ComCat Lab Welcome Guide

Welcome to the group! This guide introduces several of the key tools and services that are used in ComCat Lab and examples of their use.

You can view the complete guide in PDF form (`comcat-lab-welcome-guide.pdf`), or, alternatively, you can view the guide as a static webpage.

The best way to learn is by doing! Start by picking a tutorial and trying to work through it. The tutorials are (relatively) well documented, though, they are not self-contained. While you're getting acquainted with the material and workflows, feel free to bounce back and forth between the tutorials and the software pages. For more in-depth treatment of the theory, you are encouraged to consult the "Fundamentals" collection in ComCat Lab's Zotero collection. There you'll find annotated original references, more extensive DFT guides, and relevant textbooks. Regarding software, the documentation pages are generally extensive. Of course, if all else fails, don't hesitate to reach out to a group member!

Highlights

This guide has several valuable resources to help you get up to speed:

- **Workflows**: an overview of frequent workflows used with links to relevant software pages
- **Software Pages**: introductions to the various tools used in ComCat Lab
- **Tutorials**: walkthroughs for several common workflows/softwares
- **Sample Scripts**: example Python and SLURM scripts
- **Troubleshooting**: common issues and their resolutions/work-arounds

Prerequisites

- **Set up your local machine**: this guide assumes that your local machine is setup up with the necessary software already installed (e.g., Hatch, Python, etc.); follow the steps outlined in local-setup to satisfy this prerequisite
- **Set up your cluster account**: some of the tutorials in the guide require that you connect to the remote clusters provided by the Alliance. For these tutorials, you will need a valid CCCDB account and to set up your remote cluster environment

Quickstart

These instructions assume that you have already completed setup of your SFU workstation. If you don't have have access to a configured workstation, please

out to a group member to help you set that up, or, if you're feeling up for it, check out the local setup repository that will walk you through a comprehensive setup!

1. Create a virtual environment in the top-level directory of the Welcome Guide folder (in the same directory as this README).

   ```
   python3 -m venv .venv
   ```

2. Activate your environment.

   ```
   source .venv/bin/activate
   ```

3. Install the required Python dependencies.

   ```
   python3 -m pip install .
   ```

4. Pick a tutorial!

Workflows

It takes several steps to go from a structure to a publishable result. This page contextualizes common steps and highlights the infrastructure required to execute those steps.

1. **Obtain a structure file.**

   Structure files can be:

   - generated within various Python packages (e.g., ASE, Pymatgen)
   - obtained from the supplementary information of publications
   - retrieved from databases (e.g., Materials Project)

   Common structure file formats:

   - `.traj`: ASE trajectory file
   - `.cif`: Crystallographic Information File
   - `.xyz`: XYZ File Format
   - software-specific formats (e.g., `CONTCAR`, `Gaussian.log`, etc.)

2. **Perform structure manipulations.**

   Depending on the project, this can any combination of the following:

   - modifying the structure (e.g., creating defects, substituting atoms, creating surfaces)
   - adding adsorbates
   - and more...

3. **Determine necessary calculations and parameters.**

   This decision is informed by literature, experience, and the scientific questions that one wants to answer.

4. **Create input files.**

   The required input files for a calculation generally include:

   - a structure file
   - a parameter file for the DFT code
   - a Python script for interfacing with DFT code or generating results
   - a submission script
   - additional files for the DFT code (e.g., pseudopotentials, wavefunction files, etc.)

   We have template Python and submission scripts for several different job types. The format for parameter files and additional files required by DFT codes are generally specified in the documentation for the corresponding computational code. However, for many DFT codes, there are utilities that can help with the the creation of files b and e:

   - ASE: of provides Python interface to input file generation for a number of computational chemistry codes
   - MaterialsCloud: provides a graphical interface for generating QuantumEspresso input files
   - VSCode VASP INCAR Pilot Extension: VS Code extension providing help and support for writing VASP `INCAR`, `CONTCAR`, and `POSCAR` files

5. **Submit calculations.**

   Calculations are submitted to remote clusters managed by the Digital Research Alliance (formerly Compute Canada). This is typically done by logging into one of the clusters using `ssh` and executing something like

   ```
   sbatch run.sh
   ```

   from the command line.

6. **Retrieve Results.**

   Although the output files produced by the job contain the results of the calculation, these files are often quite large (~GB), so it is often useful to extract only the specific required outputs. These outputs may include:

   - the final positions of and forces on each atom in the final structure
   - the final energy for the system
   - orbital occupancies
   - trajectories for an optimization
   - vibrational modes

   While you can write a script to extract these data from the output files of your job, there is almost definitely a routine to extract this data in either ASE or Pymatgen, so search the documentation of existing software for your use case.

