Co-op Student Orientation Guide

Dale Dupont

December 22, 2017

Contents

1 Introduction

For new co-op students.

It is assumed that you have:

1. Completed your safety training
2. Submitted immunization records

Health Services will notify you if you need any addition immunizations or tests. If needed, you will have to go to the Princess Margaret Staff Clinic in the lower levels of the building. This can be accessed through several levels of elevators.

You will have to attend several orientation events over the course of the first several weeks of the work-term. This includes:

1. General Orientation
2. Research Orientation
3. Research Computing Orientation

You will be automatically scheduled to attend the general and research orientations. However, you will need to receive an e-mail from Susan Alexander in order to be scheduled for Research Computing. Research Computing is the most important orientation, as it will grant to access to the intranet, email, and Employee Self-Service. You will also have to obtain a badge. Your photo will be taken when you go to obtain a badge.

1.1 Advice To Co-Op Students

Based on feedback during performance evaluations, you should take care to address the following:

- Ensure that your software is documented
- Ensure that your software does not break any tests (all tests pass)
- Ensure that your software actually works in the actual deployment environments
- Ensure that your latest software that is used in deployment is made available on GitHub.

Especially at the end of the co-op term, avoid making major functional changes to the software to ensure stability when the next student takes on the project.

Ideally, the best performing students manage to co-publish papers that are indirectly connected to the software they develop. This, however, is extremely rare and requires the project to involve publishable material.

Your supervisors will communicate relatively little with you and you will have to take initiative for the most part to get projects completed and to ensure their quality. This expectation is implicit.

2 Development Timeline

Although there is little documentation of what former co-op students have done, we can infer based on the available repositories, commit histories, and remaining files what the they may have done.

- Jonathan Dursi helps lead the CanDIG project
- The authentication team is formed.
- Spring 2017 - Kevin develops the authentication for the GA4GH server in the authentication branch of the CanDIG GA4GH server.
- Spring 2017 - Another co-op student develops the frontend to the
- Spring 2017 - 2 months prior to Dale and Jone joining the Toronto authentication team of CanDIG, Richard joins the team.
- September 2017 - Dale and Jone examine the authentication code for the GA4GH server
- September 2017 - The standard GA4GH server is successfully deployed
- September 2017 - The Keycloak server is successfully deployed
- September 2017 - The authentication branch of the GA4GH server is successfully deployed
- September 2017 - The manual deployment procedures are drafted
- October 2017 - Dale creates the automated deployer program.
- October 2017 - Dale introduces command-line arguments to the program.
- October 2017 - A GitHub repository is created on BioCore
- October 2017 - Jone creates the Singularity deployment for CanDIG
- October 2017- Dale integrates the Singularity recipe into the deployer.
- October 2017- Dale creates the Token Tracer program.
- October 2017 - Dale introduces command-line arguments to the token tracer program.
- November 2017 - Dale creates a Docker deployment for Funnel.
- November 2017 - Dale creates a Singularity deployment for Keycloak and Ga4GH Server
- November 2017 - Dale forks the CanDIG Ga4GH server code and refactors the authentication branch
- December 2017 - The Singularity authentication deployment is successful at the Genome Sciences Centre
- December 2017 - This document is created along with the work-term report

3 Development Environment

3.1 MacOS

You will generally be developing on a MacOS (this machine) in the dry lab of the 11th floor of the MaRS Medical Discovery Tower.

MacOS has many features similar to Linux, but you will likely at some point need a Linux virtual machine for at least the purposes of testing. Be familiar with the Command Key. The Command Key controls copying, pasting, and cutting, as well as other features normally associated with right-clicks.

3.2 VirtualBox

You can create a Debian image through VirtualBox for such means.

How to create a Debian VirtualBox:

1. Download the Debian image.
2. Run VirtualBox
3. Create a new Virtual guest operating system
4. Load the Debian image
5. Set up the account and root

You may wish to add the user account to sudoers. However, for the testing of Singularity, we cannot assume that this is the case for the end-users, which means it may be better to use `su` to switch between root and the user.

3.3 Docker

Docker will be the primary means of deployment for your projects.

Docker will operate using VirtualBox as a hypervisor. Some newer Macs may support creating virtual machines using a hardware-based hypervisor. This is what the new version of Docker assumes. However, this will not work on this machine.

Docker can be started using the Docker Quickstart Terminal if not following the latest version which requires hardware virtualization support.

`docker cp` will allow you to move files to and from software containers.

Docker images are built using Dockerfiles. Dockerfiles have their own special command language when writing them (`ENV, RUN, CMD, COPY`).

