iVirus Documentation

Benjamin Bolduc

Jul 24, 2019

Contents:

iVirus is "a community resource that leverages the CyVerse cyberinfrastructure to provide access to viromic tools and data sets." We're focused on 3 major areas:

1. Provide *easy*-to-use tools/apps through CyVerse and *now* KBase! (wahoo!)
2. Detailed protocols through Protocols.io.
3. Databases and collections of "useful" viral datasets to the broader research community on CyVerse and KBase.

We won't go into much detail on the "science" side of iVirus - for that - check the references below and the those associated with specific tools.

This site is mainly to provide a singular resource for the code and links to use and/or understand the tools. If you're here and don't want coding stuff but want to be able to run the tools you find on CyVerse, then. . .

Go here for a collection of iVirus related protocols.

Go here was the original location for iVirus. It's not code-heavy, and was designed while iVirus (and its sister site, iMicrobe) was in its infancy. Over the years iVirus has grown onto multiple platforms, each with its own documentation requirements, and users - from non-informatician scientists to full-blown CS programmers - want info at the level of their appropriate backgrounds. Unfortunately, there's no single-page solution (or maybe there is, contact me if you are aware of one) to integrating all iVirus tools, documentation, historical background, among others. Sadly, there's only so many websites (nearly a dozen DIFFERENT PLACES) one can manage simultaneously and keep up-to-date, and iVirus.us has fallen a bit out of date. While we try to keep it updated, it's usually the last place. So look to this site for the most recent information, alongside protocols.io.

CHAPTER 1

Introduction and Overview

Based on the existing CyVerse cyberinfrastructure we are developing tools, data and metadata resources specific to viral ecology through the iVirus project. This project focuses on challenges unique to viral biology, by developing tools specific to viruses and metagenomics to enhance data reuse and collaboration among viral researchers. The CyVerse cyberinfrastructure promotes efficient data sharing, use of common compute resources, and a platform for developers to securely build new Apps and data processing pipelines for viral ecology. Specifically, we are:

1. Developing Apps and data pipelines to analyzing large-scale viral and metagenomic datasets
2. Integrating disparate viral datasets in the iVirus Data Commons

3. Using metadata capabilities and standard ontologies in the CyVerse cyberinfrastructure to enhance data and software discovery and reuse

This work is a product of the Hurwitz Lab at the The University of Arizona and the Sullivan Lab at The Ohio State University.

1.1 Full Disclosure on App Selection

Apps selected for inclusion in iVirus (and therefore CyVerse, and maybe KBase) is somewhat biased by the developer's background and experience using these tools, not to mention licensing agreements. All tools have been used by the developers, on both local and HPC environments. *Most* tools have been benchmarked, or at the very least used under a number of different parameters for a wide variety of data types. In many cases, recommended parameters are taken from the literature (cited where appropriate), though not every app is as highly "favored" as others. The apps selected are commonly those that have worked well for the Sullivan lab under a variety of conditions for their data. Each iVirus user *may not have the same experience*, so if one of the apps (ex: an assembler) doesn't work for a user's read data, then trying another is totally worth doing. iVirus seeks to bring *a lot* of useful viral ecology apps to the community - it won't get them all and might not be able to keep up with every tool published - but it'll do what it can with its available resources.

If a user has a recommendation for a tool, or is an author themselves and would like to see their app included, please do not hesitate to contact us!

1.2 Authorship and Citations

iVirus is built on tools developed by the Sullivan and Hurwitz labs, and expanded through inclusion of other, 3rd party tools. Anyone using iVirus tools (and the underlying programs) are asked to cite:

iVirus: Facilitating new insights in viral ecology with software and community data sets imbedded in a cyberinfrastructure. (2017) Bolduc, B., Youens-Clark, K., Roux, S., Hurwitz, B.L., and Sullivan, M.B. ISME Journal

Additionally, nearly all tools have some sort of reference that can be cited. These should be cited where appropriate. Citation information is included with every app, and here as well. If an author wishes to update their tool's citation or adjust how it's used, please let us know.

CHAPTER 2

Apps and Tools

At the center of *nearly* every app/tool is a singularity container. Singularity is a container solution we leverage at iVirus to make delivering apps/tools easier. A tool needs to be built only once, and its image can be run on a variety of local compute and HPCs. Not only is this easier on the developer, this lets them focus on research as well!

