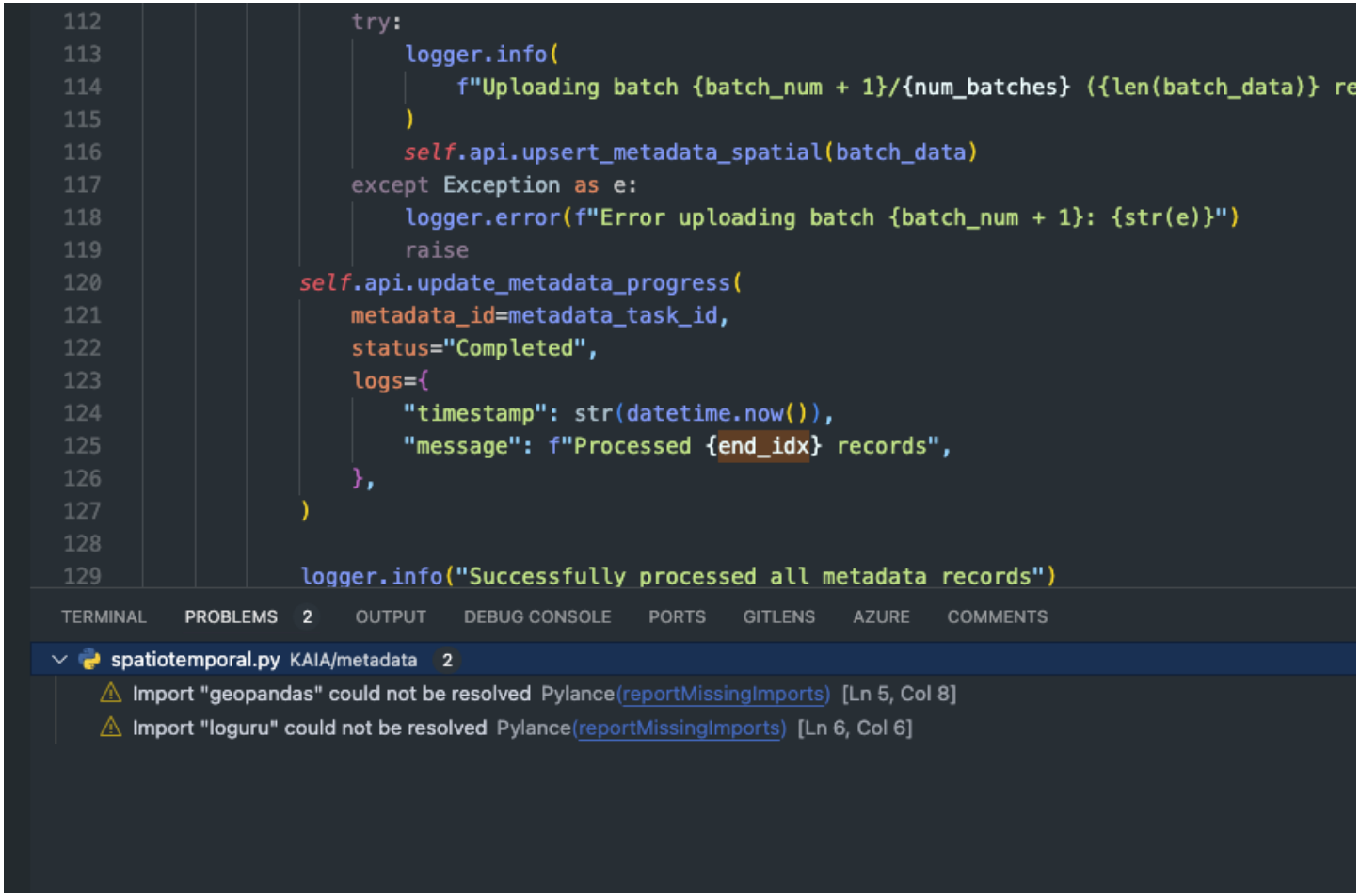


Figure: VSCode: out-of-box

VSCode vs Jetbrain PyCharm