Some useful docker commands:

- `docker cp <FILE> <CONTAINER_NAME>:<PATH>`
  - Copy local file `<FILE>` onto the Docker container `<CONTAINER_NAME>` to location `<PATH>`.
- `docker cp <CONTAINER_NAME>:<FILE> <PATH>`
  - Copy file `<FILE>` from the Docker container `<CONTAINER_NAME>` to local location `<PATH>` on the host.
- `docker build -t <IMAGE_NAME> .`
  - Build a docker image using the local Dockerfile with image tag `<IMAGE_NAME>`
- `docker exec -it <CONTAINER_NAME> bash`
  - Start an interactive terminal bash session in the Docker container `<CONTAINER_NAME>`
- `docker run <CONTAINER_NAME>`
- `docker kill <CONTAINER_NAME>`
- `docker rm <CONTAINER_NAME>`

- `docker container ls`
- `docker images`

See the extensive docker documentation:

`https://docs.docker.com`

Docker can be scaled using Kubernetes in a cluster environment.

Some things that have been suggested before related to containers:

- LXC
- LXD
- Intel Clear Containers

3.4 Text Editors

You may use one of the following (or something else, if you prefer):

- Emacs
- Vim
- Atom
- Sublime Text
- Nano

Note that Nano may not have as many features as other editors. You will generally want to use it in minimal environments (like in a recently deployed software container).

3.5 Singularity

Singularity is an alternative deployment framework for your projects. Singularity is designed for HPC environments by running the software containers without root permissions. This can make configuring deployment more difficult, as you will have to work around root permissions to get them working.

Singularity images are built using Singularity recipe files.

Singularity **cannot** be run on a Mac. There are two workarounds:

1. Use a Vagrant container to create a Linux instance that Singularity may be installed and run in.
2. Create a VirtualBox Linux Image to run Singularity on.

Follow the Singularity documentation for the build instructions. You will have to compile Singularity on the Linux machine in order to install it.

You may use scp to transfer development files to and from these virtual operating systems.

See the singularity documentation:

`http://singularity.lbl.gov`

Use Singularity Hub to distribute Singularity images:

https://www.singularity-hub.org

You will need to create dedicated GitHub repositories and sign in with the same account of those repositories in order to use Singularity.

Some useful Singularity commands:

- `singularity build <IMAGE_NAME.img> Singularity`
  - Build a read-only image `<IMAGE_NAME.img>` from recipe file Singularity.
- `singularity run <IMAGE_NAME.img>`
  - Run `<IMAGE_NAME.img>` as read-only container
- `singularity shell <IMAGE_NAME.img>`
  - Start an interactive shell terminal inside the image `<IMAGE_NAME.img>` as read-only.
- `singularity build --writable <IMAGE_NAME.img> Singularity`
  - Build a writable image `<IMAGE_NAME.img>` from recipe file Singularity.
- `singularity run --writable <IMAGE_NAME.img>`
  - Run `<IMAGE_NAME.img>` as writable container
- `singularity shell --writable <IMAGE_NAME.img>`
  - Start an interactive shell terminal inside the image `<IMAGE_NAME.img>` as writable.
- `singularity image.create <IMAGE_NAME.img>`
  - Create an empty singularity image called `<IMAGE_NAME.img>`.
- `singularity image.expand <IMAGE_NAME.img>`
  - Expand the size of the image `<IMAGE_NAME.img>`.

Changes to read-only containers will not affect their underlying image. Hence exiting a read-only container will cause all changes to be lost. Singularity images only have a finite size. Writing too much information inside a singularity container will result in a disk space error. Image size is determined at build-time automatically, but can be changed using `image.expand`.

3.6 Slack

Slack is the primary means of communication for software development and collaboration. You will contact Richard and other developers and end-users that you will be interacting with through here.

The CanDIG channel is one of the Slack groups that are used in the Bioinformatics Group. You will have to be invited by Richard in order to join. You will also have to register using your UHN email.

https://candig.slack.com/

You can format your messages using markup:

https://get.slack.help/hc/en-us/articles/202288908-Format-your-messages

You will want to get into contact with Richard and Jonathan, and any others you will be developing with or for.

3.7 Email

You will likely correspond with Natalie and Carl through e-mail for administrative affairs (such as your pay).

You will have to attend the Research Computing Orientation you gain access to your e-mail. At the end of the orientation, you will line up to have your e-mail and user account created. You will have to use a password 8 characters in length.

Your email will likely be of the form:

```
firstname.lastname@uhnresearch.ca
```

Your username will likely be of the form:

```
<firstLetterOfFirstName>lastname
```

For example, John Smith's email would be:

```
john.smith@uhnresearch.ca
```

or alternatively:

```
John.Smith@uhnresearch.ca
```

John Smith's username is thus:

```
jsmith
```

Your username will be used to sign onto the internal network and gain access to your pay stubs.

3.8 Remote Access

To get access to the internal network, you will have to get One-Time-Password keys. This is printed on one side of a physical piece of paper. You can have someone print it out for you by entering the research computing office on the fourth floor (It is directly to the right of the elevators along the curved wall). You will need your badge and ID username.

With the OTP keys, you can go to the Research Staff Remote Access portal:

`http://www.uhnresearch.ca/remote`

Type in your username and the corresponding OTP key.

Download a Windows 7 Image and run the Microsoft Remote Desktop application under `Applications` in the Mac Finder.

Run the image that appears there, and log in with your username and password.

You will then be able to access Employee Self-Service using Internet Explorer from the Research Computing website under `Human Resources`.

