



# Programming Environment



Manas Jyoti Das, PhD  
Computer Science



# Why we need an environment?

---

- Consistency
- Closed ecosystem
- Growth

# Why environment for software dev?

---

- Provides a good way to manage packages
- Dependency management
- Also known as virtual environments
- Keep things separate for different project requirements
- Options: venv (python 3), anaconda/miniconda (python, ruby, scala, c/c++), mamba, etc

# Installing a virtual environment

---

windows/mac/linux:

<https://docs.conda.io/projects/miniconda/en/latest/miniconda-install.html>

Problem in windows: Not connected to the terminal, you have to use Anaconda prompt.

Solution:??

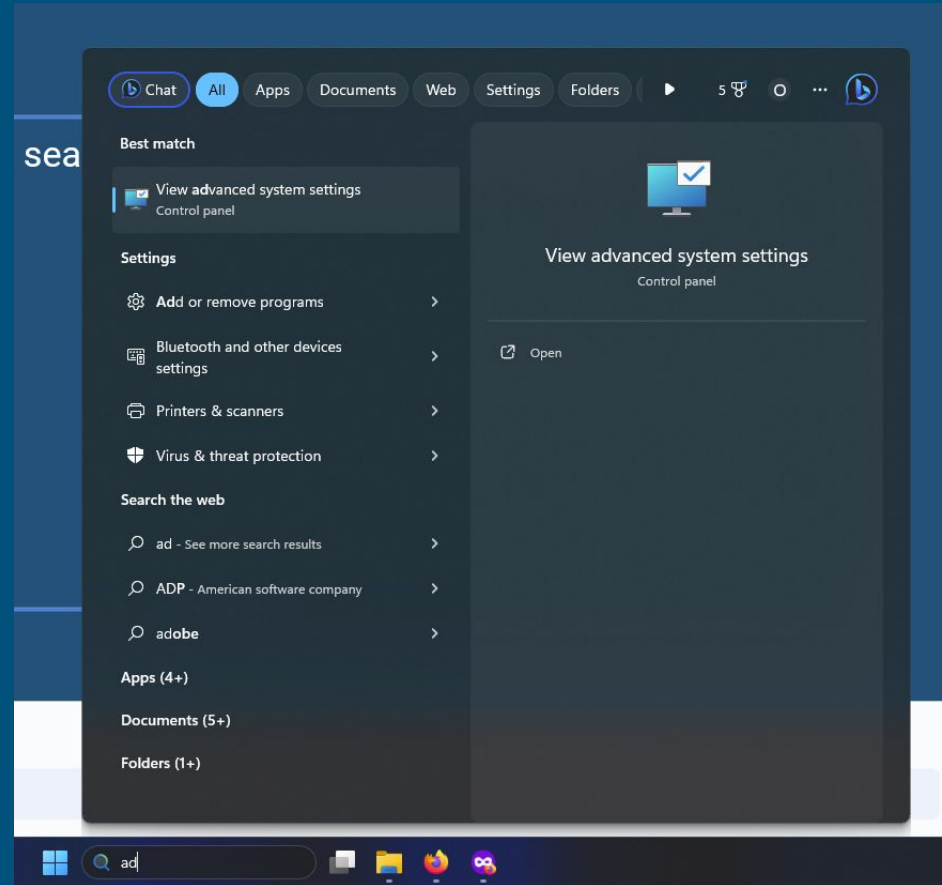
# Solution to win problem

---

- Open anaconda prompt, search for “anaconda prompt” in search bar of windows. Open the “anaconda prompt”
- Type the command : where conda (inside of the anaconda prompt)
- You will get results based on your system where you installed miniconda, and understand the paths carefully
- The results looks like: C:\User\username\Anaconda3 (example)

