Overview:
Avogadro/Gaussian - creates geometry of single molecule
CHARMM-GUI: parameterizes single molecule - https://www.charmm-gui.org/?doc=sign
Packmol - packs simulation cell with copies of single molecule - https://m3g.github.io/packmol/
VMD - parameterizes simulation cell - https://www.ks.uiuc.edu/Research/vmd/
OpenMM - performs system energy minimization, equilibration, and simulation - https://openmm.org/
MDAnalysis - for your trajectory analysis https://userguide.mdanalysis.org/stable/index.html

Commonly used commands:

1. ssh <username>@<hostname>
2. qsub -I -l walltime=1:00:00 -l nodes=1:hima:ppn=32

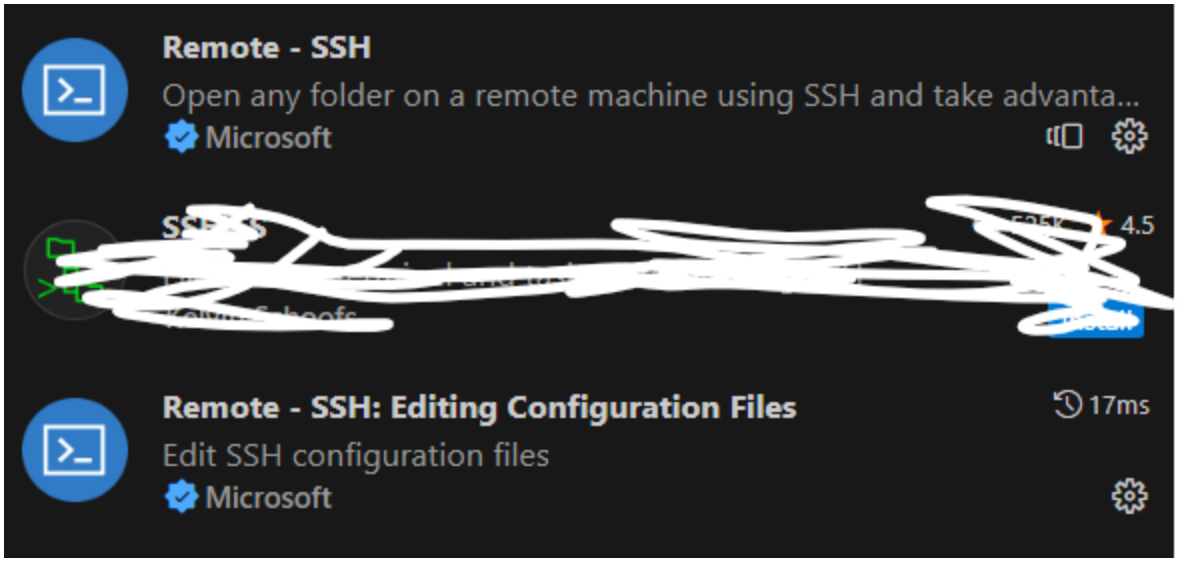
Step 1: Getting an HPC account.
https://www.wm.edu/offices/it/services/researchcomputing/acctreq/. This link should take you to the William & Mary page about requesting an account. Follow the link to request an account and fill out the form. It should take a few days for your account to get granted.

Step 3: VSCode
https://code.visualstudio.com/download. Go here and follow the instructions to download VSCode. Strongly recommend VSCode