3.9 Git and GitHub

You will need a GitHub account to use for development. You will be creating repositories on GitHub that will contain all the files relevant to your projects.

You will need to ask Richard in order to gain access to the BioCore group on GitHub.

This is the GitHub Group for the Bioinformatics Core of the Princess Margaret Research Institute (Carl's Group).

`https://github.com/Bio-Core`

You should be familiar with the following with git:

- `git status`
  - Get information about changes staged for the next commit
- `git add <FILE>`
  - Adds `<FILE>` to the next commit
- `git add -u`
  - Updates all pre-existing files for the next commit
- `git rm <FILE>`
  - Deletes `<FILE>` from the commit
- `git reset HEAD <FILE>`
  - Excludes `<FILE>` from the commit
  - `<FILE>` remains locally untracked
- `git commit`
  - Creates a commit
- `git commit --amend`
  - Updates the last commit
- `git clone https://github.com/User/Repo.git`
  - Clones the master branch
- `git clone -b <BRANCH> https://github.com/User/Repo.git`
  - Clones the `<BRANCH>` branch
- `git checkout -b <BRANCH>`
  - Switch to the `<BRANCH>` branch
- `git merge`
  - Merge the commits from the master branch with the current branch
- `git push`
  - Push the commits to the remote repository on GitHub
- `git pull`
  - Add the commits from the remote repository on GitHub to the local repository
- Forking: Perform a fork on GitHub

`https://git-scm.com/docs`

`https://git-scm.com/book/en/v2`

GitHub Releases

You can create releases to distribute large binary files (just be sure to compress them first).

3.10 Wiki

The Princess Margaret Genomics Centre has a wiki:

`https://www.pmgenomics.ca/pmgcwiki/index.php/Main_Page`

You will need to sign-in using your Coop account to access it.

The wiki mainly documents analyses, web sites, and computing systems that are used by other staff.

3.10.1 Bis

`bis` is a tool designed by Princess Margaret Genomics Centre that is used to index PubMed publications by their Medical Subject Heading (MeSH) terms. These are used to determine relevance during searches.

`https://www.pmgenomics.ca/bis/`

3.11 Meetings

You will have to meet weekly with the rest of the Bioinformatics group and report to Natalie on what you have done each week. These usually are on Thursdays at `12:00 pm` in room `11-402` of the dry lab.

3.12 Fire Alarms

Do not do anything in the event of a fire alarm. Due to the size of the facility, only respond to it if instructions have been given explicitly over the public announcement system. Generally, the public announcement system will only say "Please await for further instructions", in which case you do nothing. Elevators will not work during the course of the alarm, so you will have to use the stairs.

4 Accessing and Running the Code

You may develop on a MacOS for the most part. However, you may prefer to simply develop within a virtual machine directly.

There is a Debian virtual machine that has been used to test the Singularity deployment. You can access it with the default username and password `coop` and `waterloo`.

4.1 Deployer

Perform the following to do a full root installation:

```
git clone https://github.com/BioCore/candigDeploy.git
cd candigDeploy
pip install .
candigDeploy
```

You should now have a running GA4GH and Keycloak server.

You can view help for the program using the `-h` option:

```
candigDeploy -h
```

You can deploy funnel with the `-f` option:

```
candigDeploy -f
```

You can also deploy with the token tracer using the `-t` option:

```
candigDeploy -t
```

Change the IP address using the `-i`, `-kip` or `-gip` options. Change the port number using the `-kp` and `-gp` options.

On the Debian virtual machine, after installing the program as a non-root user, you can run:

```
candigDeploy -s
```

Which will deploy the Keycloak and GA4GH singularity containers.

Further instructions can be found in the `README.rst` of the candigDeploy directory.

4.2 GA4GH Server

The GA4GH server may be deployed either using Apache or by using Flask directly.

GA4GH server may be directly installed with the commands:

```
git clone https://github.com/BioCore???
pip install -r requirements.txt
pip install .
ga4gh_server
```

You may use the `-H` option to change the IP and `-p` to change the port number.

```
ga4gh_server -H 192.168.99.100 -p 7000
```

By default, access the GA4GH server at `127.0.0.1:8000` through your web browser.

The index page will show the REST API that you may access.

For instance to search all references, enter the URL:

```
127.0.0.1:8000/references/search
```

POST methods will not work on a web browser. Use curl to send POST requests.

The Deployer allows the authentication branch to be executed:

`https://github.com/Bio-Core/candigDeploy`

4.3 Token Tracer

You may install and run the token tracer through the following commands:

```
git clone
pip install .
tokenTracer
```

You may use the `-a` option to print all HTTP packets. You may use the `-j` option to print in JSON format.

Documentation and the source code is available on the GitHub repository:

`https://github.com/Bio-Core/tokenTracer`

5 Program Maintenance and Development

Run the tests for each program. Ensure that they all pass.

For the CanDIG, GA4GH server, some tests will not pass, as the GA4GH server was never fully developed to provide the features necessary to pass those tests before it was abandoned.