All tools are accessible as Apps in the CyVerse Discovery Environment (formerly iPlant). The CyVerse Cyberinfrastructure is a freely available resource for computation, storage, and data analysis for the life sciences. As mentioned elsewhere, we are also bringing some of these apps to The Department of Energy Systems Biology Knowledgebase (KBase), a software and data platform designed to meet the grand challenge of systems biology: predicting and designing biological function. We plan to extend the list of tools for viruses as long as we continue to receive funding (and sometimes beyond). We've also included more generalized apps for metagenomics and microbial ecology available through the iMicrobe Project.

Below is a list of every single app available through iVirus on CyVerse (both "old" and "new" versions), as well as a few yet-to-be integrated ones. It will be updated as frequently as time allows, though feel free to contact us if there's any mistakes or omissions.

2.1 The Basics: Using Singularity

Before you can use any of these apps locally, you'll need to read *Singularity 101*.

Example: One of the iVirus singularity containers is Prodigal. To build and run this container,

```
sudo singularity build Prodigal.simg Prodigal.def
singularity run Prodigal.simg --help
```

If everything worked out, the final command should pull up Prodigal's help menu. If it didn't, you'll have to do some troubleshooting to identify what went wrong.

2.2 Quality Control Apps

Generally speaking, quality control (QC) is a technique applied to to [most commonly] raw read data. This ensures that the data going into the assembly (common next step) is of high quality. Poor read quality can result in mis- or incorrectly assembled sequences. Most frequently, read data QC involves trimming reads according to their quality scores. Although some assemblers do not require QC'd reads, we highly recommend it!

2.2.1 Trimmomatic

CyVerse App

Reference: Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. Bioinformatics, btu170.

Short description: Identifies adapter sequences in raw sequencing reads and quality filters

Singularity use

2.2.2 Btrim

CyVerse App

Reference: Kong, Y. (2011) Btrim: a fast, lightweight adapter and quality trimming program for next-generation sequencing technologies. Genomics. DOI: 10.1016/j.ygeno.2011.05.009

Short description: Trims adapters and low quality regions

Singularity use

2.2.3 Scythe

CyVerse App

Reference: Buffalo V. Scythe - A Bayesian adapter trimmer (version 0.994 BETA) [Software]. Available at https://github.com/vsbuffalo/scythe

Short description: Identifies contaminating sequences in read data based on a Bayesian approach

Singularity use

2.2.4 Sickle

CyVerse App

Reference: Joshi NA, Fass JN. (2011). Sickle: A sliding-window, adaptive, quality-based trimming tool for FastQ files (Version 1.33) [Software]. Available at https://github.com/najoshi/sickle.

Short description: Sliding window quality trimmer, designed to be used after Scythe

Singularity use

2.3 Gene Calling

2.3.1 FragGeneScan

CyVerse App

Reference: Mina Rho, Haixu Tang, and Yuzhen Ye. FragGeneScan: Predicting Genes in Short and Error-prone Reads. Nucl. Acids Res., 2010 doi: 10.1093/nar/gkq747

Short description: FragGeneScan is an application for finding (fragmented) genes in short reads

Singularity use

2.3.2 Prodigal

CyVerse App

Reference: Hyatt, D. Prodigal (2.6.3) [Software]. Available at https://github.com/hyattpd/Prodigal

Short description: Fast, reliable protein-coding gene prediction for prokaryotic genomes.

Singularity use

2.4 Assemblers

Following read trimming and QC, reads can now be assembled into contiguous sequences ("contigs"). Most "recent" assemblers are designed to assemble Illumina data (short read lengths, massively deep sequencing) and are based on De Bruijn graphs (original ref). Assembler selection is dependent on the type of read data being assembled (often 454 vs Illumina vs Pacbio), source material (DNA vs. RNA, eukaryotic vs prokaryotic) and/or sample-specific determinants that may have biased the reads (high/low coverage, repetitive sequences, amplification polymerase, etc.). There is no "best" assembler, though there are assemblers that perform better with viral metagenomes than others.

2.4.1 SOAPDenovo

CyVerse App

Reference: Luo et al.: SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience 2012 1:18.