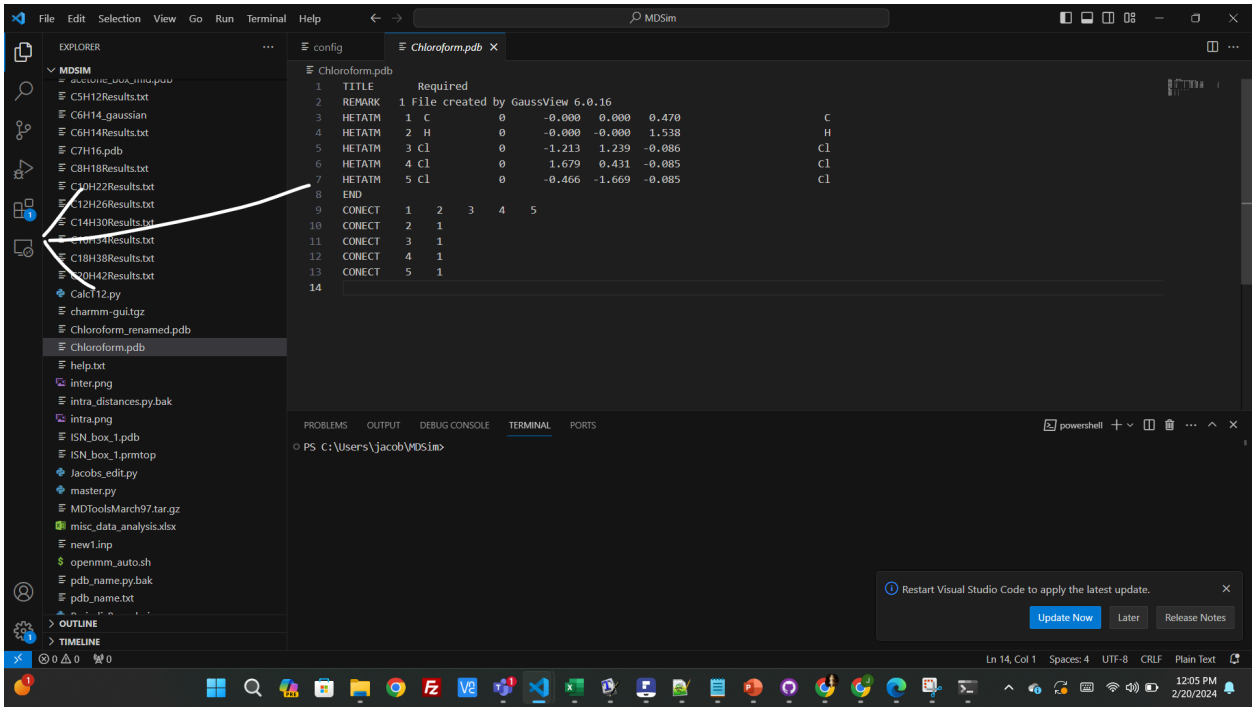
Using SSH:
https://www.wm.edu/offices/it/services/researchcomputing/using/connecting/linux/

VSCODE:
I recommend using VSCode (if you are windows) to ssh. It's very nice to have a file explorer, text editor, and terminal all available to me in the same window. There are probably other programs that have similar features, but you will have to look into that on your own (if you care). Here are the instructions for VSCode

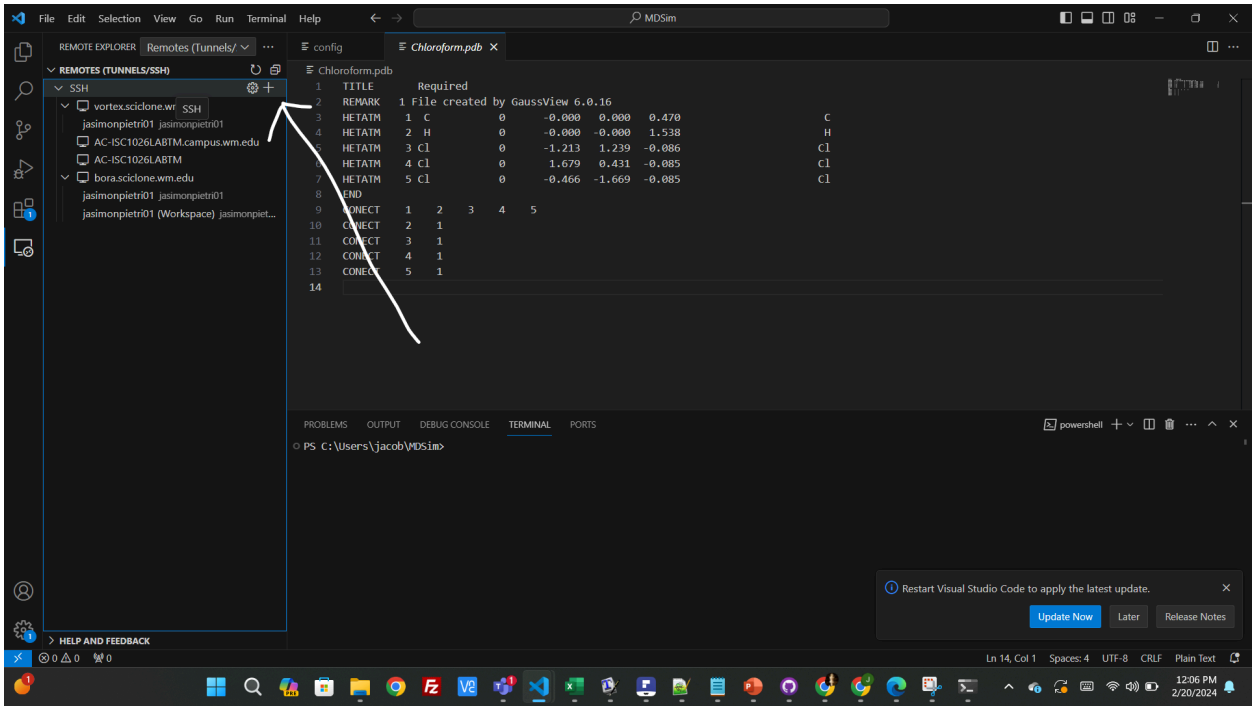
1. Install https://code.visualstudio.com/download
2. Open, and go to extensions. Type "ssh" into the search bar. Install the two extensions in the picture below.



