# Introduction to Apptainer

# The Goal:

- Utilizing **AllenSDK**: a python package which facilitates the download and manipulation of Allen Institute data sets
- We will use AllenSDK through a **Compute Canada Account**: a High Performance Compute system
    - Compute Canada gives us access to storage + computing resources that are much more powerful than using our desktop computers alone; good for working with large amounts of data
- We can create jobs for Compute Canada to run by sending scripts via **containers**
- Because Compute Canada discourages the use of virtual environments (like conda), we create a container for it

# Container Overview

- A way of packaging up code + its dependencies so that it may run smoothly on between different computing environments (our computer vs the Compute Canada Computer Node)
- Similar to a Virtual Machine, but the main difference is that it shares the same hardware as the host computer it is set up on
    - Just has its own operating system

# Image File (.sif)

- Contains scripts describing what sort of processes we would like to run
- Is Immutable
- Creating a .sif file: several ways
    - pull from a cloud source like Docker (`apptainer pull <imageName>.sif <source>`)
    - build according to a ‘recipe file’ → a .def file which describes the buildspecs
        - we can write the recipe file then execute

          ```
          apptainer build <imageName>.sif <recipeFile.def>
          ```

# Recipe File (.def)

- Create this locally to make an image
- Various fields can be edited to make specifications for your image

```
Bootstrap: docker
From: ubuntu:22.04
### explanation for the
%help
    Creates and activates the base allensdk virtual environment. Ensure that the env                              e so it can b
e set up correctly.
### copies the files we have on our computer into the container
%files
    allensdk_env.yml /allensdk_env.yml #copy the environment.yml file into the root of the container
### this runs in the terminal after building the container
%post
    # Install Miniconda
    apt-get update
    apt-get install -y wget bzip2 ca-certificates python3 #added python3
    wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O /tmp/miniconda.sh
    bash /tmp/miniconda.sh -b -p /opt/conda
    rm /tmp/miniconda.sh
    export PATH=/opt/conda/bin:$PATH

    # Initialize Conda for all future shell sessions
    /opt/conda/bin/conda init
    cp /opt/conda/etc/profile.d/conda.sh /etc/profile.d/conda.sh #add to automatic running at runtime (?)
    ### runs "getting pip dependencies" forever
    # Install PIP
    #apt install -y python3-pip #attempt to solve "getting pip dependencies". does not work

    # Create environment from file
    conda env create -f /allensdk_env.yml
%environment
    export PATH="/opt/conda/bin:$PATH"
    . /opt/conda/etc/profile.d/conda.sh #initialize conda
    conda activate allensdk #activate environment
# metadata for what "apptainer run <name>.sif" will do (?)
%runscript
    . /opt/conda/etc/profile.d/conda.sh #initialize conda
    conda activate allensdk #activate environment
    exec "$@"
%labels
    Author YourName
    Version v1.0
```

Specify the OS you want to use

%files: puts files into the container for setup; add the .yml file to read later

%post: commands to run after the container is made

%runscript: default script to execute when “apptainer run” is called

# Running a .sif file

- apptainer run <imageName>.sif
  - launches container + runs the default script in the container
- apptainer exec <imageName>.sif <command>
  - launches container + runs specific command in container
- apptainer shell <imageName>.sif
  - launches container + opens interactive shell inside it

# Tags

- apptainer run -C -B .:/mnt <imageName>.sif
    - -C
        - Separate the container from the contents of your host computer
    - -B .:/mnt
        - Mount-bind everything in your present working directory to a “/mnt” directory in the container (does not need to exist)
        - Variable is <host directory>:<container directory>
- Mounting to the /mnt directory:
    - Make sure your code reflects the directories that you are mounting to
    - You will only be able to access the outputs that are stored to the /mnt directory in the container

# Demonstration

1. Log into compute canada
2. `module load apptainer`
3. `sbatch test_section.sh`