7. **Analyze Results.**

This almost inevitably involves importing data into Excel and performing further analyses there. For some data (e.g., density of states, valence charge density difference diagram, molecular orbital visualization), you may need another software. In such a case, check the tutorials for your use case.

Software Pages

Digital Research Alliance (formerly Compute Canada)

Documentation

This is the national organization that manages our main computational resources. Their documentation is comprehensive, including details about specific clusters, available software, and how to get started. To use these resources, you'll need to create a CCCDB account.

Related:

- Creating a CCCDB account
- Transferring data to remote clusters
- Setting up your cluster environment

Slurm Workload Manager

Documentation

Slurm is the job scheduler that runs on our remote clusters. You can find a useful cheatsheet here. `sbatch`, `squeue`, and `sacct` are commonly used for submitting and monitoring jobs.

Python

Documentation

Python is a relevant programming language in which many useful utilities for our computational workflows are written. Python files can be identified by the `.py` extension.

bash

Documentation

bash is a shell program and shell-scripting language. Notably, it is the default shell on most compute clusters. Navigating and interacting with the file systems on remote clusters is done using bash. For example, the command-line functions `ssh` and `scp` are used to connect to and transfer files between the Alliance clusters.

Atomic Simulation Environment (ASE)

Documentation

The Atomic Simulation Environment (ASE) is a set of tools and Python modules for setting up, manipulating, running, visualizing and analyzing atomistic simulations.

Pymatgen

Documentation

Pymatgen (Python Materials Genomics) is an open-source Python library for materials analysis.

Vienna Ab Initio Simulation Package (VASP)

Documentation

VASP is a proprietary, plane-wave-based code that we use for studying periodic systems.

Quantum Espresso

Documentation

Quantum Espresso is an open-source plane-wave-based code that we use for studying periodic systems.

Gaussian

Documentation

Gaussian is a proprietary, gaussian-orbital based code that we mainly use for studying molecular systems.

ORCA

Documentation

ORCA is a free-for-academic gaussian-orbital based code that we mainly use for studying molecular systems.

Markdown

Documentation

Markdown is a markup language commonly used for writing documentation. A common extension for Markdown files is `.md`. The documentation for this Welcome Guide is written in Markdown. Markdown files can easily be converted into the pages of a static website using a tool like mkdocs.

Lmod

Documentation

Lmod is a software for managing software environments. The necessary commands and variables required for a software to be used are specified in 'module files' that are written in the Lua programming language. Modules are managed using commands like `module load`, `module purge`, and `module unload`.

Git

Documentation

Git is a version control system (VCS) that allows you to keep track of multiple versions of files at once. With git it's easy to revert back to an old version of a file, merge different versions of files. This is very helpful when you're trying to determine what change resulted in a particular error.

WIP

- Materials Cloud
- Materials Project
- Globus
- FileZilla
- Cyberduck
- bader
- Multiwfn
- Lobster
- chargemol
- Fireworks
- jobflow
- Custodian
- atomate2
- catmap
- matplotlib
- ccu (e.g., FancyPlots)
- autojob

`.bashrc`

A standard bash profile for use on remote clusters defining utility functions and environment variables for software.

`py title="samples/bash/.bashrc" --8<-- "./samples/bash/.bashrc"`

!!! important "Reminder" Replace `username` and `USERNAME` with your username.

Samples

Here, you can find sample files for:

- Python input files
- bash profile files
- Slurm input files
- Software-specific files (e.g., VASP, Quantum Espresso, etc.)

Python Scripts

This page describes the collection of Python sample scripts contained in the Python sample scripts folder.

Run a VASP relaxation using the internal quasi-Newton optimization algorithm (RMM-DIIS) in VASP

```
py title="samples/python/run.py" --8<-- "./samples/python/run.py"
```

!!! important “Reminder” Replace `"in.traj"` with the name of your structure file.

Run a VASP relaxation using the BFGSLineSearch optimization routine in ASE

```
py title="samples/python/run_ase.py" --8<-- "./samples/python/run_ase.py"
```

Run a VASP relaxation using `ccu`

```
py title="samples/python/run_ccu.py" --8<-- "./samples/python/run_ccu.py"
```

`ccu` is a set of tools for computational chemistry workflows. In particular, `run_relaxation` is a wrapper function around `ase.atoms.Atoms.get_potential_energy()` that handles the logging and archiving of a calculation's final results.