3. Click on the monitor icon on the left.



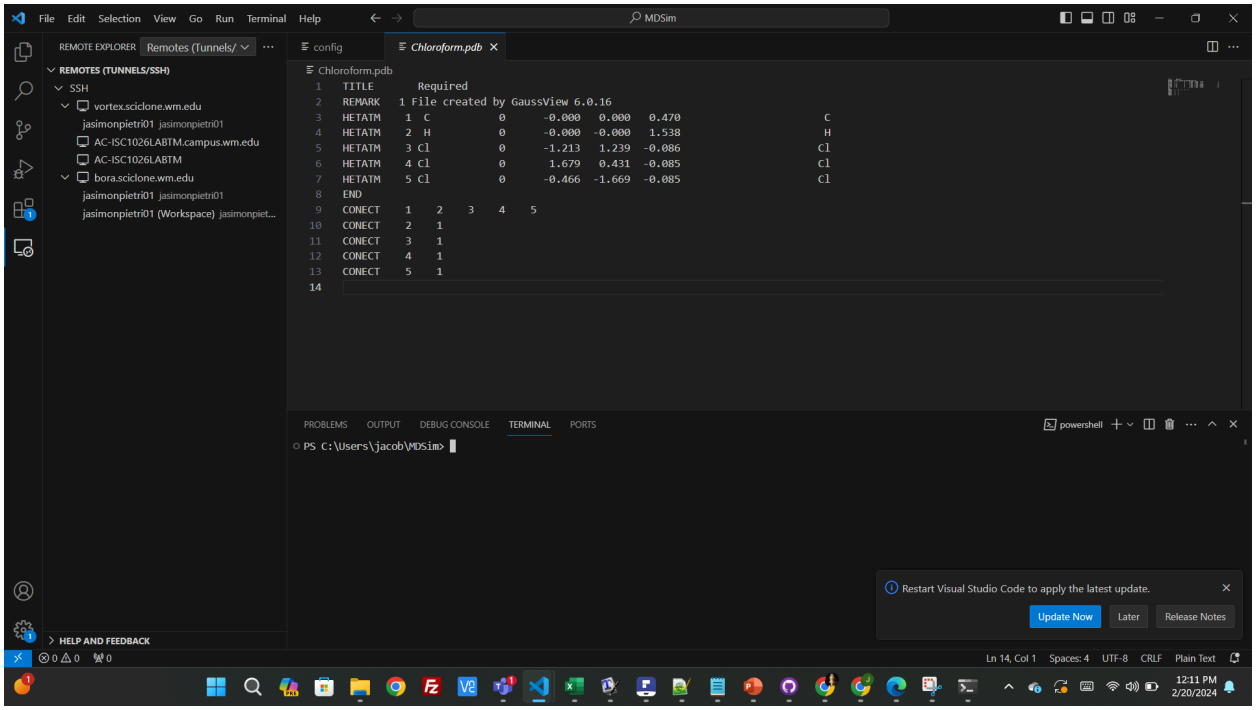
4. Click on this plus



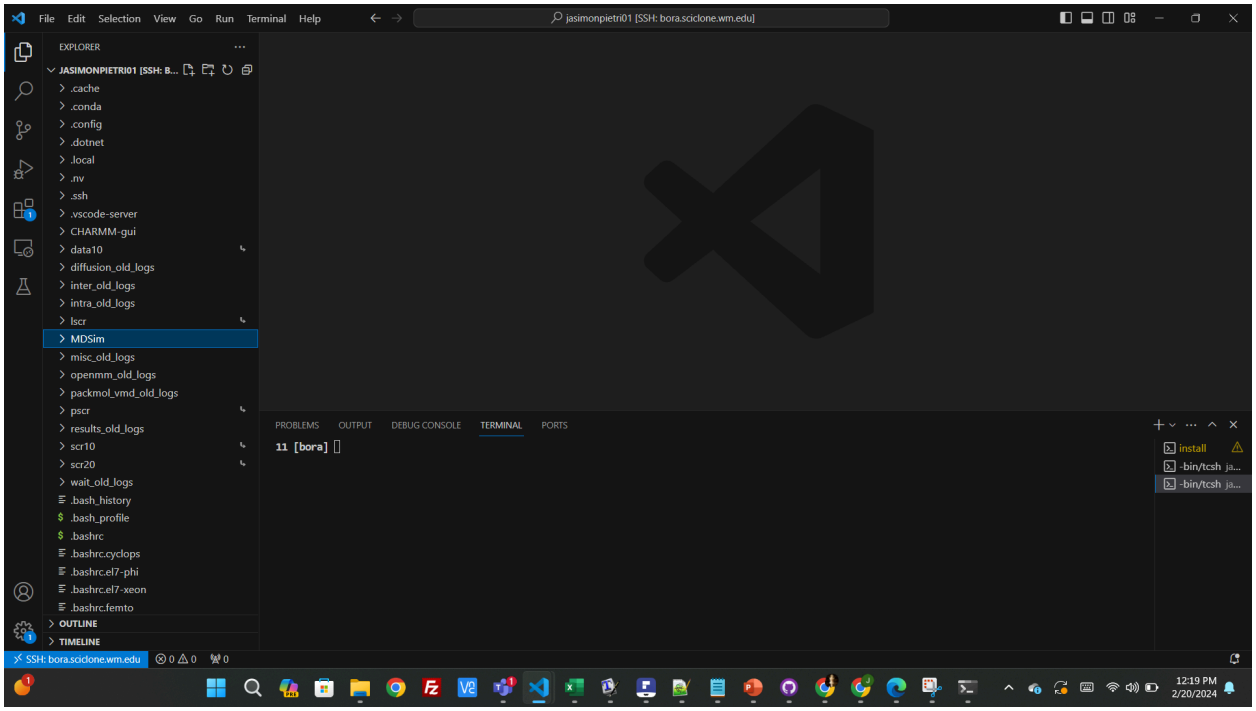
5. Type this command in the command line at the top center of the screen.
    a. "ssh <username>@<hostname>" - FOR ON CAMPUS
        i. Or "ssh -J <username>@bastion.wm.edu <username>@hostname" for off campus
        ii. NOTE: when switching between off campus and on campus. If you click on the gear by the SSH and edit your ssh config file, you should see something like this. Simply add this ProxyJump line if it isn't there. This can be commented out when you are on campus and commented in when you are off.
        iii.

```
Host bora.sciclone.wm.edu
  HostName bora.sciclone.wm.edu
  #ProxyJump jasimonpietri01@bastion.wm.edu
  User jasimonpietri01
```

    b. Username = your W&M username
    c. Hostname = bora.sciclone.wm.edu
        i. You can use other subclusters on the HPC but get started with this one.
6. It will ask you to update a config file. Just select the top entry in the list.
7. You should see bora.sciclone.wm.edu and your username popup under ssh. On the left. You might need to refresh/click on the ssh bar thing. Hover over your username and a right arrow will appear. Click on that arrow and you will connect.