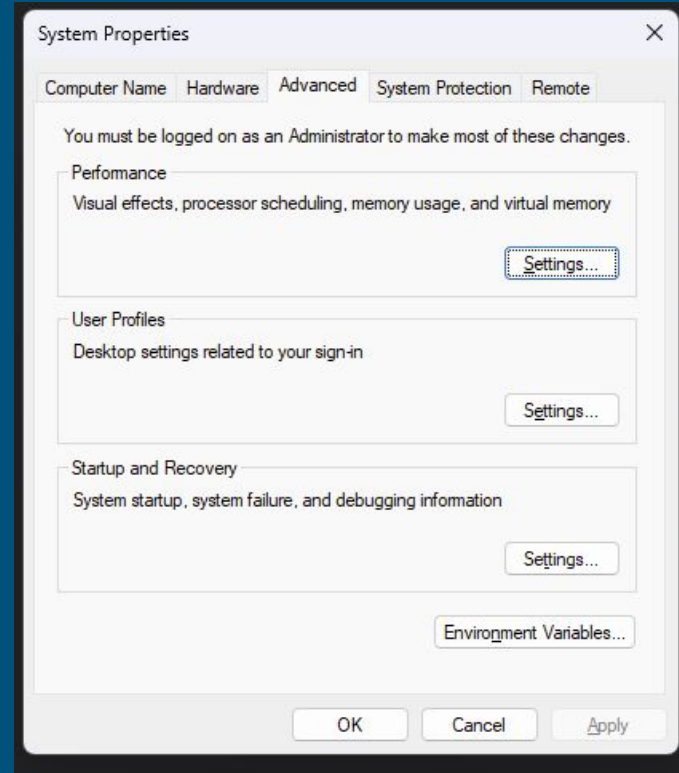
# contd...

- In windows search bar, search for advanced system settings
- Click on “view avan...”



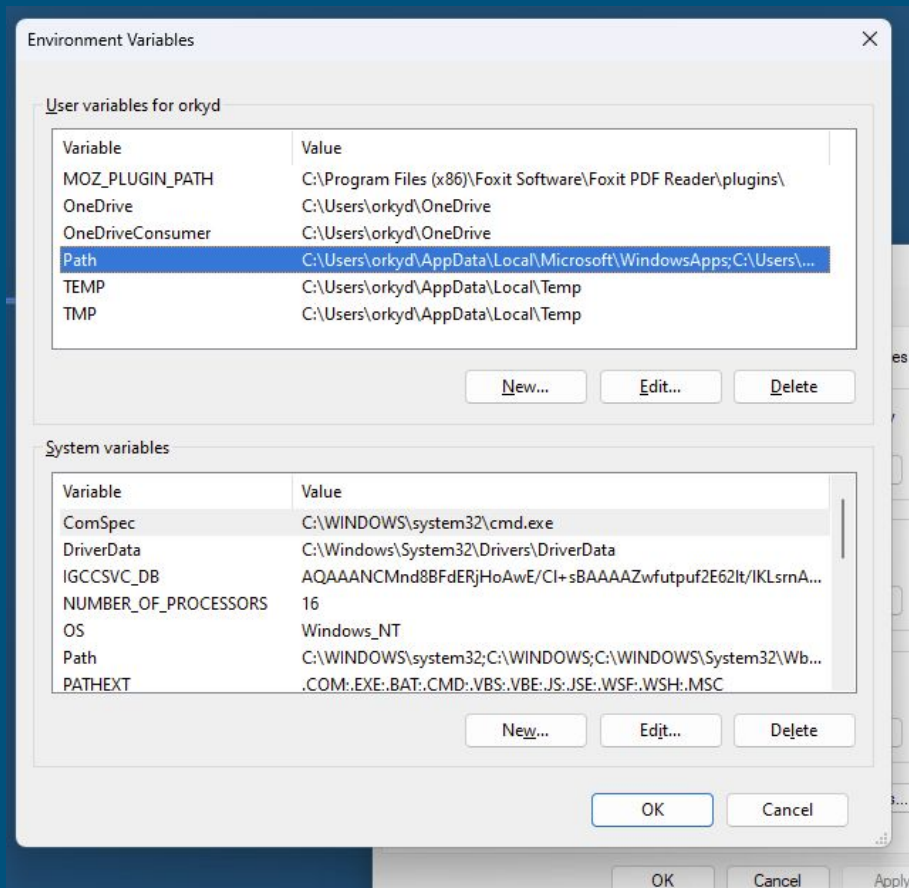
# contd...