You will likely want to branch the code to develop experimental features. Then, when the code is ready, send a pull request to the branch that you wish to update. If it is a different branch, or has been updated, you will want to merge locally first. Ensure that no tests have been broken.

To create your own repository:

5.1 Documentation

When you write documentation, you have a few choices for the file format:

1. Source-Code Comments (`.py`) (Recommended)
2. Markdown (`.md`)
3. ReStructuredText (`.rst`) (Recommended)
4. LaTeX (`.tex`)

It is best to follow the Python conventions for documentation, and include generous amounts of comments in source-code files and include additional documentation in reStructuredText Files.

Markdown is the simplest markup language and is recommended for generic GitHub projects. Python supports an extension of Markdown called reStructuredText that should be used.

LaTeX is what this document was written in before it was complied to PDF. LaTeX should generally only be written if the documents are intended to be solely distributed as PDFs. I have distributed this as a PDF rather than in .rst format in case you do not yet have a text editor prepared. PDFs generally do not work as well for documenting source code in comparison to reStructuredText.

With reStructuredText, your documentation will automatically be formatted whenever it is pushed to GitHub. You can also preview what the reStructuredText will look like with a converter such as `pandoc`.

`http://docutils.sourceforge.net/rst.html`

Include the following in your documentation

- Overview/Summary
- Design documentation
- Future changes/TODO
- Quickstart
- Tutorials/Walkthroughs
- Examples
- Command-line arguments
- Installation instructions

Python Style Guide PEP:

`https://www.python.org/dev/peps/pep-0008/`

Docstrings PEP:

`https://www.python.org/dev/peps/pep-0257/`

When you comment a method or function, include the type of its arguments and a description of each argument. Also include the type of its return value and a description of what it returns. Also include any side-effects, pre-conditions, or post-conditions. The first line should give an overview as to what the function does.

5.1.1 Comments Outside of Source Code

When you create documentation outside of source code, you should also add comments. These comments will not appear when the documentation is rendered as markup, but they can be used to structure your documentation and explain the inclusion of information.

This is especially useful in writing major documents, where each paragraph can receive its own paragraph of comments that explains the role of the paragraph in the document.

5.1.2 Diagrams

Visuals will help improve the presentation of your work. Often, visuals are the fastest way to communicate information to a reader.

You may design visuals using languages such as `Asymptote`, `PSTricks`, or `Tikz`. However, these are full-fledged programming languages that support computer graphics directly. Using a programming language with computer graphics capabilities gives you precision in designing graphics and makes it easier to make small changes to a diagram.

However, it can take longer to design visuals using a programming language, so you may defer to using a mouse-based interface such as `draw.io`, which is a web application for creating diagrams. There are also other diagramming software such as `Umbrello`. You may also consider vector graphics software such as `Inkscape` or raster graphics software `GIMP`. These programs rely more on your artistic skill than with programming languages.

Draw.io:

`https://www.draw.io`

Asymptote:

`http://asymptote.sourceforge.net`

Note that Asymptote will not compile to the appropriate image size and will remain at page size A8 on MacOS. Compile Asymptote programs on Linux to resolve this (use VirtualBox).

5.2 Packaging

You will need to structure your projects properly.

Conventionally, the root subdirectory will not contain the source code.

Rather it will contain all the subdirectories important to development as well as the source code that is used to build the application.

The top-level directory will also contain code related to packaging the Python project.

This will allow end-users to install the package using pip. This will also allow you to distribute the project using the Python Package Index.

Other directories that you might include could be `/docs` or `/tests`.

A `__init__.py` file must be included in every subdirectory in order for the files in that subdirectory to be considered to be part of the package. Only these files will be included in the package installation and will be detected in Python import statements.

The package itself is a subdirectory of the root directory in a Python package.

5.2.1 File Compression

Use programs like `gzip` to compress large binaries. Programs like `tar` are useful when compressing directories and their subcontents.

This is particularly useful for distributing binaries, container images, and data, such as the Singularity image for Keycloak.

```
gzip file.txt
gunzip file.txt.gz
tar -xvzf dir.tar
```

5.3 Testing

You will have to test your program many times.

It is best that you write automated tests.

However, you will also have to run tests manually in more complicated deployments or as you implement new features.

Tests are a way to guarantee the integrity of a program.

You can use a unit testing framework such as `unittest` or `pytest`.

`https://docs.python.org/2/library/unittest.html`

`https://docs.pytest.org/en/latest/`

`http://nose2.readthedocs.io/en/latest/getting_started.html`

You may also try proving your programs correct using Linear Temporal Logic, Computational Tree Logic, Hoare Triples, and other logics. This is generally very difficult to do.

5.4 Refactoring

You will likely have to do a lot of refactoring, both of your own code and the code of previous co-op students.

With refactoring, you want to reduce the number of lines of code that the program has while retaining its function. This will make the program considerably easier to change and understand in the future, and improve maintainability. Code that becomes too unwieldy often suffers the fate of being completely thrown out and replaced with a new application coded from the ground-up.

A simple way of refactoring things is to place code into different classes and compose the classes together. Classes can be used to group the code together based on its function.

