# ANACONDA ROUND TRIP

## Load the module

Before we start working with Anaconda, we need to reset all the loaded modules to avoid any conflicts. To do this, use the command:

module reset

After resetting, you can load the Anaconda module

module load Anaconda3

## Create an environment

Anaconda environments are used to isolate different projects and their dependencies. For example, we could create an environment with the SciPy stack for data analysis. To create an environment, use:

conda create -n myenv

Replace 'myenv' with the name of your environment.

## Activate the environment

To use an environment, you have to activate it. Activating an environment modifies your PATH so that the versions of software installed in the environment can be used:

conda activate myenv

The base environment is the environment where Anaconda is installed, and it includes a bunch of useful libraries. When you activate another environment, such as the one we just created, the software installed in the base environment is not available unless it is also installed in the new environment.

## Install packages

You can install additional packages in your environment with the conda install command:

conda install numpy

This command installs the NumPy library in the currently active environment.

## Use the packages

You can now use the packages installed in your environment. For example, if you installed NumPy, you could start Python and do:

import numpy as np  
np.array([1, 2, 3])

## Create an env file for sharing on Cheaha

If you want to share your environment with others, you can export it to a YAML file:

conda env export > environment.yml

This file includes all the packages in your environment and can be used to recreate the environment.

## Create an env file for broader collaboration

For broader collaboration, such as sharing your environment on GitHub, you might want to create a more curated environment file that only includes the main dependencies of your project, excluding transitive dependencies and packages installed for personal use. You can manually create such a file. Here's an example:

name: myenv  
channels:  
 - anaconda  
dependencies:  
 - numpy=1.21.5  
 - scipy=1.7.1

## Delete the environment

If you no longer need an environment, you can remove it:

conda env remove --name myenv

## Create the environment from the env file

You can create a new environment from an environment file with:

conda env create -f environment.yml

This creates a new environment with the same packages as the original environment. This is useful for sharing your work with others, as they can recreate your environment and run your code without any dependency issues.

# Running Jupyter Lab in a Browser using an SSH tunnel

First, open a terminal on your local machine.

## SSH into Cheaha:

ssh your\_username@cheaha.rc.uab.edu -p22

Replace your\_username with your actual Cheaha username. You will be prompted to enter your password. More information on seting up your Cheaha SSH session can be found here.

## Start a new tmux session:

tmux new -s jupyter

This starts a new tmux session named jupyter. You can replace jupyter with a different name if you prefer.

You can always SSH into Cheaha and check which tmux windows you have open by running tmux and pressing Ctrl+b followed by s. This will list the running windows.

## Run Jupyter Lab on a partition in Cheaha

To run Jupyter Lab on the short partition in Cheaha, you can use a combination of the srun command (which is part of SLURM) and the jupyter lab command. Here is an example:

First, you need to request a session using srun.

srun --partition=short --pty bash -l

This command requests an interactive session (--pty bash -l) on the short partition (--partition=short). You can choose any of the available partitions.

module reset  
module load Anaconda3  
conda create -n myenv  
jupyter lab --no-browser --ip=0.0.0.0

The --no-browser option prevents Jupyter Lab from trying to open a web browser, and the --ip=0.0.0.0 option allows connections from any IP address.

This will start Jupyter Lab on port 8888 and display a URL that you can use to connect to it. The URL will look something like this: **Copy this url into a text file for future use**

http://c0172:8888/?token=your\_token

Make a note of the partition number (c0172 in this case), port number (8888 in this case) and the token (your\_token).

Detach from the tmux session by pressing Ctrl+b followed by d. This will leave Jupyter Lab running in the background.

## Exit the SSH session:

exit

Now, you're back to your local machine's terminal.

## Set up the SSH tunnel

To set up an SSH tunnel to the Jupyter Lab, use the following:

ssh -N -L localhost:8000:c0172:8888 your\_username@cheaha.rc.uab.edu

In this command:

* ssh starts the SSH client.
* -N tells SSH that no command will be sent once the tunnel is up.
* -L specifies that the connections from the client should be forwarded to the server, then to the destination.
* localhost:8000 is the local port that you will use to access Jupyter Lab in your web browser.
* c0172:8888 is the remote port where Jupyter Lab is running on the server.
* your\_username@cheaha.rc.uab.edu should be replaced with your actual username and the address of the remote server.

After running this command, you should be able to access the Jupyter Lab by opening your web browser and going to http://localhost:8000/lab?token=YOUR TOKEN.

Keep in mind that YOUR TOKEN (dd616fc123c988812edbe1854ca884ebfb01e4fdb7693e7b) is used for authentication and it will change each time Jupyter Lab is started. You should always use the current token provided by the Jupyter Lab server.

Remember, you need to keep the SSH command running as long as you're using the Jupyter Lab session. If you close it, the tunnel will also close and you'll lose access to Jupyter Lab. Alternatively you can set up a Tmux session locally and run it in the background. Using the above commands.

If you encounter an error like:

bind [127.0.0.1]:8888: Address already in use  
channel\_setup\_fwd\_listener\_tcpip: cannot listen to port: 8888  
Could not request local forwarding.

The error message you're seeing indicates that the port 8888 on your local machine is already in use by another process. You can resolve this issue by choosing a different local port for the SSH tunnel.

For example, you could choose port 8889 (or any other available port) instead of 8888.