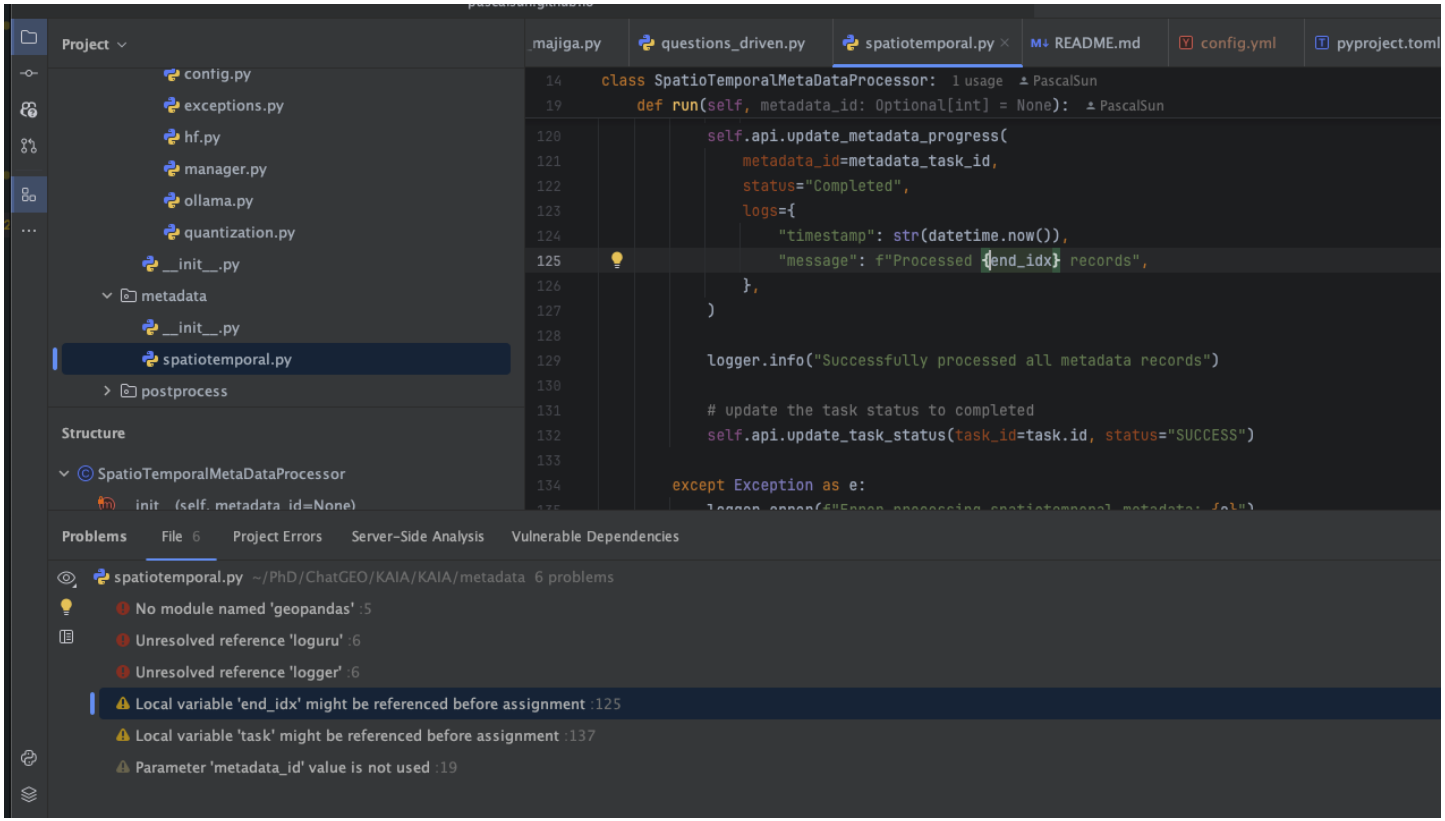
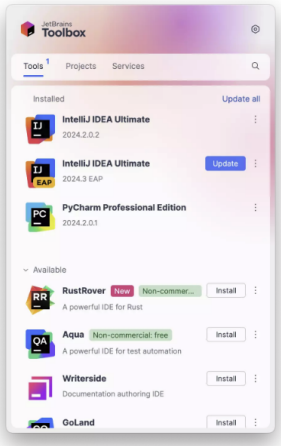
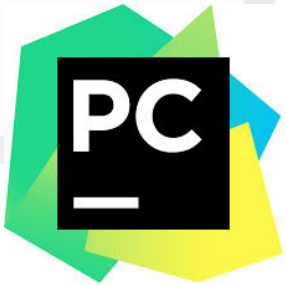