Short description: Single-genome assembler tuned for metagenomics.

Long description: SOAPdenovo is a novel short-read assembly method that can build a de novo draft assembly for the human-sized genomes. The program is specially designed to assemble Illumina GA short reads. It creates new opportunities for building reference sequences and carrying out accurate analyses of unexplored genomes in a cost effective way. Now the new version is available. SOAPdenovo2, which has the advantage of a new algorithm design that reduces memory consumption in graph construction, resolves more repeat regions in contig assembly, increases coverage and length in scaffold construction, improves gap closing, and optimizes for large genome. (taken from SOAPDenovo website)

Singularity use

2.4.2 gsAssembler (aka Newbler)

CyVerse App

Reference: Genivaldo, GZ; Silva, Bas E; Dutilh, David; Matthews, Keri; Elkins, Robert; Schmieder, Elizabeth A; Dinsdale, Robert A Edwards. "Combining de novo and reference-guided assembly with scaffold_builder". Source Code Biomed Central. 8 (23). doi:10.1186/1751-0473-8-23.

Short description: De novo assembly based on overlap-layout-consensus

Notes on use: 454 Life Sciences was purchased by Roche in 2007 and shut down in 2013. There haven't been any updates for the software since then, making it an increasingly aging tool.

Singularity use

2.4.3 SPAdes

CyVerse App

Reference: Bankevich A., Nurk S., Antipov D., Gurevich A., Dvorkin M., Kulikov A. S., Lesin V., Nikolenko S., Pham S., Prjibelski A., Pyshkin A., Sirotkin A., Vyahhi N., Tesler G., Alekseyev M. A., Pevzner P. A. SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. Journal of Computational Biology, 2012

Short description: SPAdes – St. Petersburg genome assembler – is an assembly toolkit containing various assembly pipelines

Notes on use: SPAdes, as with many de Bruijn assemblers, can consume incredibly amounts of memory. In the context of viral metagenomics, it's been known to use 2-3, and upwards of 6 TB of memory (and more if you give it more data!). There are multiple implementations on CyVerse using different runtimes and memory allocations. However, if the job will take more than 48-hr to run, there's a good chance it'll fail on CyVerse. For this, you may want to install it on a big memory machine locally.

Singularity use

2.4.4 IDBA-UD

CyVerse App

Reference: Peng, Y., et al. (2010) IDBA- A Practical Iterative de Bruijn Graph De Novo Assembler. RECOMB. Lisbon.

Peng, Y., et al. (2012) IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth, Bioinformatics, 28, 1420-1428.

Short description: IDBA-UD is a iterative De Bruijn Graph De Novo Assembler for Short Reads Sequencing data with Highly Uneven Sequencing Depth. It is an extension of IDBA algorithm.

Long description: IDBA-UD is a iterative De Bruijn Graph De Novo Assembler for Short Reads Sequencing data with Highly Uneven Sequencing Depth. It is an extension of IDBA algorithm. IDBA-UD also iterates from small k to a large k. In each iteration, short and low-depth contigs are removed iteratively with cutoff threshold from low to high to reduce the errors in low-depth and high-depth regions. Paired-end reads are aligned to contigs and assembled locally to generate some missing k-mers in low-depth regions. With these technologies, IDBA-UD can iterate k value of de Bruijn graph to a very large value with less gaps and less branches to form long contigs in both low-depth and high-depth regions. (taken from website)

Singularity use

2.4.5 Trinity

CyVerse App

Reference: Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, Adiconis X, Fan L, Raychowdhury R, Zeng Q, Chen Z, Mauceli E, Hacohen N, Gnirke A, Rhind N, di Palma F, Birren BW, Nusbaum C, Lindblad-Toh K, Friedman N, Regev A. Full-length transcriptome assembly from RNA-seq data without a reference genome. Nat Biotechnol. 2011 May 15;29(7):644-52. doi: 10.1038/nbt.1883. PubMed PMID: 21572440.

Short description: Trinity assembles transcript sequences from Illumina RNA-Seq data.