- Next click on “environment variables”



# contd...

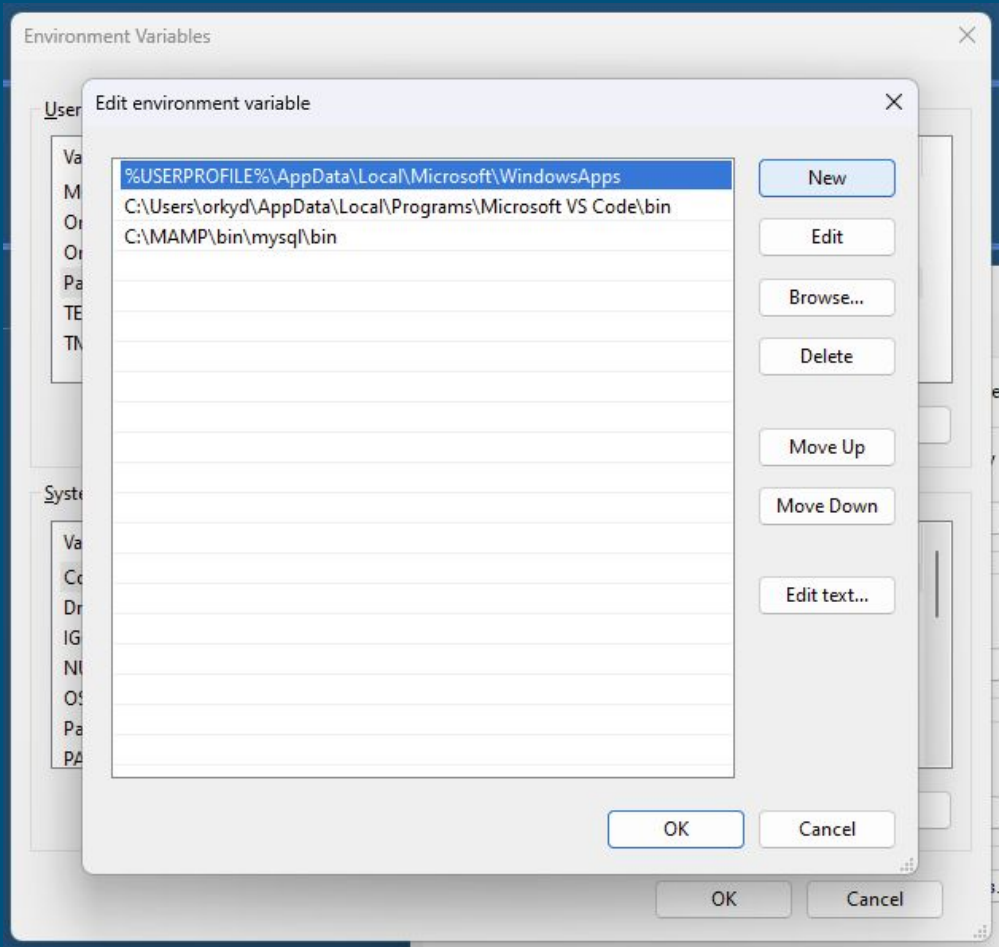
- Next click on “path” and  
Click “edit”





# contd...

- Click on “new”



# contd..

---

- Add these paths separately by clicking new for each entry
- Paths to include:
  - C:\Users\userName\Anaconda3\Scripts
  - C:\Users\userName\Anaconda3
  - C:\Users\userName\Anaconda3\Library\bin
  - C:\Users\userName\Anaconda3\Library\mingw-w64\bin (if it is there in your system)
  - C:\Users\userName\Anaconda3\Library\user\bin (if it is there)

\* After that, fire up the terminal and type : `conda - -version` (do not copy past the command, due to formatting)

\* If not working have a look at the environments again.