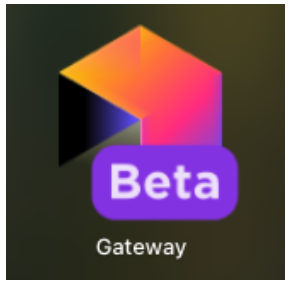
Figure: PyCharm: out-of-box

Homework

- Install JetBrains Toolbox[a]
- Through Toolbox, install:
  - PyCharm Professional
  - WebStorm
  - Gateway: Equal to VS Code Remote SSH







[a] https://www.jetbrains.com/toolbox-app/

Git and GitHub

Why do we need Git?

Before Git...

How we used to manage versions:

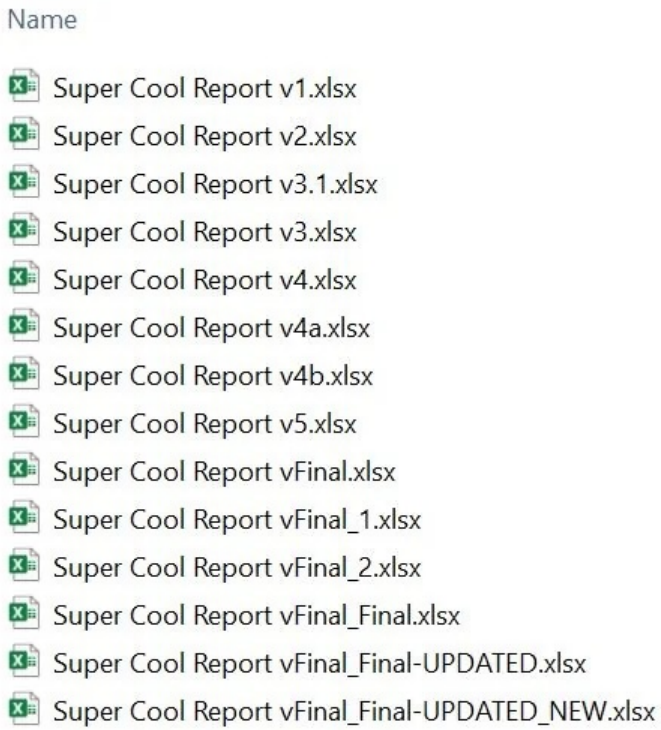


Figure: The classic "final_final_FINAL_v2" problem

And Thanks to this man

Figure: Linus Torvalds' Workspace

Figure: Linux Foundation Headquarters

Creator of Linux & Git: Linus Torvalds

Why Git is Great for Code Management?

The Problem:

- Code changes constantly
- Multiple people work together
- Need to track who changed what

How Git Helps:

- Saves complete snapshots of code
  (Not just the changes - the whole picture)
- Makes teamwork simple
  (Everyone can work independently)
- Easy to undo mistakes
  (Go back to any previous version)

Result: Safe, organized, and efficient code management[5]

[5] We will touch base detailed explain later

Git vs GitHub: What's the Difference?

Git: The Tool

- Version control software on your computer
- Tracks changes in your code
- Works completely offline

GitHub: The Platform

- Website that hosts Git repositories
- Adds collaboration features
- Like a cloud service for Git projects
- Collaborate with Teammates
- GitLab, Bitbucket are competitors of GitHub
- Acquired by Microsoft

Relationship:

- Git is the tool, GitHub is the service
- Like Microsoft Word (tool) and Google Drive (storage service)

Git & GitHub Homework

1. Install Git on Your Computer

- Mac: Use Terminal command
  `brew install git`
- Windows: Treat it as Linux via WSL2
  Install it through the WSL2 Ubuntu 22.04 Terminal
  So the command can be either
  `brew install git | apt install git`
- Verify installation:
  `git --version`

2. Setup GitHub Account

- Create account at github.com
- Get student benefits:
  `https://education.github.com/pack`
- Verify your student status using school email

