# ARC Tutorial & Getting Started

## Basic ARC Documentation & Resources

**Create an account:** see Create an Account.

**Access ARC**

These videos will walk the user through accessing our systems for the first time (and streamlining access for subsequent logins): Login with SSH plus Using SSH Keys and Agent to simplify logins

This is a **simpler version** of what is explained in the above videos:

1. in terminal/prompt on your local computer, run: `ssh-keygen -b 4096 -t rsa`
2. it will ask you to enter the file in which to save the key; press enter/return to use the default setting (which is `$HOME/.ssh/id_rsa` on MacOS/Linux and `$env:USERPROFILE\.ssh\id_rsa` on Windows)
3. it will then ask for passphrase; simply leave it blank for no passphrase by pressing enter/return
4. on MacOS and Linux systems, run: `ssh-copy-id <your_PID>@tinkercliffs2.arc.vt.edu`; on Windows, run `type $env:USERPROFILE\.ssh\id_rsa.pub | ssh <your_PID>@tinkercliffs2.arc.vt.edu "cat >> .ssh/authorized_keys"`
5. enter your PID password (note: this is your PID password, not your network password; you will not be able to see your password as you type it in but hit enter once you think you're correct)
6. **check the DUO app on your phone to approve the access**
7. now the key has been added to your ARC account and you can log in by running: `ssh <your_PID>@tinkercliffs2.arc.vt.edu` (if you set up a passphrase in step c, it will ask for that; otherwise, you can directly log in)

Open OnDemand is an alternative option for login but not recommended. It can be accessed here.

Acceptable use of a login node includes composing or editing a job script with a text editor, submitting and monitoring jobs, and organizing files. Please avoid running your computationally intensive jobs at the login node (>3 min/job). Please use a compute node as part of a job. See Submit Interactive Jobs and Submit Batch Jobs. For more details, see the Use of a Login Node.

**VT Documentation**

**Video Tutorials**

**FAQs**

## First Time Logging In

Run `cd ~ && ls -a`. This will list all files under your own `$HOME` directory. Follow the instructions below to finish the configuration for Bash.

1. Run `nano .bashrc` (or your favorite text editor) to open the file
   a. Use the arrow keys to go to the last line and add the following lines to the file

```
if [ -f /projects/leaph/bash_config/bashrc ]; then
  . /projects/leaph/bash_config/bashrc
fi
```

   b. Now press Ctrl+O on keyboard and hit enter/return to save the file. Then press Ctrl+X to exit the text editor.

2. Check `.bash_profile`

- If `.bash_profile` exists, run `cat .bash_profile` to print the file. Make sure it contains the following lines:

```
if [ -f ~/.bashrc ]; then
  . ~/.bashrc
fi
```

- If there is no `.bash_profile` file, run `cp /projects/leaph/bash_config/bash_profile ~/.bash_profile`

**Now log out and log in again.** The two files should look like the following:

```
[redacted]@tinkercliffs1:~$ bat .bash_profile
       File: .bash_profile
   1   if [ -f ~/.bashrc ]; then
   2     . ~/.bashrc
   3   fi
[redacted]@tinkercliffs1:~$ bat .bashrc
       File: .bashrc
   1   # this is required for module to work
   2   if [[ -f /etc/bashrc && ! $BASH_SOURCED ]]; then
   3       . /etc/bashrc
   4   fi
   5
   6   if [ -f /projects/leaph/bash_config/bashrc  ]; then
   7     . /projects/leaph/bash_config/bashrc
   8   fi
```

## Python Environments

The latest version of Mambaforge distribution with Python 3.11.3 is pre-installed. The base environment already includes some common packages used within the group such as pandas, numpy, scipy, matplotlib, seaborn, and scikit-learn (see the list below). If you only need to use these packages, you can skip the following steps and modify/run code as you like. If you ever need to use additional packages, please follow the instructions below to create/activate a new development environment.