Singularity use

2.5 Annotations, Sequence Analysis

2.5.1 Prokka

CyVerse App

Reference: Seemann T. Prokka: rapid prokaryotic genome annotation Bioinformatics 2014 Jul 15;30(14):2068-9. PMID:24642063

Short description: Prokka is a software tool to annotate bacterial, archaeal and viral genomes quickly and produce standards-compliant output files

Singularity use

2.5.2 Diamond

CyVerse App

Reference: B. Buchfink, Xie C., D. Huson, "Fast and sensitive protein alignment using DIAMOND", Nature Methods 12, 59-60 (2015)

Short description: DIAMOND is a sequence aligner for protein and translated DNA searches, designed for high performance analysis of big sequence data.

Singularity use

2.6 Viral Analysis

Analyzing viral data remains a major challenge in the field of viral ecology. A variety of approaches have been proposed, each dependent on the source of data and the underlying biological question. A relatively recent method of analyzing complex viral data is by organizing viral sequence space, often through the use of protein clustering techniques. Protein clusters can be used as a diversity metric, or as units for ecological studies when compared against other datasets, or functional profiling of the community.

2.6.1 PCPipe

CyVerse App

Reference:

Short description: Protein clustering pipeline and annotation

Singularity use

2.6.2 VIRSorter

CyVerse App

Reference: Roux S, Enault F, Hurwitz BL, Sullivan MB. (2015) VirSorter: mining viral signal from microbial genomic data. PeerJ 3:e985 https://doi.org/10.7717/peerj.985

Short description: Identify viral contigs in a microbial metagenomes

Singularity use

2.6.3 vConTACT

CyVerse App

Reference: Bolduc B, Jang H Bin, Doulcier G, You Z, Roux S, Sullivan MB. (2017). vConTACT: an iVirus tool to classify double-stranded DNA viruses that infect Archaea and Bacteria. PeerJ 5: e3243.

Short description: Guilt-by-contig-association automatic classification of viral contigs

Singularity use

2.6.4 vConTACT-PCs

CyVerse App

Reference:

Short description: Generate PC-profiles using vContact/MCL

Singularity use

2.6.5 vConTACT-Gene2Genome (formerly known as "Gene2Contig")

CyVerse App

Reference:

Short description: Conditions files for use in vContact

Singularity use:

2.6.6 BowtieBatch

CyVerse App

Reference:

Short description: Performs mass alignment of paired and unpaired reads against a reference dataset using Bowtie2 and Samtools.

Singularity use:

2.6.7 Read2RefMapper

CyVerse App

Reference:

Short description: Consumes input from BowtieBatch to generate coverage profiles.

Singularity use:

2.7 In some stage of development

Below are a list of apps that could be at any stage of the app development process. That means they could be 99% implemented and moments away from going public, or they could be a note taken on a napkin.

2.7.1 GAAS (Genome Abundance and Average Size)

Estimates relative abundance and average size of metagenomic sequences

2.7.2 Circonspect

Generates contig spectra for downstream modeling of community structure

2.7.3 PHACCS (Control In Research on CONtig SPECTra)

Estimates structure and diversity of viral communities

2.7.4 MARVEL

MARVEL is a pipeline for recovery of complete phage genomes from whole community shotgun metagenomic sequencing data.

CHAPTER 3

Protocols

Below is a collection of protocols.io links for using iVirus-enabled apps in CyVerse. Due to the nature of this media, it isn't possible to include guides with the same level of "visual" detail. This only seeks to centralize them in one location.

A few quick notes:

- Guides are not intended to assist users in understanding the biology behind the tools nor how the tools function.
- Where possible, Apps have links to their documentation on CyVerse as well as their citations (or original home pages).
- In some cases, many Apps are available to solve a particular problem. Guides will choose to highlight one or two.
- These guides assume you've created an CyVerse account and can access your account. Check out the getting started guide for assistance.

3.1 Guides and Use Cases

Several "use cases" are available at protocols.io. For nearly all these use cases, we'll use (as a basis) actual reads from the Ocean Sampling Day (2014) and process them using Cyverse. In some cases we'll take the user from using raw read files to assembly to identifying viral sequences and preliminary analysis. Other use cases will tackle ways of analyzing a viral metagenome, either reads or contigs, using traditional and non-traditional approaches. As a reminder, all these protocols are on protcols.io and should be considered the most up-to-date versions, though *they definitely can fall behind depending on developer's time*.