!!! important “Reminder” Replace `"in.traj"` with the name of your structure file.

Slurm Submission Files (WIP)

Submit a DFT calculation

=== “VASP”

```
``` py title="samples/slurm/vasp.sh"
--8<-- "./samples/slurm/vasp.sh"
```
```

=== “Espresso”

```
``` py title="samples/slurm/espresso.sh"
--8<-- "./samples/slurm/espresso.sh"
```
```

=== “Gaussian”

```
``` py title="samples/slurm/gaussian.sh"
--8<-- "./samples/slurm/gaussian.sh"
```
```

=== “ORCA”

```
``` py title="samples/slurm/orca.sh"
--8<-- "./samples/slurm/orca.sh"
```
```

!!! Reminder

```
Don't forget to replace `JOB_NAME`, `SFU_ID`, and `PYTHON_SCRIPT` with
appropriate values in addition to setting your desired SLURM parameters.
```

Creating a CCCDB Account (WIP)

You should receive an email invitation.

Transferring Data to a Remote Cluster (WIP)

Using `scp`

Using `rsync`

Using Globus

Using another option

viz. FileZilla or CyberDuck.

Tutorials (WIP)

This page explains how to get started with the tutorials provided in this guide and highlights many useful sources for additional tutorials.

Getting Started

1. Download the linked Jupyter Notebook.
2. Follow along!

!!! note

`You must have Conda Package Manager installed to proceed with the tutorials in this guide.`

Additional Tutorials

ASE Tutorials: how to do basic property calculations, adsorption studies, slab generation, and more...

matgenb: a collection of Jupyter notebooks for Pymatgen, Custodian, and Fireworks compiled by the creators of Materials Virtual Lab, Materials Project

ORCA Tutorials: walkthroughs of how to perform various calculations; part of the official ORCA documentation

Modeling materials using density functional theory: "The Kitchin book"; a textbook with examples of how to use ASE to perform various DFT calculations with VASP; also contains explanations of the underlying theory with references; a PDF version can be found here

Open Catalyst Intro Series: A YouTube playlist from the Open Catalyst Project motivation atomistic simulation

QE-2019: course materials from the 2019 Summer School on Advanced Materials and Molecular Modeling; contains background theory and examples with Quantum Espresso

Online Course from Ghent University: a free, online course in Computational Materials Physics offered by Ghent University; examples in Quantum Espresso

Hands-On Quantum Espresso: a collection of tutorials for installing Quantum Espresso and using the SCF (`PWscf`) and phonon (`PHonon`) modules in Quantum Espresso with theoretical background

Atomistic Computer Modeling Of Materials (SMA 5107): lecture notes, videos, and problem sets from the Ceder/Marzari graduate class taught at MIT

QE YouTube Tutorials: a YouTube playlist with beginner tutorials for Quantum Espresso

Troubleshooting

This page contains potential solutions to whatever may currently be troubling you. Your mileage may vary.

VASP

VASP internal routines have requested a change of the k-point set

Context Running phonon calculations for nitrate reduction intermediates on M-BHT slabs.

Environment

- **Cluster**: Cedar
- **Date**: July 9, 2024
- **Software Versions**: Python 3.11.9, ASE 3.23.0, VASP 5.4.4

Relevant Files Input files for the calculation.

=== “INCAR”

```text
--8<-- "./docs/sources/resources/files/kpt_set_change_error/INCAR"
```

=== “run.py”

```py
--8<-- "./docs/sources/resources/files/kpt_set_change_error/run.py"
```

=== “vasp.sh”

```py
--8<-- "./docs/sources/resources/files/kpt_set_change_error/vasp.sh"
```

The Error

“text title=“vasp.out”

|
EEEEEEE RRRRRR RRRRRR OOOOOOO RRRRRR ### ### ### |
E R R R R O O R R ### ### ### |
E R R R R O O R R ### ### ### |
EEEEE RRRRRR RRRRRR O O RRRRRR # # # |
E R R R R O O R R |
E R R R R O O R R ### ### ### |
EEEEEEE R R R R OOOOOOO R R ### ### ### |
|
VASP internal routines have requested a change of the k-point set. |
Unfortunately this is only possible if NPAR=number of nodes. |
Please remove the tag NPAR from the INCAR file and restart the |
calculations. |

“text title=“vasp.out”

|
—-> I REFUSE TO CONTINUE WITH THIS SICK JOB . . . , BYE!!! <—- |
|

“‘

This is an error due to symmetry breaking (see here).

The Solution Set `ISYM=0`. in your `INCAR` file.