1. Create a new Python environment by running `mamba create --name envname` where `envname` is the name of your environment.
   - By default, the env is installed in `$HOME/.conda/envs/envname`. Please do not change this!
   - In case you want any customization, you can find more info here.
   - Note: If you want to create an environment using non-3.11 Python, please refer to bullet point 3 in section "Creating an environment with commands" of the above link.
2. To activate an environment, run `source activate envname`.
   - **DO NOT** use `conda activate` or `mamba activate`
   - See here for why
3. To install packages in the environment, run `mamba install pkgname`
   - **DO NOT** use `conda install`
   - You can search for package names on https://anaconda.org/
   - If the package requires non-3.11 Python, you need to create the environment by specifying the version. Please refer to bullet point 1 above.
   - ~~If you encounter any permission related issues, try to create a `.condarc` file in your `$HOME` directory and put into the file the following text~~

     ```
     pkgs_dirs:
       - ~/.conda/pkgs
       - /projects/leaph/.pyenv/versions/mambaforge/pkgs
     ```

     ~~This will overwrite the default conda/mamba configuration to prioritize $HOME/.conda/pkgs for caching packages for your new environment. More info about `.condarc` can be found here.~~
4. To deactivate the environment and get back to the base environment, run `conda deactivate`
   - **DO NOT** use `mamba deactivate`
5. To list all environments, run `mamba info --envs`
6. More info about environment management can be found here, including sharing and removing

### Preinstalled Python packages in the base environment

- jupyter lab: `mamba install jupyterlab`
- jupyter notebook: `mamba install notebook`
- pandas: `mamba install pandas`
- numpy: `mamba install numpy`
- scipy: `mamba install scipy`
- matplotlib: `mamba install matplotlib`
- seaborn: `mamba install seaborn`
- scikit-learn: `mamba install scikit-learn`
- scikit-bio: `mamba install scikit-bio`

- basemap: `mamba install basemap`
- statannot: `mamba install statannot`
- statannotations: `mamba install statannotations`

## Other environments

You can run `mamba info --envs` to show all environments.

## R Scripts and Packages

Several essential R packages from Anaconda have already been installed in the base environment, including

- r-essentials
- r-base
- rpy2

If you need any packages that are not listed above, please create your own environment following the instructions here. (Note that you need to run mamba instead of conda.)

## Pre-Installed Software

The following software packages are pre-installed in the **base** Python environment and are available to all group members; if you need to use other software, please ask Jingqiu Liao if it should be installed in your `$HOME` directory or made available to all group members. If you need to install in your own `$HOME` directory, please note that some software can be installed via the `mamba` command (see section Python Environment).

1. GTDB-Tk version 1.7: https://github.com/Ecogenomics/GTDBTk
    - via `mamba install -c conda-forge -c bioconda gtdbtk`
    - Why not 2.*? See: https://github.com/Ecogenomics/GTDBTk/issues/416
    - If you want to install version 2.*, please do it in your own environment
2. hyphy: https://www.hyphy.org/
    - via `mamba install -c bioconda hyphy`
3. kSNP: https://sourceforge.net/projects/ksnp/
    - v4.0; installed from the .zip in `/projects/leaph/bin`
    - Anaconda only has older versions
4. RaxML: https://cme.h-its.org/exelixis/web/software/raxml/
    - via `mamba install -c bioconda raxml`
5. snakemake: https://snakemake.readthedocs.io/en/stable/index.html
    - via `mamba install -c conda-forge -c bioconda snakemake-minimal`
6. Quast: https://quast.sourceforge.net/quast
    - via `mamba install -c bioconda quast`
7. MMseqs2: https://github.com/soedinglab/MMseqs2
    - Static build with AVX2 in `/projects/leaph/bin`
8. prodigal: https://anaconda.org/bioconda/prodigal
    - via `mamba install -c bioconda prodigal`
9. pyani: https://anaconda.org/bioconda/pyani
    - via `mamba install -c conda-forge -c bioconda pyani`
10. MUSCLE: https://2018-03-06-ibioic.readthedocs.io/en/latest/install_muscle.html
    - via `mamba install muscle`
11. Kraken2: http://ccb.jhu.edu/software/kraken/

- Installed from https://github.com/DerrickWood/kraken2
- Kraken 2 website: https://ccb.jhu.edu/software/kraken2/index.shtml