You can also split monolithically large modules into smaller ones that are also separated based on their function. This makes it simpler to understand the code and predict what effects changing code will have on the system.

Boilerplate code can be replaced using classes (an object-oriented approach), using higher-order functions (a functional approach), or using data structures (a declarative approach).

For instance, we can turn a series of calls to the same function initializing different versions of the same object by packing the arguments into a list of tuples, each tuple containing a unique set of arguments. We then iterate over this list, passing each tuple into the object instantiation.

Some design architectures include structuring your group as a tree or graph. You can design nodes that will pass their outputs onto to the nodes that they call. Nodes can then be assigned functions at run-time when the graph is built.

6 Python

Your development will likely take place mostly (if completely) in Python.

Apart from the standard libraries, you will likely have to work with Flask and its extensions. Flask is a framework for developing web applications. It is used in software such as the GA4GH server and the Laboratory Information Management System.

It is recommended that you go through the Python standard library.

`https://docs.python.org/3/library/index.html`

You will be developing in Python 2.7.

A more advanced reference is the Python Language Reference:

`https://docs.python.org/3/reference/index.html`

Some useful things to know:

- Partial Functions
- Decorators, wrappers
- Events, Eventloops, asyncio
- threading, subprocesses
- os, sys, shutil
- Exceptions
- Inheritance
- Classes, methods, fields
- JSON, Yaml, XML

6.1 Strings

You can format strings as follows:

```
"hello {0}".format("world")
```

Which prints `"Hello world"`.

Multiple arguments can be used:

```
"hello {0}, {1} {0}".format("world", "goodbye")
```

Which prints `"Hello world, goodbye world"`.

6.2 Coding Conventions

Flake8 enforces the following coding conventions on GA4GH:

- No trailing whitespace
- No blank lines with whitespace

- One space between methods
- Two spaces between classes
- Two spaces between functions
- Zero to one space between imports
- Inline comments must have a space following the hash (#) symbol

For the sake of running the automated tests through Travis, the GA4GH code must pass Flake8's tests. Flake8 was used by the development team who originally created GA4GH server.

You do not have to follow these mainly aesthetic conventions for your own projects.

However, it is good practice to include:

- Use inline commands
- docstrings on all modules, classes, and functions
- Use encapsulation
- Split code into modules with dedicated functions
- Replace repeated code with helper functions
- Replace code with data structures and algorithms
- Encapsulated related code into a class
- Use composition to join functionally different parts together
- Use singletons to encapsulate globals
- Avoid singletons and globals as much as possible
- Parameterize functions as is helpful
- Include helpful default arguments
- Use functional programming
- Use inheritance to give functionally different code the same interface
- Use list comprehension
- Use lists, dicts, and tuples
- Use YAML configuration files
- Store data in JSON files
- Do not hard-code paths
- Use variables/constants to encode repeatedly used literals
- Unit tests
- Do not repeat code
- Make programs extensible

These practices make the function of your code evident through its structure. It also makes the code less complicated by coupling its behaviour so that changes in the code will effect only small and predictable differences in its behaviour.

You should also document your code further in a dedicated `/docs` subdirectory and document essential end-user information in the `README.rst` in the root directory.

6.3 Packing and Unpacking Arguments

`*` unpacks a list.

`**` unpacks a dictionary.

We may use this to cope with functions that take a large number of arguments:

```
def func(a, b, c, d, e, f, g, i):
    do something

argDict = { "a" : A, "b" : B, "c" : C, "d" : D,
            "e" : E, "f" : F, "g" : G, "i" : I}
func(**argDict)
```

A function may be defined as:

```
def func(*args, **kwargs):
    do something
```

Here, `args` is an alias for unnamed mandatory arguments. `args` is taken as a list.

`kwargs` is an alias for key word arguments. `kwargs` is taken as a dictionary.

`args` precedes `kwargs` since mandatory unnamed arguments must always be included first.

6.4 Iteration

In Python, a for loop is often sufficient for iteration:

```
for i in list:
   do(i)
```

You can use the `range()`

```
for i in range(list):
   do(i)
```

You can also use recursion, iterators, and generators.

Generators PEP: `https://www.python.org/dev/peps/pep-0255/`

Furthermore, list comprehension is a compact way to generate lists:

```
[ func(i) for i in data if i == True ]
```

You can also functional methods such as `map`, `filter`, and `reduce`.

6.5 Logging

Python offers logging facilities through the `logging` package.

You can import a logger and set its logging level.

1. CRITICAL

2. ERROR
3. DEBUG

CRITICAL will result in the fewest statements being written. DEBUG will result in the greatest number of statements being written.

The logger may be configuration to write to a log file rather than stdout.

`https://docs.python.org/2/library/logging.html`

6.6 Data Structures

6.6.1 Tuples and Lists

Lists are useful for mutable sequential data. Tuples are useful for immutable ordered data

A common data structure is a list of tuples for configuration:

```
[(a, A, 0), (b, B, 1), (c, C, 2)]
```

Where the nth entry of each tuple is the same type. Each tuple is the configuration for a distinct object. Uniform data access is guaranteed by the tuple structure.