# Conda commands basics

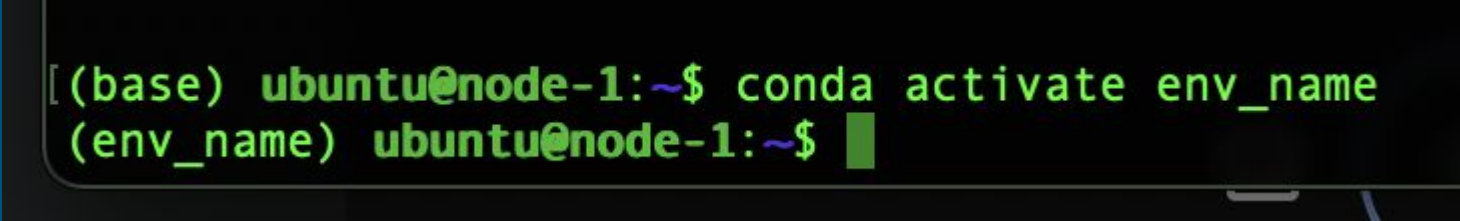
---

- To list all the environments: `conda env list`
- Initially it will be “base”
- To activate “base” environment the command is: `conda activate base`
- To deactivate an environment the command is: `conda deactivate`

# Creating a new environment

---

- Command: `conda create --name env_name python=3.9`
- Instead of 3.9 you can give any python version number
- Instead of env\_name give a appropriate name of your choice
- Then activate the created environment: `conda activate env_name`
- If it is activated you will see the environment name in your terminal prompt.  
As shown in the below image. Base -> env\_name
- Sanity check: `python3 --version`

A terminal window with a dark background and green text. The first line shows the prompt `(base) ubuntu@node-1:~$` followed by the command `conda activate env_name`. The second line shows the prompt has changed to `(env_name) ubuntu@node-1:~$` followed by a green cursor block.

```
(base) ubuntu@node-1:~$ conda activate env_name
(env_name) ubuntu@node-1:~$
```

# Installing packages

---

- Always read reviews and read many articles before installing packages
- There are many packages in the world for a task at hand. Be aware of fake packages
- Always do the research
- Search like example “python package to scrap an URL” then read about it in many articles form a consensus
- Search for that package like conda xyz package
- Almost all the packages are there in conda repository
- If not there are ways to install it in conda using pip3

# Packages installer

---

- By default the package installer is pip3 for python3.
- pip3 and pip are not similar, one is for python3 another is for python(v2). It depends on the creator of the package. Sometimes you can use pip to install software for python3 without any problem inside an active environment
- Not all packages may be there in conda. You may have to use pip. Always use pip it inside of an active environment
- Usually the command looks like: `pip install package_name`

# Clone an environment

---



- Command: `conda create --name new_env --clone env_name`
- Give appropriate name to new environment
- Removing an environment:
  - `conda remove -n env_name --all` (use `--name` or `-n`)

# To search for a package for different version

---

Command: `conda search pkg_name` (example: `conda search pandas`)

- To list all installed packages: `conda list` (packages installed with pip will be marked)
- Removing a package: `conda remove pkg_name` (note all other dependencies will also get removed)
- Installing a particular version: `conda install numpy=1.21.5` . General version: `conda install pkg_name=version`



# Updating an existing package

---

## Example

- `conda install numpy=1.21.1`
- You want to use v1.25.0 instead of v1.21.1
- `conda install numpy=1.25.0`
- It will automatically update the previous package
- Use `conda list` for sanity check
- `conda update pkg_name` (but will update to the latest version)

# YAML !!!

---

- Export the configuration of an environment to a file
- The file extension used is yml/yaml
- Activate the environment that you want to export
- Inside the environment type: `conda env export > requirement.yaml`
- Where the file is getting saved? Look at the command prompt where you are right now, the path