12. blast: https://www.ncbi.nlm.nih.gov/books/NBK52640/
13. MetaPhlAn4: https://github.com/biobakery/MetaPhlAn/wiki/MetaPhlAn-4
    - Need to activate environment: `source activate mpa`
    - via `mamba create --name mpa -c conda-forge -c bioconda python=3.7 metaphlan=4.0.1`
14. instrain: https://anaconda.org/bioconda/instrain
    - Need to activate environment: `source activate instrain`
    - via `mamba create --name instrain -c conda-forge -c bioconda -c defaults instrain`
15. Metagenome-Atlas: https://github.com/metagenome-atlas/atlas/blob/master/docs/index.rst
    - Need to activate environment: `source activate atlasenv`
    - via `mamba create -y -n atlasenv metagenome-atlas=2.16.3`
16. RGI: https://github.com/arpcard/rgi#rgi-bwt-tab-delimited-output-details
    - Need to activate environment: `source activate rgi`
    - via `mamba create --name rgi --channel conda-forge --channel bioconda --channel defaults rgi`
17. Samtools: http://www.htslib.org
    - via `mamba install -c bioconda samtools`
18. IQ-TREE: http://www.iqtree.org/
    - Installed in `/projects/leaph/bin/iqtree-2.2.2.6-Linux`
19. HUMAnN 3.7: https://huttenhower.sph.harvard.edu/humann/
    - Need to activate environment: `source activate biobakery3`
20. PopCOGenT: https://github.com/philarevalo/PopCOGenT/tree/master
    - Need to activate environment: `source activate PopCOGenT`
21. ISEScan: https://github.com/xiezhq/ISEScan
    - via `mamba install -c bioconda -c conda-forge -c defaults isescan`

22. SPAdes: https://github.com/ablab/spades
    - ~~Follow instructions for Linux binaries on the GitHub page~~
    - `mamba install -c bioconda spades`
23. Trimmomatic: http://www.usadellab.org/cms/?page=trimmomatic
    - Jar file is installed in `/projects/leaph/bin/Trimmomatic-0.39/trimmomatic-0.39.jar`
    - To run the program, follow the instructions on the webpage: `java -jar trimmomatic-0.39.jar ...`
24. eggNOG-mapper: https://github.com/eggnogdb/eggnog-mapper
    - `mamba install -c bioconda eggnog-mapper`
25. BBmap: https://sourceforge.net/projects/bbmap/
    - `mamba install -c bioconda bbmap`
26. PAL2NAL: https://anaconda.org/bioconda/pal2nal
    - `mamba install -c bioconda pal2nal`
27. fastQC: https://anaconda.org/bioconda/fastqc
    - `mamba install -c bioconda fastqc`
28. bowtie2: https://bowtie-bio.sourceforge.net/bowtie2/index.shtml
    - Need to activate environment: `source activate bowtie2`
    - via `mamba create --name bowtie2 python=3.10.13 bowtie2`
29. TnFinder: https://tncentral.ncc.unesp.br/TnFinder.html
    - Need to activate environment: `source activate tn3`
    - Tn3 Transposon Finder: `/projects/leaph/bin/tn3/Tn3+TA_finder.py`
    - Composite Transposon Finder: `/projects/leaph/bin/tncomp/TnComp_finder.py`
    - IS Associated with Antibiotic Resistance Genes Finder `/projects/leaph/bin/isabr/ISAbR-0.1.6.py`
30. KMA: https://bitbucket.org/genomicepidemiology/kma/src/master/
31. plasmidFinder2: https://bitbucket.org/genomicepidemiology/plasmidfinder/src/master/
    - DB installed in `/projects/leaph/bin/plasmidfinder_db`
32. platon: https://github.com/oschwengers/platon
    - Database downloaded to `/projects/leaph/bin/platon/db`
    - via `mamba install -c conda-forge -c bioconda -c defaults platon`

- follow instructions in the GitHub repository

33. CheckM2: https://github.com/chklovski/CheckM2
    - Repo is downloaded to `/projects/leaph/bin/checkm2` and database is downloaded to `/project/leaph/bin/checkm2/database`
    - Need to activate environment: `source activate checkm2` before running `checkm2`

To-be-installed:

1.

## Submit Interactive Jobs

The following will walk the user through the process of submitting interactive jobs for testing/development and batch jobs for production research runs:

- https://www.docs.arc.vt.edu/usage/faq.html#interac
- Interactive and Batch Jobs

Below is a shorter version.