All example files can be found within the Cyverse datastore. To find these files, login to the Discovery Environment. Under "Data", go to Community Data –> iVirus –> ExampleData. Alternatively, you can copy-and-paste the following into the "Viewing" bar under the data browser: /iplant/home/shared/iVirus/ExampleData/

All tools have "Input" and "Output" directories, so not only does the user have valid input data, but also the expected output data as well.

3.2 Processing a Viral Metagenome

Description: A long-standing challenge in viral metagenomics is actually processing a viral metagenome (we're not talking about the science side!). For many reasons enumerated elsewhere, processing these datasets requires skilled bioinformaticians and computational resources not available to many researchers/labs. iVirus seeks to tackle this head-on.

Protocol "collection": This collection connects individual protocols and goes from raw reads to processing with vConTACT.

Individual steps:

- Cleaning up sequencing reads using Trimmomatic
- Assembling QC'd reads using SPAdes
- Identifying putative viral sequences using VirSorter
- Preparing data for vConTACT
- (New 2018-12-19) Preparing data for vConTACT2
- Running vConTACT and visualization in Cytoscape
- (New 2018-12-19) Running vConTACT2 and visualization in Cytoscape

3.3 Mapping Metagenomic Reads to a Reference Collection

Description: One of the most commonly used procedures for analyzing viral metagenomic data is to map their reads (or reads from another dataset) against a set of references, often those from the read assembly. For example, if one wanted to know how well-represented viruses in NCBI's Viral Reference Sequences (ViralRefSeq) were in ocean viromes, they could map reads from lots of ocean viral metagenomes against ViralRefSeq. This is generally done using Bowtie2 or BWA, by selecting a reference set of sequences, and then providing paired or unpaired reads to Bowtie2/BWA. Then the results must be processed/filtered to generate coverage tables. Dealing with setting up multiple reads files (10 paired metagenomes = 10 alignment runs) and the processing those read files can be challenging (not to mention computational resources).

Protocol: Mapping reads

Individual steps:

- Mapping reads from multiple metagenomes to a set of references
- Filtering mapped reads and generate coverage tables

CHAPTER 4

Singularity 101

4.1 Singularity Notes

Much of this documentation is taken from the singularity website, it's simply re-hashed and organized in a less "quick start" or "overwhelming documentation" style.

Overall thoughts

Singularity is a container-based environment. There's always TWO "systems", the first is the *host* system, that's the one where the container/image is running. The 2nd is the *guest* system, that's whatever's **in** the container. The cool thing about Singularity is that the guest can "see" outside itself (depending on permissions, more below), meaning that it can run on files outside the container. In fact, once a container is built, it can *almost* be treated just like any other executable.

What does this really mean?

The big advantage of Singularity containers is that they are generally more secure than other container-based options. In a nutshell, Singularity containers work with the user's privileges. *So if the user wants to create/edit operating system stuff, then they need to be sudo for creating/editing the container.* If the user just wants to run a command (for ex: ls, top, grep, cat) then the user doesn't need to be sudo and they can *run the container*. This becomes **a really important distinction when you move from creating a Singularity image to running it in a production environment**.

4.2 Before you begin

Assuming you want to **develop** Singularity-based container apps. . .

You need to install Singularity to a machine where you have root/sudo privileges (more below). There's a number of ways to do this: Mac, Linux and Windows

I use a Mac, so installation is a bit more involved. It requires using vagrant to set up and build a VM, then connecting to that [linux-based] VM to install Singularity. It's not bad, but I need to keep remembering where I am:

Mac -> VM -> Singularity-Container

and there's a shared space to transfer files from Mac<–>VM, so it's all about shuttling files from Mac to the shared space, then from the shared space to the VM.

If you're using linux you're home free. Singularity is just another tool you can install on the command line.

4.3 Typical Workflow

High-level summary:

- [On local machine] Create Singularity definition file
- [On local machine] Create Singularity image from definition file
- Transfer Singularity image to remote machine / HPC
- [On HPC] Run/Execute Singularity image as local user

On local machine

Keep in mind that most everything is done on a local machine, i.e. a machine where you have root/sudo privileges.