# To create a new environment

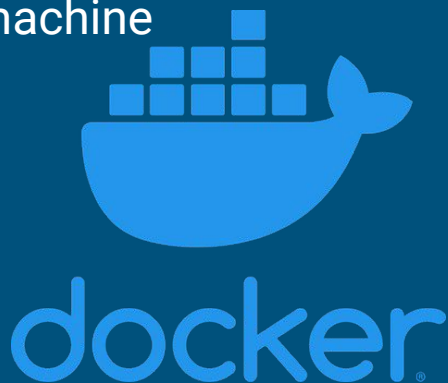
---

- The developer can send the requirement.yaml file to another developer and it will install all the dependencies on their system
- To create a new environment from an yaml file,
- command-1: `conda create --name my_yaml_env --file requirement.yaml` (you can give a new name to the environment)
- OR command-2: `conda env create -f requirement.yaml` (this will have the same environment as the previous environment)
- `conda clean -p` (clean up the cache)

# Other forms of environments

---

- Containers: for portability, isolation, scalability and efficiency. What does it means??
- Examples: **docker**, podman, singularity ,etc
- Kubernetes (K8s) why K8s??
- Do not confuse between virtual environment and virtual machine
- Many a times docker and container means the same

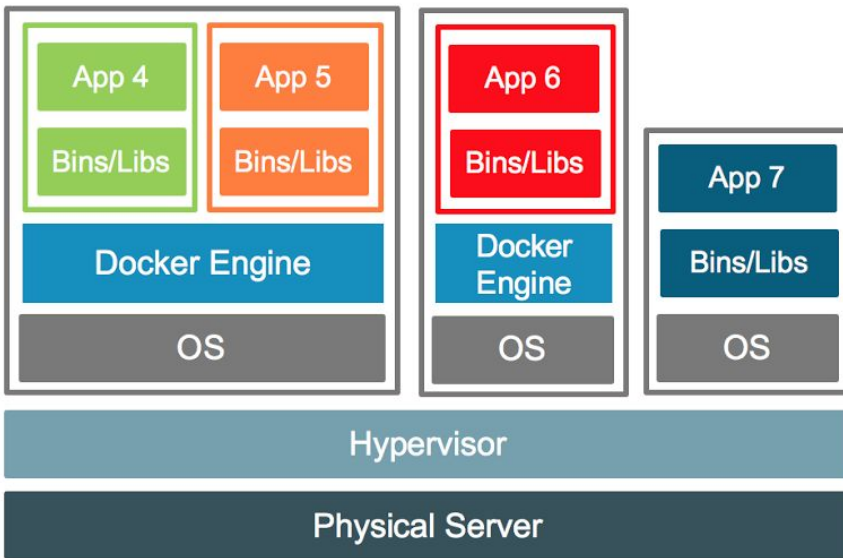
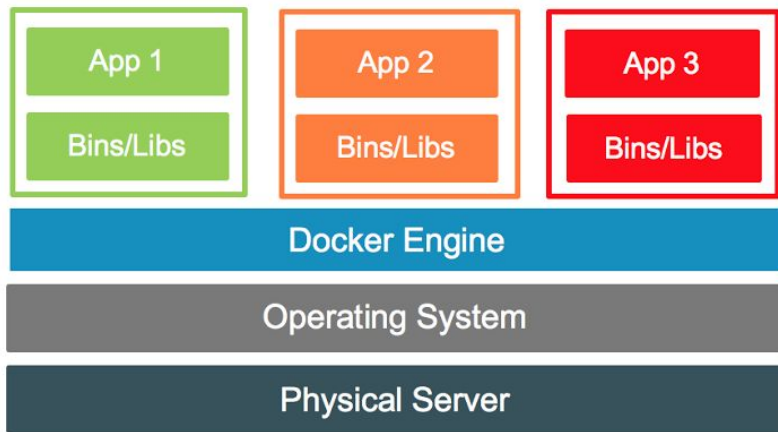