Docker and Docker Compose

It works on my machine

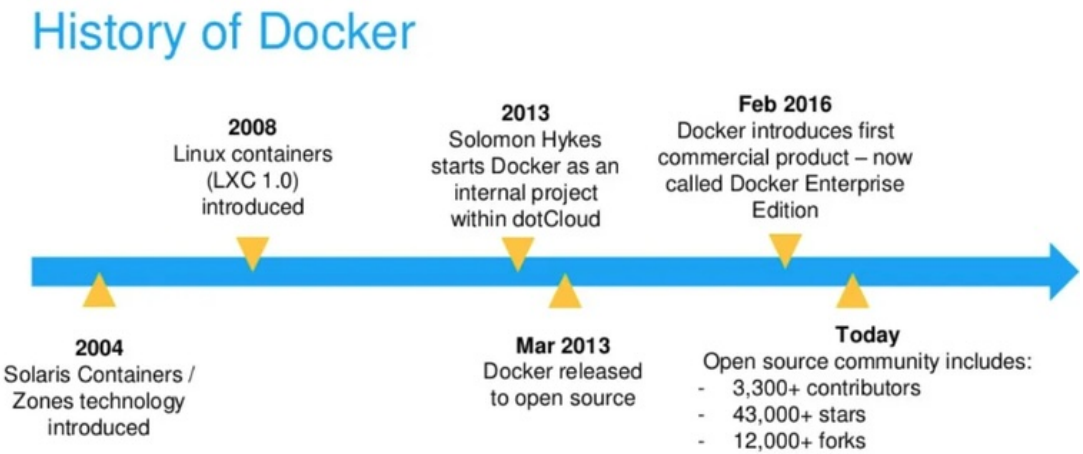
Figure: It works on my machine

How to solve this problem?

Ship your machine to customers.

But how?

Use Docker



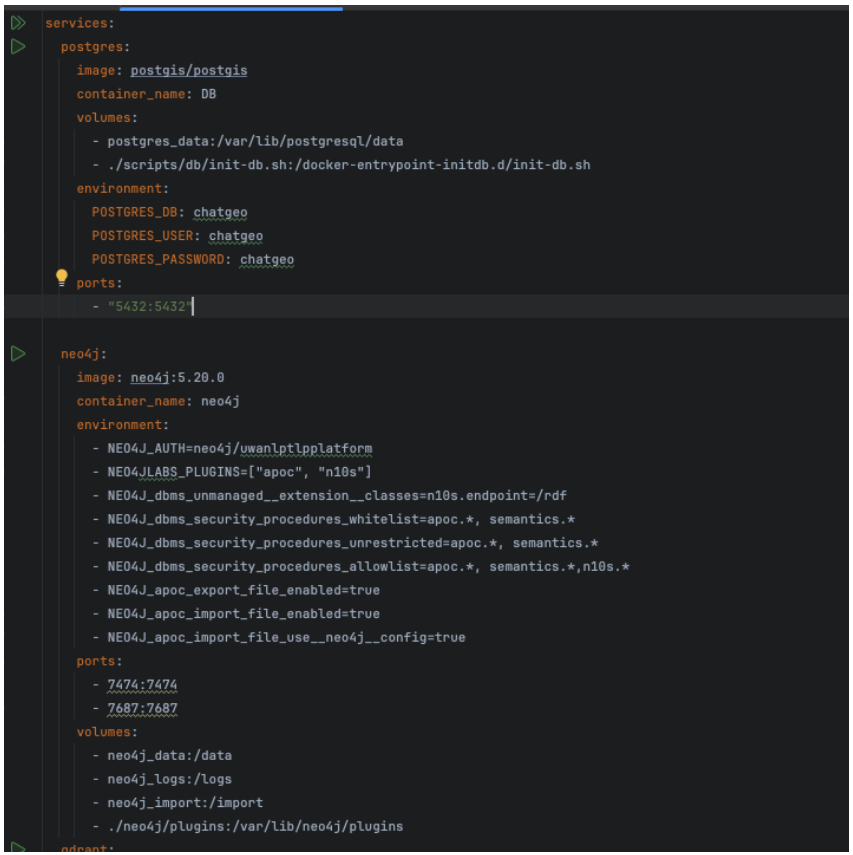
Docker: The foundation infrastructure in the modern world.

