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

Using SSH:
https://www.wm.edu/offices/it/services/researchcomputing/using/connecting/linux/
Command: ssh <username>@<hostname>
Comments: I recommend using hostname = bora.sciclone.wm.edu. Username is just your W&M username. You should see “[bora]” in the command line.

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 (Tyani I can also help with your trajectory analysis. From looking into it very little, you will need to perform the autocorrelation of the energy of interacting electric dipoles to calculate the IR stretches, whereas I need to calculate the ACF of the energy of interacting magnetic dipoles)
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.

    Command: nano ~/.cshrc.el7-xeon - nano is a command line text editor

2. comment out 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"

    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"
Hopefully this will work perfectly. I haven’t actually tested this myself.
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. 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.
   https://www.wm.edu/offices/it/services/researchcomputing/using/xfers/sftp/
   Command: “scp filename <username>@<hostname>”
   “$ scp -r ~/local_dir user@host.com:/var/www/html/target_dir” use -r for all files in a directory
2. Take one of the provided input files (I will suggest water) and pack it into simulation cell using Packmol
3. Take the pdb output by Packmol and turn it into a PSF using VMD

    a. https://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-html/node6.html
2. Load in the correct forcefield files (usually a .rtf, .prm, and .str), pdb and psf into OpenMM to perform the simulation
    a. 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.

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.

Part 3, new molecules

1. Use Gaussian or Avogadro to output a PDB.
2. These PDBs can be submitted straight to CHARMM-GUI Ligand Reader & Modeler
    a. Look out for errors. For example, we found out the hard way that CHARMM doesn’t support chloroform
3. The Program searches through its files for the molecule’s topology there are three possible end cases
    a. The molecule's topology is located in one of the six force field topology (RTF) files. In this case, the molecule's topology is fully specified by this file.
    b. The molecule's topology is located in one of the additional stream (STR) files. In this case, the molecule's topology is fully specified by this file and the corresponding force field topology file. The correct file is usually clear from the file naming conventions (the file toppar_all36_lipid_model.str, which contains n-decane, relies on top_all36_lipid.rtf). If this is not the case, The correct topology file can be identified by shared atom names.
    c. The molecule's topology file is not found in the CHARMM36 files
4. If the molecule's topology is found, the program outputs the file that contains it and the molecule's residue name. If the molecule is not found, the program will ask for a residue name and generate topology and parameter files for the molecule (I find it easiest to combine these into a .str file). In both cases, a new PDB is output in which the atom names and atom order match the topology file. The Ligand Reader & Modeler makes two mistakes when generating this PDB. Firstly, if the molecule is found in the C36 files, it will have the incorrect residue name. Secondly, atom lines will begin with 'ATOM' instead of 'HETATM'; 'HETATM' should be used for small molecules. These problems can be easily fixed by the user.
5. Perform either an interface or bulk simulation to ensure your PDB is valid