# Container

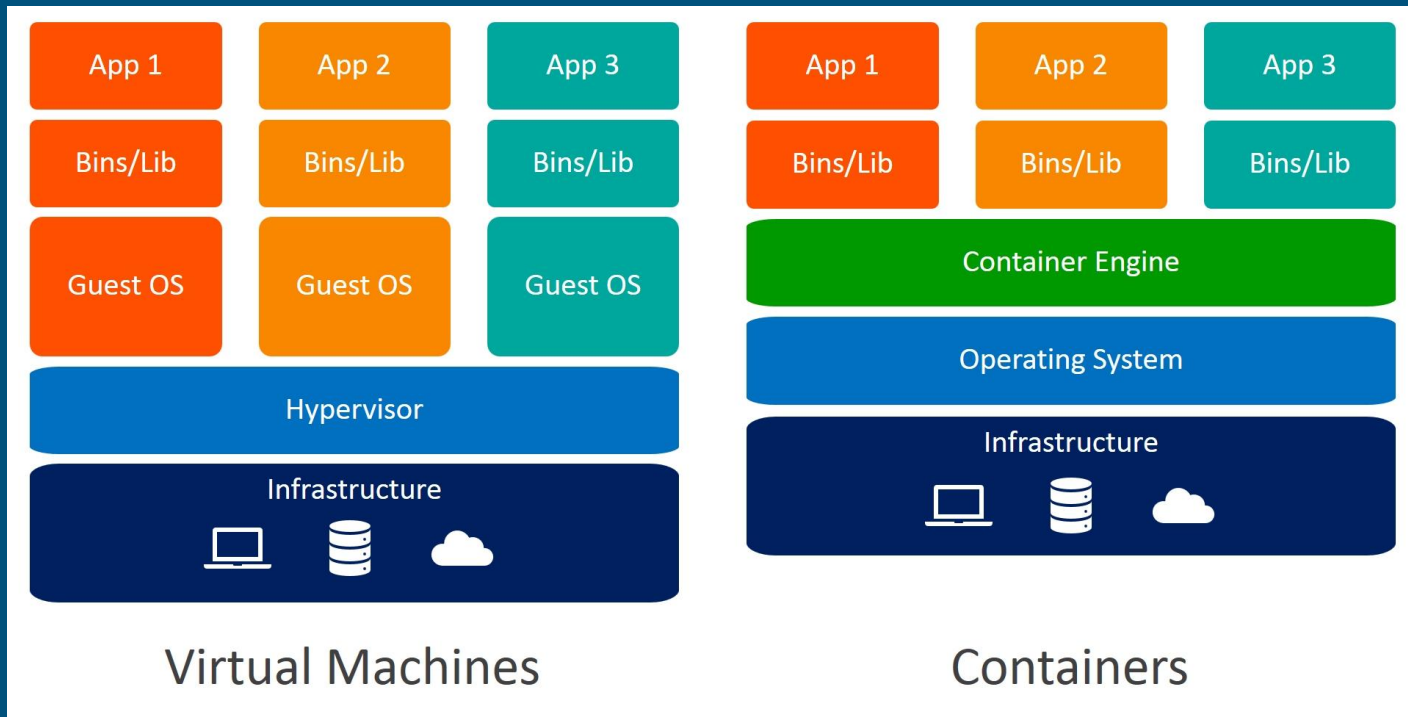
---

- A pre-packaged box for your software
- This box will contain everything: code, libraries, settings and dependencies
- You can spawn multiple containers: for example, can spawn multiple services in a computer server for different users. Also known as microservices.

# Container



# Container vs Virtual machines



# Virtual environment vs Virtual machines(VM)

---

- VM: Virtualize the entire hardware layer, creating a complete computer system with its own CPU, memory, storage, and operating system
- Virtual environments: Are software-level isolations within a single operating system
- VM: Require significant resources as they need to emulate all the hardware of a computer
- Virtual environments: Are lightweight and use resources shared with the main operating system



# Thank you and see you all in the next class

---

- Hope if time permits we can learn about container towards the end of the course