4.4 Create Singularity definition file

The definition file is functionally equivalent to Dockerfiles. It contains all the commands that one would need in order to process dependencies, build and compile tools. Basically, if you'd need to type out 30 commands to get your favorite tool to compile with its million libraries, you'll need to copy-and-paste those in the definition file.

The basic structure is as such:

```
BootStrap: debootstrap
OSVersion: stable
MirrorURL: http://ftp.us.debian.org/debian/

# Where files go to transfer TO app once everything is finished
%files

# Whatever other labels you want added to container
%labels
MAINTAINER ben

%environment
COOL_LIB=/code/to/library
export COOL_LIB

%runscript
echo "I'm in the container!"
exec /code/to/tool.py "$@"

# This is what is run to build and create container
%post
apt-get update
apt-get install dependency1 dependency2 library1

git clone git.repo
cd git.repo
./configure
```

(continues on next page)

(continued from previous page)

```
make
make install
```

Obviously the above won't run. I don't really use %labels, and since %files is run AFTER %post, I don't need it (I usually pull data from the web using wget, curl or similar). You could use %files to copy a database to a specific location in the container. I just prefer to rely on a single definition file with no calls to non-public data, as to avoid *the black box effect*. %environment is fine, but for some reason it doesn't work as I expect it to during %post, so I usually export my variables (i.e. export $BINPATH=/usr/locl/bin) within %post when installing stuff.

4.5 Build Singularity image with the definitions file

```
sudo singularity build app.img app.def
```

And you'll want to check out the various ways to run it

```
singularity run app.img --help
singularity exec app.img /path/to/tool --help
./app.img --help
```

All the above are equivalent and function identically. Of course, there's probably some subtle differences between them, but that's a bit beyond me.

At this point you want to copy the app.img file over to the HPC where it's needed.

On the Ohio Supercomputer Center (OSC)

Load module

```
module load singularity/current
```

At this point you're ready to run the container.

```
singularity --debug run -H /users/<account#>/<userid> app.img -h
```

Strangely enough, it appears that the singularity homedir needs to be bound by the user.

4.6 Bugs?

Sometimes you can get a binding error. It can be fixed by doing:

```
sudo singularity shell --writable app.img
```

on a local/development machine and then exiting. Stupid, I know, but it's minor annoyance and it works.

4.7 Links

Singularity Hub

CHAPTER 5

Building-A-Pipeline

5.1 From Containers to Pipelines

This guide serves as a reference for taking Singularity-based containers and publishing them to CyVerse. This guide is just that, a guide. Every effort has been made to keep this *somewhat* up to date with the best practices of CyVerse. If something doesn't work, let us know and we'll do our best to update it. Also, this won't teach you everything about the Agave API. There's a lot of information about the setup and features, but we only use A FEW of those tools.

5.2 Before you begin

You'll need a few tools installed, and access to a few systems (one of which requires admin powers).

- Singularity installed on a machine w/ admin/sudo powers
- The Agave CLI installed on either your local machine or TACC

Personally I use TACC because it has fast access to CyVerse's servers and it's an excellent testing environment. If it works on TACC's systems (as a test job, see more below), there's a good chance it'll publish without problem.

(Documentation will one day be updated to go through all the steps to install the CLI, but it'll be copy-and-pasted from the installation guide. . . so go there for installation help)

The guide below is going to use vConTACT2-0.9.3 (versions don't matter all that much), which can be found as vConTACT2.def under the singularity directory.

5.3 Step 1 - On your local machine

Building the container

Build the singularity image and ensure that it functions correctly on your local machine. Since I use a Mac, I also use Vagrant to manage my Docker containers that I then connect to.

```
cd <location with vagrant file>
vagrant up && vagrant ssh
```

This then connects to the running docker container. (Yes, this can be a little confusing. You're using vagrant to help manage Docker, connecting to that container, and then using that container to make another container)

5.4 Step 2 - In the docker container

```
cp /vagrant/vConTACT2-0.9.3.def .
sudo singularity build vConTACT2-0.9.3.simg vConTACT2-0.9.3.def
```

Here I copied the singularity definition file *into* the running Docker container and then built the Singularity container. Alternatively, you can also build directly using the definitions file w/out copying the file into the container.

```
cp /vagrant/gene2genome_proteins.csv .
cp /vagrant/VIRSorter_viral_prots.faa .
singularity run vConTACT2-0.9.3.simg --raw-proteins VIRSorter_viral_prots.faa --rel-
↪mode 'Diamond' --proteins-fp gene2genome_proteins.csv --db
↪'ProkaryoticViralRefSeq85-Merged' --pcs-mode MCL --vcs-mode ClusterONE --c1-bin /
↪usr/local/bin/cluster_one-1.0.jar --output-dir vConTACT2-Output
```

Copy the test data over into the Docker container and run the singularity container using the test files. **Ensure that all vConTACT2 output files are generated.** This is essential to the testing process.