A typical Dockerfile

```dockerfile
# Use an official Python runtime as a parent image
FROM python:3.9

RUN apt-get update -y && apt-get install libhdf5-dev -y
# Install PostgreSQL client
RUN apt-get update && apt-get install -y postgresql-client ffmpeg
RUN apt-get install -y gdal-bin libgdal-dev python3-gdal
# Set work directory
WORKDIR /usr/src/app

# Install dependencies
COPY ./requirements.txt /usr/src/app/requirements.txt
COPY ./requirements.dev.txt /usr/src/app/requirements.dev.txt
RUN pip install --upgrade pip

# RUN pip install --upgrade pip setuptools wheel
RUN pip install --no-cache-dir -r requirements.txt
RUN pip install --no-cache-dir -r requirements.dev.txt

# Copy the current directory contents into the container at /usr/src/app
# copy but ignore the files in .dockerignore
COPY . /usr/src/app/

# Make sure the script is executable
RUN chmod +x /usr/src/app/scripts/api/entrypoint.sh
```

A Dockerfile defines how to build your "machine"

Docker Compose: Managing Multiple Containers



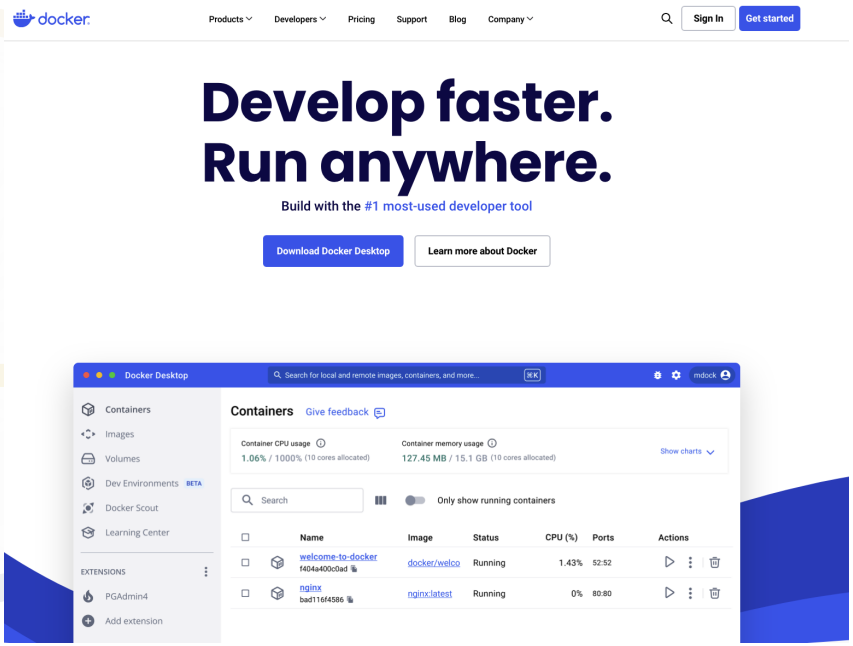
```yaml
services:
  postgres:
    image: postgis/postgis
    container_name: DB
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./scripts/db/init-db.sh:/docker-entrypoint-initdb.d/init-db.sh
    environment:
      POSTGRES_DB: chatgeo
      POSTGRES_USER: chatgeo
      POSTGRES_PASSWORD: chatgeo
    ports:
      - "5432:5432"

  neo4j:
    image: neo4j:5.20.0
    container_name: neo4j
    environment:
      - NEO4J_AUTH=neo4j/uwanlptlpplatform
      - NEO4JLABS_PLUGINS=["apoc", "n10s"]
      - NEO4J_dbms_unmanaged__extension__classes=n10s.endpoint=/rdf
      - NEO4J_dbms_security_procedures_whitelist=apoc.*, semantics.*
      - NEO4J_dbms_security_procedures_unrestricted=apoc.*, semantics.*
      - NEO4J_dbms_security_procedures_allowlist=apoc.*, semantics.*,n10s.*
      - NEO4J_apoc_export_file_enabled=true
      - NEO4J_apoc_import_file_enabled=true
      - NEO4J_apoc_import_file_use__neo4j__config=true
    ports:
      - 7474:7474
      - 7687:7687
    volumes:
      - neo4j_data:/data
      - neo4j_logs:/logs
      - neo4j_import:/import
      - ./neo4j/plugins:/var/lib/neo4j/plugins
  adpapt:
```

Docker Compose: Orchestrate multiple containers in one configuration
Everything will simplify to `docker compose up`

Homework

Install Docker

- Docker Desktop includes Docker Compose and Docker
- Windows users: Install in both WSL2 & Windows[a]



[a] tutorial.nlp-tlp.org/local-development-environment-setup/how-to-install-docker

AI as your code buddy

How should we handle AI?