8. It will ask if you want to continue. YES
9. Enter your password. Hopefully you connect and see something like this



Getting Started Guide:

Step 1, loading modules on startup:

I use both the Bora/Hima and Vortex subclusters.
Bora and Vortex are both used for the parallelization of the autocorrelation calculation
Bora is typically much busier than vortex, but the nodes have more memory, which is desirable for the ACF calculations. Specifically, more memory will support more processes per node.
Vortex is less busy than bora, but inter jobs are limited to ppn=5
Every other job type (besides inter and intra) is run on hima, as they are almost always free.
hi07 is crucial to this workflow since its GPU capabilities have drastically decreased the simulation (OpenMM) time:
For Bora/Hima:

1. open the startup file, ~/.cshrc.el7-xeon.
    a. With VSCode, just scroll through the file explorer on the left to find it and click it to open it.
2. comment out (or delete) the following line "module load intel/2017 openmpi/2.1.1/intel"
3. add in the following line "module load anaconda3/2021.05 intel/2019 intel/2019-mpi cuda/11.7"
    a. It should look like this.

```
.cshrc.el7-xeon
#
#  Environment configuration for EL 7.x / Xeon environment.
#
#  Revised:
#    12/06/2016 lsh       - Adapt from RHEL 6.x / Xeon.
#


#
#  Personal module configuration:
#    - Defaults to a 64-bit environment optimized for Intel Haswell-EP
#      Xeon processors.
#    - ISA module must be loaded before other modules.
#    - Available ISA's include:
#         haswell       (breeze: s16)
#         broadwell     (Bora: s18b, c21; Hima: c22; mistral: s19)
#
module load isa/broadwell
#module load intel/2017 openmpi/2.1.1/intel
module load anaconda3/2021.05 intel/2019 intel/2019-mpi cuda/11.7
```

We need these modules for all our code to work; adding this line to this file tells the Bora/Hima nodes to load these modules in on startup
For Vortex:
1. open the startup file, .cshrc.vortex

2. add in the following line "module load anaconda3/2021.05 intel/2019 intel/2019-mpi"

Step 2, creating a conda environment:
You are going to need three environments if you want to use the vortex nodes, two otherwise. For starters just make the two for bora/hima
You could set this up however you want, but the problem is that the modules MDAnalysis and VMD are mutually exclusive. Not sure what the problem is, just that there is a problem
It is recommended to have one environment, called mda, that has MDAnalysis.
The second environment, called MDenv_main, can have everything else installed
1. request an interactive job with the command "qsub -I -l walltime=1:00:00 -l nodes=1:hima:ppn=32". This command requests all processors of one hima node for one hour. Nothing that is remotely computationally intensive should be run on the starting node. If you do, it will get killed. This node is used by all other users to operate the terminal.
2. create the first conda environment with the command "conda create -n mda"
3. activate the environment using "conda activate mda"
4. run these commands "conda config –add channels conda-forge" and "conda config –set channel_priority strict"
5. install mdanalysis "conda install -c mdanalysis"
6. pip install "pip install pdb-tools mpi4py"
7. repeat steps 2-4 with a new conda environment, called MDenv_main
8. package install "conda install -c openmm openmm-setup openff-toolkit packmol vmd"
9. pip install "pip install pdb-tools"

Here is a basic summary of what all the packages do. Feel free to google anything for more information:
MDAnalysis - used for trajectory analysis. We use it to easily select and get the coordinates of the hydrogens in the acf calculations
mpi4py - used to parallelize the acf calculation
openmm - python simulation module. Runs the simulations
packmol - packs the simulation cell
vmd - simulation visualizer. We use it to watch the simulations to make sure they look correct, as well as to convert pdbs to psfs
pdb_tools - "swiss army knife for pdbs" used to edit pdbs output by gaussian for use with packmol

Part 1, Bulk simulation - Remember, nothing intensive on the startup node. Everything below should be done on a Hima node requested with the command found above. Also, everything will need to be done in the MDenv_main conda environment. For now, you can run OpenMM this way as well. You should drastically shorten the number of steps to shorten the simulation length (we don't care about the output of these simulations, just that they are outputting things). Eventually, you will have to request a specific node with GPU acceleration.