6.6.2 Dictionaries

Dictionaries are useful for unordered, labelled data.

6.6.3 Sets

Sets are efficient for determining whether an item belongs to some data structure or set.

Use the keyword `in` to determine whether something is in a set.

```
item = thingX
set = { thing1, thing2, ..., thingN }
if item in set:
   do something
```

6.7 Regular Expressions

Regular expressions are useful in searching and parsing. They are a compact means of specifying formal languages or sets of symbols.

`.*`	Match anything
`|`	Match the pattern on the left or right
`[]`	Match any inside
`+`	Match one or more characters
`*`	Match zero or more characters
`.`	Match any single character
`$`	Match the end of the line
`^`	Match the beginning of the line

6.7.1 Globbing

Globbing is not the same as using regular expressions. By default, the bash shell performs globbing.

Regular expressions are available only as command-line arguments to certain programs such as grep or egrep.

6.8 Python Packaging (pip)

You will need packaging in order to allow your program to be run from the command-line in any location and for the program to be installed or downloaded using pip.

You will have to follow the directory structure for a Python package.

In the top-level directory, create the following:

```
setup.py
MANIFEST.in
setup.cfg
```

You will also want:

```
README.rst
License.txt
```

The `README.rst` will be the documentation and information that users first see when they view the project on GitHub or on PyPi.

`https://packaging.python.org/tutorials/distributing-packages/`

`https://setuptools.readthedocs.io/en/latest/`

6.8.1 pkg_resources

To avoid hard-coded paths, you can use the `pkg_resources` library to get the absolute path of your program files regardless of their location on the system. These must be relative to the directory of the current module, so it is best to nest these files in lower subdirectories.

6.8.2 Licensing

The `License.txt` makes it legal for others to download and use your code. You may choose one of the following licenses:

1. GNU Public License (GPL)
2. Apache 2.0 License
3. MIT License

The MIT License is the simplest and least restrictive. You will generally want to copy the license of the libraries and programs that you are using for development.

6.8.3 Custom Modules

With a Python package, declare `__init__.py` in each subdirectory with which you want to import python modules into other modules. You can then write import statements that declare the sequence of subdirectory names, ending with the module name, and starting with the first subdirectory from the top-level directory.

You do not need to include `__init__.py` in directories that only contain data or configuration files. However, you will need to include them in `MANIFEST.in` in order to be copied to the installation directory using pip.

The top-level directory of your package (where `setup.py`) cannot be imported.

6.9 argparse

To develop command-line programs, you will need argparse (or some other similar library). Argparse implements command-line options in Python.

`https://docs.python.org/3/library/argparse.html`

7 Networking

7.1 Network Packets

Network packets has various layers:

1. Frame
2. Ethernet Protocol
3. Internet Protocol
4. Transmission Control Protocol
5. Hypertext Transfer Protocol

In a packet sniffer program, you can filter based on layers and their fields.

7.2 Pyshark

Pyshark is used to implement the token tracer. Pyshark's packet capturer objects are used to provide an interface to access packets on a network interface for inspection.

To obtain packets, Pyshark interacts with instances of `tshark`, the terminal implementation of `Wireshark`. Both of these packet sniffing programs, along with `tcpdump`, are based on the `libpcap` C library.

`https://github.com/KimiNewt/pyshark`

`http://kiminewt.github.io/pyshark/`

tcpdump: `https://www.tcpdump.org/tcpdump_man.html`

Wireshark: `https://www.wireshark.org/docs/wsug_html_chunked/`

tshark: `https://www.wireshark.org/docs/man-pages/tshark.html`

7.3 HTTP Protocol

`cURL` may be used to send requests programmatically in place of a web browser.

`https://curl.haxx.se/docs/`

An HTTP request contains a header and body. HTTP packets may be requests or responses. Requests indicate their method. Responses will indicate their response code. Only certain methods are allow on certain URLs.

Important HTTP methods:

- GET
  - Get the URL specified
  - Data may be passed by append to the URL
- POST
  - Send data to the URL specified
  - Data is carried in the body
- OPTION

  - Show available HTTP methods accepted by the URL

- HEAD
  - Show only the response headers

These are important for communicating with the GA4GH API.

Hypertext Transfer Protocol 1.1: `https://tools.ietf.org/html/rfc2616`

7.4 Web Servers

7.4.1 Web API

An application programming interface may be made available through the network with a world-wide web connection by making available certain endpoints.

Endpoints are constructed using URLs. Each unique URL represents a unique endpoint. Endpoints are registered to accept only create endpoints and to provide specific responses. Endpoints may be programming through a web framework such as Flask.

The GA4GH server contains a frontend that encodes the endpoints for the GA4GH server. Keycloak has endpoints that are used during authentication.

7.4.2 Apache

Apache is used in production environments. Apache provides load-balancing. Apache is configured through the /etc/apache2 directory. Here, the ports that the Apache server may listen to are changed through the ports.conf file. Servers under the sites-available subdirectory may be used by the Apache server. Servers under the sites-enabled subdirectory are actively executed by the Apache server.