Embrace it, you have no other options

Modern AI Development Tools:

- **Code Completion**
  - GitHub Copilot
- **Chat Assistants**
  - Claude
  - GPT-4
- **AI-Powered IDEs**
  - Cursor (cursor.com)

Will AI replace developers?

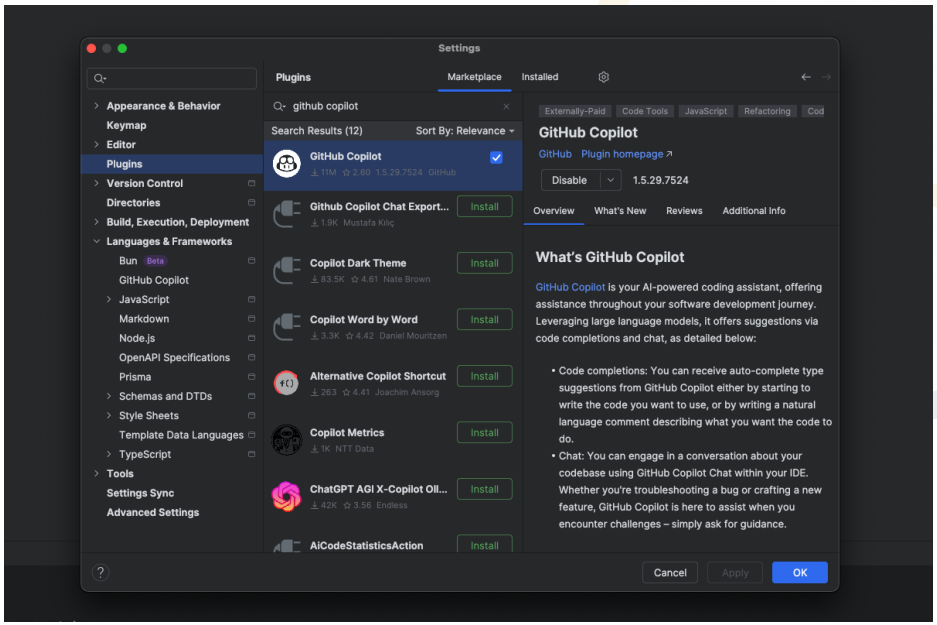
- I do not know.
- **Why this Bootcamp?**
  - Industry call for developers to get the job done, whatever it takes
- **Current State of AI**
  - Can handle repetitive tasks
  - Can generate code from instructions
  - Cannot design, architect and implement complex solutions
- **Developer's Role Today**
  - Like a Lego master builder
  - Organizes components into new products
  - Requires "Full Stack" knowledge

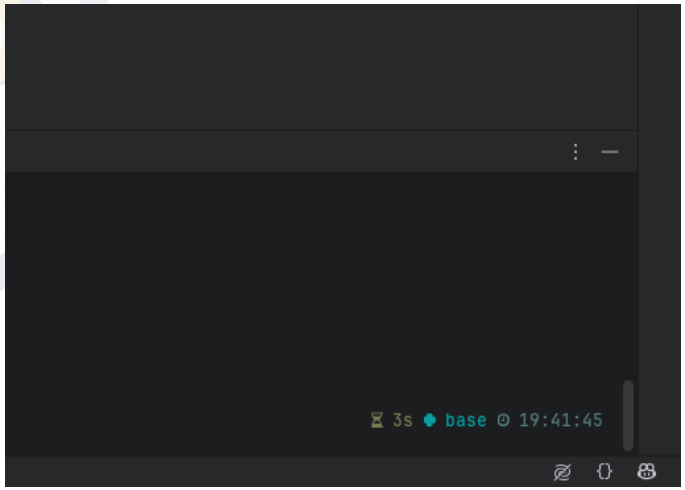
What we will use in this Bootcamp?

- **Github Copilot**
  - Free with student GitHub account
- **Claude**
  - Paid service, 20USD per month
  - Recommended investment
  - You can cancel if you do not think it is good
- **Best practice to use them?**
  - You must clearly know what you want to achieve.

Homework

- Grab the free license from GitHub
- Install Github Copilot in your IDE:
  - VSCode, PyCharm, WebStorm





Demo

Q&A

Q&A

Questions?

Feel free to ask anything