1. Transfer input PDB and forcefield files: Copy the pdb and str (or rtf and prm if still separte) into your folder in the HPC.

a. Just open a new instance of VSCode and find them with the file explorer. Copy them and paste them into the VSCode which is connected to the HPC.

2. Take one of the provided input files and pack it into simulation cell using Packmol
    a. See the example .inp file. Look at packmol user guide for instructions to run packmol.
3. Take the pdb output by Packmol and turn it into a PSF using VMD. You will need to supply the toppar (topology, parameters) files containing the topology for your molecule. Here is a list of all the molecules I have and their corresponding toppar (AKA force field) files. The str and rtf contain the topology, and the str and prm contain the parameters. Note: the CGenFF files aren't needed for water. They are needed for a graphene surface which I was working on.

```
def get_CHARMM_files(mol):
    ref_mols = {'C5H12': 'cgenff','C6H14': 'cgenff','C7H16': 'lipid','C8H18': 'cgenff','C10H22': 'lipid','C12H26': 'cgenff',
                'C14H30': 'lipid','C16H34': 'lipid','C18H38': 'cgenff','C20H42': 'cgenff','acetone': 'cgenff','ethanol': 'cgenff',
                'toluene': 'cgenff', '1-2-diaminopropane': 'cgenff', 'DGEBA': 'cgenff', 'ethylene_glycol': 'cgenff', 'glycerol': 'cgenff',
                'isopropanol': 'cgenff', 'aluminum': 'interface', 'water': 'water'}
    for key in ref_mols:
        if key == mol:
            if ref_mols[key] == 'cgenff':
                return '\nstr = MDSim/toppar/'+mol+'.str\nrtf = MDSim/toppar/top_all36_cgenff.rtf\nprm = MDSim/toppar/par_all36_cgenff.prm\n'
            elif ref_mols[key] == 'lipid':
                return '\nstr = MDSim/toppar/toppar_all36_lipid_model.str\nrtf = MDSim/toppar/top_all36_lipid.rtf\nprm = MDSim/toppar/par_all36_lipid.prm\n'
            elif ref_mols[key] == 'interface':
                return '\nstr = MDSim/toppar/empty.str\nrtf = MDSim/toppar/top_interface.rtf\nprm = MDSim/toppar/par_interface.prm\n'
            elif ref_mols[key] == 'water':
                return '\nstr = MDSim/toppar/toppar_water_ions.str\nrtf = MDSim/toppar/top_all36_cgenff.rtf\nprm = MDSim/toppar/par_all36_cgenff.prm\n'
```

    a. https://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-html/node6.html
    b. See the example .vmd file. See link for running command. I recommend replacing -e with -startup. -e tells VMD to run this file and then run normally. -startup tells VMD to run this file only.
    c. Use'quit' command to exit VMD
2. Load in the correct forcefield files (usually a .rtf, .prm, and .str), pdb and psf into OpenMM to perform the simulation. Look at the
    a. For Tyani - Feel free to adjust any and all parameters. Basically all of my parameters come from Singer (2017) "Molecular dynamics simulations of NMR relaxation and diffusion of bulk hydrocarbons and water." You should look for papers detailing the process of calculating IR frequencies from simulations - it could be that different Integrators or parameters are required for those calculations.
    b. For Shalom - Just use the parameters from Singer 2017

Part 2, Interface simulation

1. Basically the same process. You can either do the same process for both molecules and combine the PDB (using MDAnalysis) and PSF (using VMD) at the end, by using packmol to pack adjacent boxes and loading both topologies into VMD with two different segments. Feel free to pick any two molecules to create the interface.
    a. Tyani - I recommended water and hexane
    b. Shalom - Use aluminum pdbs and psfs in the Github and any molecule.

Part 3, new molecules

1. See Adding_a_new_molecule for more indepth information on how to add new molecules.
2. Perform either an interface or bulk simulation to ensure you did everything correctly

Once all this gets done, we can talk about automation. Basically, I can take any single molecule PDB and perform every step after by running one script. I haven't built this to work well with new types of simulations - it's only good at the simulations I need.