Finding the chameleon URL

- Go to https://www.chameleoncloud.org/
- Log-In
- Use the Menu at the top to select *Experiment -> Hardware Discovery*
- Use this page to explore various hardware resources available within chameleon. For instance, u can select various CPUs, RAM sizes , architectures and also, at times, Advanced filters. Once u selected filters click on view to view all resources which have that combination of hardware.
- Note, at this stage you can see not all sites will have all the hardwares. For instance, IB networks are only available at TACC.
- Once you identify the site, select the site from the top three buttons. Eg, CHI@TACC.
- Optionally, you can use *Experiment -> CHI@TACC*. (This works much faster in my computer)
- You can use this link directly now to access the cluster at that site and make reservations, allocate nodes,etc,.

Set Up SSH

- Now you need to setup Key pairs by adding your ssh keys.
    - Go to Compute -> Key Pairs
    - Here you can either create a new key-pair and use it to login or import an existing key-pair.
        - For a new Key-Pair (recommended)
            - Click on Create Key-Pair
            - Assign a unique name
            - Select SSH key in key pair
            - Download the key-pair
            - Store this key in your computer, usually under `~/.ssh`
            - Change permissions of the key `chmod 400 ~/.ssh/name.pem`
            - Optional: Set up a config file. (Note that you don't have an ip yet)

              ```
              Host name1
              Hostname ip
              User cc
              PubKeyAuthentication yes
              IdentityFile ~/.ssh/name.pem
              ```

            - Once done, you can connect by using `ssh name1`
        - For importing
            - Generate a key with ssh-keygen
            - Click on import public key
            - Assign a unique name

- Either load your .pub file or copy paste the key.

Create a lease

- navigate to *Reservations* -> Leases
- Click on Create lease.
    - Fill in all the information based on the hardware discovery you did in the website
    - As a good practice, always have one node as a head node which can hosts NFS, assign public IP etc.
    - Click on create
- Wait for lease to become ACTIVE (on the status column)
- NOTE, the lease duration is only for 7 days. If you want more, on the 5th day you can go to lease and extend it. If no one else is using those nodes.

Create an instance

- Go to *Compute -> Instances*
- Click on *launch Instances*
- Assign a unique name, count and your reservation (if u assign count > 1 then this name is appended with the count starting from 0)
- Then select Source tab and select the image. Generally, we start with a base IMAGE of an OS like UBUNTU, CentOS, etc,. Unless, you have an IMAGE created that you want to use (we will see how to do this later).
- Select Flavor, as Bare Metal
- Go to key-pair tab and ensure your key-pair is selected.
- Default Security group is more than enough, if you need any change to this let us know.
- Once done, click on Launch Instance
- It might take several minutes to launch the instance. When it is done the Status becomes Active and Power State to be Running.

Assign floating IP to head node

- Go to *Network -> Floating IPs*
- Click *Allocate IP to Project*
- Give it a name and create it.
- Click Associate on your newly created IP
    - Here select your head node from *Port to be associated.*
    - Once Status changes to ACTIVE, your floating IP is ready to use.
- Ideally, associate a public IP only to your head node, once inside you can connect to other nodes using ssh internally.

Using the instances

- Connect to login node using ssh client.
- You now have full access to a Linux Machine
- You have “full” sudo access.

Create a Ticket

- Go to https://www.chameleoncloud.org/hardware/
- User Icon (top right) -> Dashboard
- Click on *Open a Ticket*
- Provide complete information of the problem, which instances from the overview of the instance, etc.
- Create

Install MPI

- Go to https://www.mpich.org/downloads/. MPIch is the implementation of MPI that we will be using. We are going to compile from source.
- We need to download the file from http://www.mpich.org/static/downloads/3.3.2/mpich-3.3.2.tar.gz (newer versions might need a better link).
- Use `wget http://www.mpich.org/static/doswnloads/3.3.2/mpich-3.3.2.tar.gz`
- It is my recommendation that you create a folder name software and two subfolders installs and tarballs. Put the downloaded file under tarballs and decompress it using `tar zxf mpich-3.3.2.tar.gz`
- We now need to configure it, build it, install it and add it to the path.
    - Configure:
        - `cd /home/cc/software/tarball/mpich-3.3.2`
        - `./configure -prefix=/home/cc/software/install |& tee c.txt`
    - Build
        - `make 2>&1 | tee m.txt`
    - Install
        - `make install |& tee mi.txt`
    - Add the new path to your .bashrc. Use vim ~/.bashrc and add the following line:
        - `export PATH=/home/cc/software/install/bin:$PATH`

    - If you install everything in /software install, then all your libraries are going to be in the same `~/install/bin directory`

Snapshot

- If you install anything on the system (outside of workspace) then you need to update the image
    - To update the image
        - Run `cc-snapshot <UNIQUE_IMAGE_NAME>` utility preinstalled in the system
- Takes a while and requires sudo usage and to indicate your chameleon username and password.
- Once created, they can be used to spawn new instances at the same point of development by going to *compute->images* and hitting launch or through the normal steps but selecting the snapshot when selecting the source.
- Very useful once software has been installed, so we can avoid installing mpi multiple times.

Passwordless SSH

- Additionally, you are going to have to establish passwordless-ssh with all the other nodes in the cluster (a requirement of MPI). Here are the steps:
    - Download the script `git clone https://github.com/JaimeCernuda/sshSyncScript.git`
    - Edit the file in */etc/hosts*. You need sudo.
        - In this file add the <IP> <HOSTNAME> for each instance you want to sync. One instance per line.
            - The local IPs of all the instances can be viewed from the website and the name is up to you. Ex:
            - `192.168.0.125 master`
            - `192.168.0.241 slave1`
    - Move the keys used to log into chameleon into our master node, to do so use `scp ~/.ssh/name.pem cc2020:~/.ssh/id_rsa`
    - Run ./sync.sh
    - This script should now show all the hostname you had added.
    - Once you press enter, it will synchronize all instances so that they can access each other without password.

Install NFS

- Server.

    - Install the server: `sudo apt install nfs-kernel-server`
    - When configuring an NFSv4 server it is a good practice is to use a global NFS root directory and bind mount the actual directories to the share mount point
    - Create the root directory: `mkdir -p /export/cc`
    - Mount /home/cc into the root folder: `sudo mount --bind /home/cc /export/cc`
    - Ensure that it happens in runtime by adding to `sudo vim /etc/fstab` the following line:
        - `/home/users /export/users none bind 0 0`
    - Put the file system on the network by adding to `sudo vim /etc/exports` the following two lines:
        - `/export 192.168.0.0/24(rw,fsid=0,insecure,no_subtree_check,async)`
        - `/export/cc 192.168.0.0/24(rw,nohide,insecure,no_subtree_check,async)`
    - `Run: sudo exportfs -ra`
    - Restart the service to apply changes: `sudo service nfs-kernel-server restart`
- Client
    - Install the client sudo `apt-get install nfs-common`
    - Mount the network device into our home: `mount -t nfs -o proto=tcp,port=2049 <nfs-server-IP>:/cc /home/cc`
    - Ensure that it happens in runtime by adding to `sudo vim /etc/fstab` the following line
        - `<nfs-server-IP>:/cc  /home/cc  nfs  auto  0  0`
    - If after mounting, the entry in `/proc/mounts` appears as `<nfs-server-IP>://` (with two slashes), then you might need to specify two slashes in `/etc/fstab`, or else umount might complain that it cannot find the mount.