After logging into the TinkerCliffs login node (e.g., `tinkdercliffs2`), run the `interact` command to submit an interactive job. An example is shown below:

```
interact --time=00:30:00 --mem=16G --account=personal
```

This command sets the duration of the interactive job to 30 minutes. It also specifies the required memory to 16 gigabytes. These two configs (`--time=` and `--mem=`) are optional. The command is using the `personal` account but you may change it to `leaph` if you don't have a personal allocation from ARC. Essentially, `interact` takes any valid sbatch options (see Submitting Batch Jobs for examples; you may also check the official documentation to see the full list of options and their usage).

Record the job id of your interactive job and use `sacct --format="Elapsed,MaxRSS" -j JOB_ID` to check how much time and memory is used. You may need this to determine how much time and memory to request when submitting a batch job.

## Submit Batch Jobs

For long-run jobs, submit via Slurm: https://www.docs.arc.vt.edu/usage/slurm.html#slurm

A sample batch job script is provided on TinkerCliffs at `/projects/leaph/sample_scripts/sample_batch_job.sh`. Below is its content. The lines starting with `#SBATCH` are specifying options passed to the `sbatch` command. You can check the official documentation for the full list of valid options.

```
#!/bin/bash
#SBATCH -J hello-world # job name
#SBATCH --account=leaph
#SBATCH --partition=normal_q
#SBATCH --time=0-00:10:00 # 10 minutes; format: days-hours:minutes:seconds
#SBATCH --mem=1G # based on memory used when testing with interactive jobs
#SBATCH --mail-user=pid@vt.edu #enter desired email address for updates
#SBATCH --mail-type=BEGIN #include to get emailed when job begins
#SBATCH --mail-type=END #include to get emailed when job ends
#SBATCH --mail-type=FAIL #include to get emailed if job fails


# More info: https://www.docs.arc.vt.edu/usage/slurm.html
source activate base # activate env if necessary
echo "hello world from..."
hostname
```

To submit, use command `sbatch`:

```
sbatch  /projects/leaph/sample_batch_job.sh
```

This will return a message with the job ID such as:

```
Submitted batch job 978815
```

To check a job's status, use the `squeue` command:

```
squeue -j 978815
```

To check resources being used for this job, the `jobload` command:

```
jobload 978815
```

To check the status of more than one job or the queues in general, use squeue. Examples include:

```
squeue --users=username  # view only a given user's jobs
```

To remove a job from the queue, or stop a running job, use the command `scancel`:

```
scancel 978815
```

When your job has finished running, any outputs to stdout or stderr will be placed in a file in the directory where the job was submitted. More information can be found at https://www.docs.arc.vt.edu/usage/slurm.html#output.

## Run Jupyter on ARC

1. `interact` with enough memory and time. See Submit Interactive Jobs.
2. On the node, `cd` to a reasonable starting directory
3. Run `jupyter-lab --no-browser --ip='0.0.0.0'` to launch JupyterLab
4. Note server address and token should look like:

```
http://tc307:8888/?token=1b06b6370a4cd8157002f979c9b0e4355101bab0a07a771e
```

5. Open another local terminal, run:

```
ssh -N -L localhost:<available_local_port>:<server_address>
<your_PID>@tinkercliffs2.arc.vt.edu
```

Try using the same port from server address (e.g., 8888) for `<available_local_port>`

6. In your browser, log in to

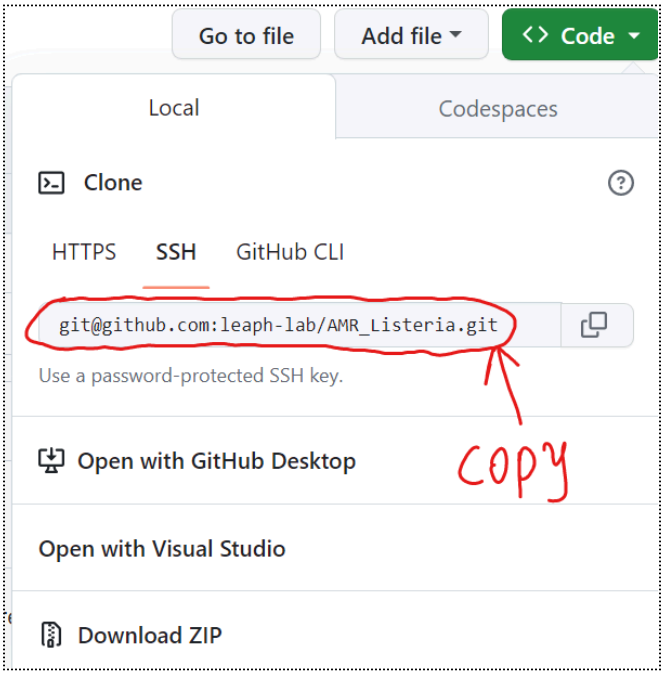
```
http://127.0.0.1:<available_local_port>/lab?token=<token>
```