`https://httpd.apache.org/docs/2.4/`

7.4.3 Flask

Flask is a Python web framework that allows Python programs to be written that operate as web servers.

Flask implements the WSGI Python standard for interacting with web servers such as Apache.

The Flask server works through the Flask application object. It contains all the information related to the server, including its configuration. It is also the object that is run to execute the server. This same object is also used to register endpoints.

`http://flask.pocoo.org/docs/0.12/`

8 Authentication

8.1 Keycloak

Keycloak is used as the authentication server of the CanDIG infrastructure. Keycloak hosts an embedded H2 java database. Keycloak is written in Java.

Keycloak implements the Open ID Connect standards for authentication protocols.

More information may be found on Keycloak's website:

`http://www.keycloak.org`

Keycloak's Documentation is available through their website:

`http://www.keycloak.org/docs/latest/getting_started/index.html`

Keycloak's GitHub may be accessed here:

`https://github.com/keycloak/keycloak`

Keycloak also supports authorization. However, it has not been decided as to whether to use Keycloak to implement authorization, as there are many third-party alternatives for a separate authorization layer.

8.1.1 Basic Deployment

1. Download the Keycloak server:

`http://www.keycloak.org/downloads.html`

```
wget https://downloads.jboss.org/keycloak/3.4.0.Final/keycloak-3.4.0.Final.zip
```

2. Extract the file:

```
unzip keycloak-3.4.0.Final.zip
```

3. Run `bin/standalone.sh`:

```
./keycloak-3.4.0.Final/bin/standalone.sh
```

The Keycloak server will then be made available on localhost over port 8080 with no configuration. You wil have to create an administrator account once you log in.

8.1.2 Configuration

You may configure the Keycloak server either via the Administration Console from a web browser after logging into the master realm as the administrator, or via the command-line interface program `bin/kcadm.sh`.

Admin Console: `http://www.keycloak.org/docs/latest/server_admin/index.html#admin-console`

Command-line interface `kcadm.sh`: `http://www.keycloak.org/docs/latest/server_admin/index.html#the-admin-cli`

Note that `kcadm.sh` requires a Keycloak server to aleady be running, as it will attempt to log into it.

The configuration of the Keycloak server can then be exported once configured. The Keycloak configuration can be exported either as a directory of files or a single monolithic JSON file. The exported file or directory can

then be imported into an arbitrary number of other Keycloak servers in order to replicate the configuration settings on other deployments.

The administration console only offers a partial export of some of the data of the server. To fully export all of the Keycloak data (including database information), the original command-line script `standalone.sh` must be used:

```
bin/standalone.sh -Dkeycloak.migration.action=export
-Dkeycloak.migration.provider=singleFile -Dkeycloak.migration.file=<FILE TO EXPORT TO>
```

Importing is done when the `standalone.sh` is invoked to start the Keycloak server:

```
bin/standalone.sh -Dkeycloak.migration.action=import
-Dkeycloak.migration.provider=singleFile -Dkeycloak.migration.file=<FILE TO IMPORT>
-Dkeycloak.migration.strategy=OVERWRITE_EXISTING
```

Importing and exporting: `http://www.keycloak.org/docs/latest/server_admin/index.html#_export_import`

8.2 Open ID Connect

Open ID Connect is a specification for authentication which uses the exchange of JSON Web Tokens. These tokens are signed using the JSON Web Signature and encrypted using the JSON Web Encryption.

These tokens are stored locally on secure applications. Such tokens are associated with an expiry time, after which new tokens will have to be requested. They may be requested using Refresh tokens provided those tokens have not expired. Otherwise, the user will have to log in again.

Tokens may be intercepted from a Keycloak server using the Token Tracer program. If we decrypt a token we can learn some information about the user to which the token belongs.

Open ID Connect adds an identity layer on top of OAuth 2.0 by defining Identity Tokens.

`http://openid.net/specs/openid-connect-core-1_0.html`

8.3 OAuth 2.0

OAuth 2.0 is the successor to the OAuth 1.0 standards.

OAuth 2.0 defines Access Tokens which are used to grant access to resources.

Resources can be services or data provided by remote servers.

OAuth 2.0 Framework: `https://tools.ietf.org/html/rfc6749`

Bearer Token Usage: `https://tools.ietf.org/html/rfc6750`

Threat Model and Security Considerations: `https://tools.ietf.org/html/rfc6819`

JSON Web Signature: `https://tools.ietf.org/html/rfc7515`

JSON Web Encryption: `https://tools.ietf.org/html/rfc7516`

JSON Web Token: `https://tools.ietf.org/html/rfc7519`

User-Managed Access: `https://docs.kantarainitiative.org/uma/rec-uma-core.html`

8.4 Flask-oidc

A Flask extension that has been used to provide authentication for the GA4GH server. Flask-oidc provides an interface for the Keycloak server to communicate with the authenticated GA4GH server.