Useful Links

This page is a catch-all for any links on specific topics that you may find useful that don’t easily fall into the other categories.

Selecting the Right Number of Cores for a VASP Calculation

Matter Modeling Stack Exchange

Gaussian Error Wiki

More Gaussian Error Help

Contributing (WIP)

This page will explain how to how to report issues and suggest features for the welcome guide. In particular, this page should explain which templates to use and how to use them.

- issues
- features (tutorials, documentation)

Building the Documentation

This page describes how the documentation pages of the Welcome Guide are built.

Build Tools

This guide uses `mkdocs` to convert the Markdown files to HTML files and build the webpage. All the configuration for using `mkdocs` is contained within the `mkdocs.yml` configuration file in the project root. For a detailed description, please see the mkdocs documentation. For theme options, please see the documentation for material for mkdocs.

Building the Webpage

To build the webpage, ensure that `mkdocs` and the necessary extensions specified in the `mkdocs.yml` configuration file are installed in the activate Python environment. For convenience, the `docs` Hatch environment is maintained to contain all necessary Python dependencies for building the documentation. To build the webpage, run

```
mkdocs build
```

from within a suitable Python environment. The output should be something like this:

```
INFO    -  Cleaning site directory
INFO    -  Building documentation to directory: /Users/ugo/Projects/nwt/welcome-guide/docs/s
INFO    -  The following pages exist in the docs directory, but are not included in the "nav
             - resources/conferences.md
             - resources/links.md
             - resources/references.md
             - resources/social.md
             - software_pages/index.md
INFO    -  Documentation built in 0.19 seconds
```

Serving the Webpage

To view a live version of the Welcome Guide as a webpage, ensure that `mkdocs` and the necessary extensions specified in the `mkdocs.yml` configuration file are installed in the activate Python environment. For convenience, the `docs` Hatch environment is maintained to contain all necessary Python dependencies for serving the documentation. To serve the webpage, run

```
mkdocs serve
```

The output should be something like this:

```
INFO    -  Building documentation...
INFO    -  Cleaning site directory
INFO    -  The following pages exist in the docs directory, but are not included in the "nav
             - resources/conferences.md
             - resources/links.md
             - resources/references.md
             - resources/social.md
             - software_pages/index.md
INFO    -  Documentation built in 0.16 seconds
INFO    -  [16:49:00] Watching paths for changes: 'docs/sources', 'mkdocs.yml'
INFO    -  [16:49:00] Serving on http://127.0.0.1:8000/
```

Now, enter the IP address shown, (here, `127.0.0.1:8000/`), into your browser to view the webpage.

Building the PDF Version of the Guide

To build the PDF version of the guide, you will need to install Pandoc and a LaTeX engine. Then run

```
cd docs/sources && pandoc --file-scope -s -o ../../comcat-lab-welcome-guide.pdf -f markdown
```

Getting Started for Local Development

This page covers the basic setup to be able to locally develop the welcome guide.

Installing Prerequisites

1. Install Python 3.10 or greater.
2. Install Hatch, the build backend and environment manager for the welcome guide. If `pipx` is installed:

   ```
   pipx install hatch
   ```

   Otherwise, follow the instructions here.

Getting the Source Files

1. Create your own fork of the Welcome Guide (look for the "Fork" button on GitHub).
2. Clone your fork Welcome Guide repository.

   with `ssh` (requires that your `ssh` key is added to your GitHub account):

   ```
   git clone git@github.com:yourusername/welcome-guide.git
   ```

   with `https` (requires login to GitHub account):

   ```
   git clone https://github.com/yourusername/welcome-guide.git
   ```

Making Your Changes

1. Change your current working directory to be that of the project root.

   ```
   cd welcome-guide
   ```

2. Initialize the development virtual environment.

   with Hatch:

   ```
   hatch env create default
   ```

   with pip:

   ```
   python -m venv .venv && python -m pip install .
   ```

3. Create and checkout a branch for your local changes. (Not directly committing changes to the `main` branch helps ensure that, if synced with the fork source, everyone's main branch `main` means the same thing.)

   ```
   git checkout -b name-of-feature-or-change
   ```

4. Activate the development environment.

   If installed via Hatch:

   ```
   hatch shell
   ```

   If installed via pip:

   ```
   source .venv/bin/activate
   ```

5. Make your changes (e.g., add/change/remove files).

6. Commit your changes.

   ```
   git commit -S -m "I made a change"
   ```

7. Push your changes to **your** remote.

   ```
   git push origin name-of-feature-or-change
   ```

8. Create a pull request for your change on GitHub.