Once that's done, copy the files back to the host system (for me, it's the Mac).

```
cp vConTACT2-0.9.3.simg /vagrant/
```

5.5 Step 3 - On your local machine

Now copy the Singularity image over to TACC, as well as the files required for a functional app on CyVerse.

```
rsync -Pavz <path-to-apps>/vConTACT2-0.9.3 username@stampede2.tacc.utexas.edu:<path-
↪to-work-directory>/iVirus-Apps
```

5.6 Step 4 - On TACC

At TACC, you should have installed the Agave CLI (above) and validated that everything was functional. (More details might be added later).

5.6.1 Test that the singularity image works

Yes, sounds obvious. But *it is possible* that it won't work. Often times this is because you need to make sure that the directories /home1 /scratch and /work are created. And there are rare occasions where interactions between the tool, singularity, and the host system don't mesh (i.e. for "some" reason, the tool doesn't recognize files outside the container).

```
sbatch test.sh
```

The job should get submitted, and output generated. Hopefully it's the same output as was created on your local machine. If not, investigate!

5.6.2 "Push" the app to your private system

Once the Singularity container works, push it to CyVerse as a private app on your private system.

```
auth-tokens-refresh -S -v

files-upload -S data.iplantcollaborative.org -F <app-folder-on-TACC> <CyVerse-home-
↪directory>/apps

apps-addupdate -F <app-folder-on-TACC>/vConTACT2-0.9.3.json

apps-list --privateonly
```

A lot of things happen here. First, we're refreshing our CyVerse token (explained extensively in the Agave CLI docs). Then we're uploading the app data (*everything* that's needed to run the app) to CyVerse. The storage system is iplantcollaborative, the folder is the app folder on TACC, and it's being uploaded to your CyVerse home directory under the "apps" folder. Of course, this could be anywhere you have write permission to on CyVerse, but for now that's the easiest place. Then, you add the app to CyVerse's app system. The json has all the details about the app's run parameters, what should be displayed to the user through the CyVerse UI, version info, and where to find the app. Finally, ensure that the app has been successfully published to your private system (apps-addupdate will tell you if something really goes wrong) by getting a list of your private apps. Whatever your app name + version is should be displayed.

5.6.3 Test the CyVerse app

```
jobs-submit -v -F vConTACT2-0.9.3-TestJob.json
```

Now the job should get submitted using the parameters set in the TestJob.json file to make sure that the app works on CyVerse. (Things *can* and *do* go wrong here. Even if everything works up to this point, variables in the app parameters can be wrong/incorrect, or a mispelled argument doesn't get passed correctly to the wrapper. A whole bunch of stuff can go wrong. It's **here** that the final test of the app is done.)

And finally, if your job takes a little while but you want to get the status of the job…

```
jobs-status JOB_ID
```

5.6.4 If everything works…

Go ahead and request that your app be made public from the CyVerse staff.

Congratulations, you've gone from a Singularity definitions file to a published app that others can benefit!

CHAPTER 6

References:

- **iVirus: Facilitating new insights in viral ecology with software and community data sets imbedded in a cyberinfrastructure**. (2017) Bolduc, B., Youens-Clark, K., Roux, S., Hurwitz, B.L., and Sullivan, M.B. ISME Journal

CHAPTER 7

Funding:

We'd be remiss if we didn't mention the support of awards that make iVirus possible. iVirus wouldn't be possible without the following grants: Gordon and Betty Moore Foundation Investigator Award (#3790), and since Sept 2018 an NSF Advances in Biological Infrastructure Award (#1759874).

The NSF award page is here

Thanks so much for making this work possible!

CHAPTER 8

Indices and tables

- genindex
- modindex
- search