This is done by wrapping endpoints in Flask with security methods. In particular, for end point method `getEndpoint()`, we decorate the function with `require_login()`:

```
# app is the global Flask application singleton
app = flask.Flask(__name__)
# oidc is the flask-oidc extension OpenIDConnect singleton
oidc = flask.ext.oidc.OpenIDConnect(app)

@oidc.require_login
def require_login():
    do something
```

This redirects the user to the Keycloak login screen whenever they attempt to request an endpoint without appropriate credentials.

`http://flask-oidc.readthedocs.io/en/latest/`

`https://github.com/puiterwijk/flask-oidc`

9 Canadian Distributed Infrastructure for Genomics (CanDIG)

The CanDIG project is a Canada-wide project with teams based in Toronto, Montreal, and Vancouver.

Dale's work concerned the CanDIG project. Jone worked on the Laboratory Information Management System. The Toronto team is responsible for developing the Authentication capabilities of the entire CanDIG project. In order to solve this, we are using Keycloak, which implements authentication to secure the application servers that compose the CanDIG project.

Dale's work is subdivided into three interrelated projects:

1. Deployer Program
2. Token Tracer
3. GA4GH Fork

The Deployer program is designed to deploy a working version of the CanDIG project based on the applications that are working and have been chosen for support. This is important for testing in development.

The token tracer is part of the logging functions of the CanDIG project and is also considered an independent program for use by Carl's group. The Bioinformatics group maintains its own Keycloak server of which the token tracer may be used to monitor packet activity.

The GA4GH variant/reads server was created by the Global Alliance for Genomics and Health Consortium. It has been abandoned and it no longer maintained.

CanDIG is currently using this server and is developing its own version based on the code. The authentication work is concentrated in securing the Flask frontend of this server.

Some of the objectives that have been set out are:

- Fix the tests for GA4GH server
- Containerization of Keycloak, ga4gh-server (via Docker, Singularity)
- Packaging/Containerization of test data
- Review and update funnel
- Examine job-running backends for funnel
- Determine how to implement shims for staging in/out data through APIs

`https://candig.github.io`

9.1 Global Alliance for Genomics and Health (GA4GH) Server

A variant/reads API server designed by the Global Alliance for Genomics and Health. The project has been abandoned, but a fork has been used for the development of the CanDIG project. Eventually, it is planned to replace GA4GH with a new variants/reads server for CanDIG.

`https://github.com/CanDIG/ga4gh-server`

The authentication branch contains the authentication developed by former co-op students.

The fork on BioCore contains the latest modifications to the authentication code:

`https://www.ga4gh.org`

Current development is focusing on getting the tests working on the authentication branches (mainly `auth-deploy-stable-test`).

There has been an attempt (by Dale) to refactor the frontend (in the branch `auth-deploy-fixes`), but this has resulted in the failure of the majority of the test suite.

Branch `auth-deploy-stable-test` contains the configuration functionality needed (located in the `frontend.py`) as well as deployment configuration in the (`/deploy` directory for apache) in order to deploy the GA4GH server in Docker and Singularity containers.

The unit tests are not well understood, and there are many of them. However, current development has managed to reduce the number of tests (and endpoints) necessary.

9.1.1 Flask Execution

GA4GH server may be run directly through Flask by pip installing the GA4GH server:

```
pip install -r requirements.txt
pip install .
ga4gh_server
```

9.1.2 Apache Execution

GA4GH server may be run through Apache:

```
pip install -r requirements.txt
pip install .
cp deploy/001-ga4gh.conf /etc/apache2/sites-enabled
cp deploy/ports.conf /etc/apache2
apache2ctl restart
```

Apache executes by being told where the application.wsgi is from the 001-ga4gh.conf file. `application.wsgi` tells Apache which class controls the application by naming it `application` in the module.

9.1.3 Authentication

Authentication is implemented in the GA4GH server by decorating the endpoints with a special decorator `requires_login` from the `flask.ext.oidc` library.

There are also two other non-functional decorators:

1. The old OAuth 2.0 wrapper `requires_auth`
2. The custom `requires_token` wrapper

The `requires_token` wrapper essentially does nothing, requiring `KEYCLOAK` to be set to `True` in the configuration.

9.1.4 Running the Tests

In the root directory of the GA4GH server, execute `ga4gh_run_tests`. This will execute the test suite in the following order:

1. Run the Flake8 static linter

2. Run the nose2 unit tests

The Flake8 linter has rather strict requirements on the code syntax, and must be satisfied in order to proceed with the unit testing.

9.2 Funnel

Funnel is a project to develop a job scheduling tool for performing bioinformatics analyses. Funnel may support various backends.

A co-op student designed an authentication frontend written in NodeJS (Javascript) in order to add authentication facilities to Funnel. The official authentication capabilities only support HTTP Basic, which is insufficient for CanDIG's needs.

`https://ohsu-comp-bio.github.io/funnel/`

`https://github.com/ohsu-comp-bio/funnel`

`https://github.com/CanDIG/funnel-node`

9.3 PROFYLE

Integrates with the GA4GH server.

The web-based dashboard that it provides must be authenticated.

`https://github.com/CanDIG/PROFYLE_ingest`

9.4 htsget

Used for large-scale data transfers between sites.

`https://github.com/CanDIG/htsget`