**An alternative way is to run `launch_jupyter.sh`, a helper script that automatically runs Jupyter and prints out the SSH command and the URL. To terminate Jupyter, press Ctrl-C for a couple of times.**

## Basic Git Usage

The lab GitHub organization page is https://github.com/leaph-lab. All code is managed and shared through GitHub. If you haven't done so, ask the PI to add you as a member of the GitHub organization to get permission to access all code repositories.

1. Add an SSH public key to your account if you haven't done that before. Instructions are available here. This allows you to clone a remote GitHub repository to a local machine (such as ARC's TinkerCliffs server).
2. To clone a repository, go to its page (e.g., https://github.com/leaph-lab/AMR_Listeria) and click on the green "<> Code" button and copy the URL from the "SSH" tab. An example is below:




3. In the terminal, first log into ARC's TinkerCliffs server. Then go to any directory you'd like to store the repo (e.g., `/projects/leaph/liaoj/`) and run the following command to clone the repository to ARC:

```
git clone <repo_git_url_just_copied>
```

4. Now you can go to the local repo (e.g., `cd /projects/leaph/liaoj/AMR_Listeria`) and make any modifications. Note: You may need to run Jupyter in the process.
**Note: Before you start modifying anything, the best practice is to sync your local repo with the remote GitHub repo, just in case there are any updates from your collaborators. To do this, run**

```
git pull
```

5. After you've done with all the editing, run the following command to check your modifications:

```
git status
```

The status message should look like this:

```
[redacted]@tinkercliffs2:/projects/leaph/liaoj/AMR_Listeria$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test

nothing added to commit but untracked files present (use "git add" to track)
```

If there are any undesired changes, go and reverse the changes.

6. Stage the verified changes for the next commit:
    - If all changes look good, run the following command to stage all changes

```
git add .
```

    - If you only want to commit a particular change (e.g., 'test' in above example), run:

```
git add test
```

If you re-check the status now, it should look like this:

```
[redacted]@tinkercliffs2:/projects/leaph/liaoj/AMR_Listeria$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   test
```

7. Now it's time to commit the changes to make them permanent locally:

```
git commit -m "some meaningful text, e.g., the purpose of the commit"
```

If you ever unintentionally made a commit, you could run `git reset HEAD~` to revert it.

8. Now you've checked and committed the changes locally, it's time to push your local commit back to GitHub:

```
git push
```

9. Go to GitHub to double-check that the commit is indeed accepted
10. Repeat steps 4–9 to make any further changes to files.

You may find a more detailed introduction to Git on its official website: https://git-scm.com/doc or GitTutorial (credits to Izaak Coleman in the Korem Lab)

## Install PathoFact

https://git-r3lab.uni.lu/laura.denies/PathoFact

1. Go to any folder under `$HOME` that you want to put the source code in (such as `~/workspace` in the following example) and clone repository. Note: you **MUST NOT** install miniconda. We use the Mambaforge distribution of Python. Please refer to Python Environments for more details.

```
mkdir -p ~/workspace && cd ~/workspace
git lfs install
git clone -b master --recursive \
    https://git-r3lab.uni.lu/laura.denies/PathoFact.git
```

2. Create PathoFact pipeline environment (by default, the env is installed in `~/.conda/envs`)

```
cd ~/workspace/PathoFact
mamba env create -f=envs/PathoFact.yaml --name PathoFact
```

3. To run PathoFact, follow the instructions here.
    - To configure PathoFact, change every parameter in the `config.yaml` file in `~/workspace/PathoFact` that "requires user input"
        - The path for `signalp` is `/projects/leaph/bin/signalp-5.0b/bin`
    - Use `source activate PathoFact` to activate the environment