Genome Informatics 2020

Lesson 2 - Portable and reproducible bioinformatic analysis

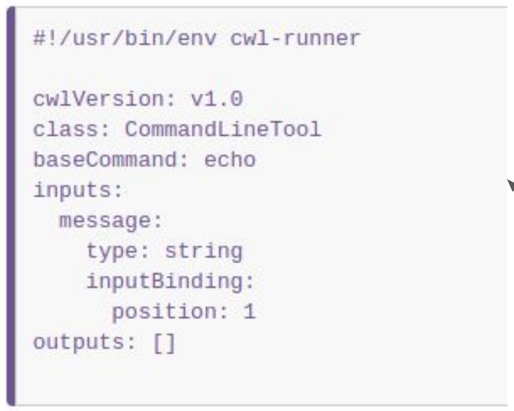
Lesson overview

1. Common Workflow Language (CWL). Building apps (tools and workflows
2. Docker
3. Constructing and running portable and reproducible bioinformatics analysis
4. Jupyter Notebook bioinformatic analysis on the cloud

Common Workflow Language

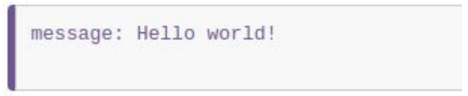
- CWL is a way to describe command line tools execution
- Every tool has defined set of inputs and outputs
- Every tool is executed in its own environment (Docker)
- Execution on the cloud or local environment
- Enables portable and reproducible execution

1st-tool.cwl



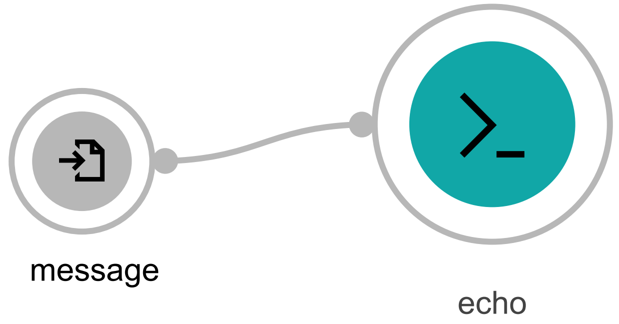
```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: CommandLineTool
baseCommand: echo
inputs:
  message:
    type: string
    inputBinding:
      position: 1
outputs: []
```

echo-job.yml

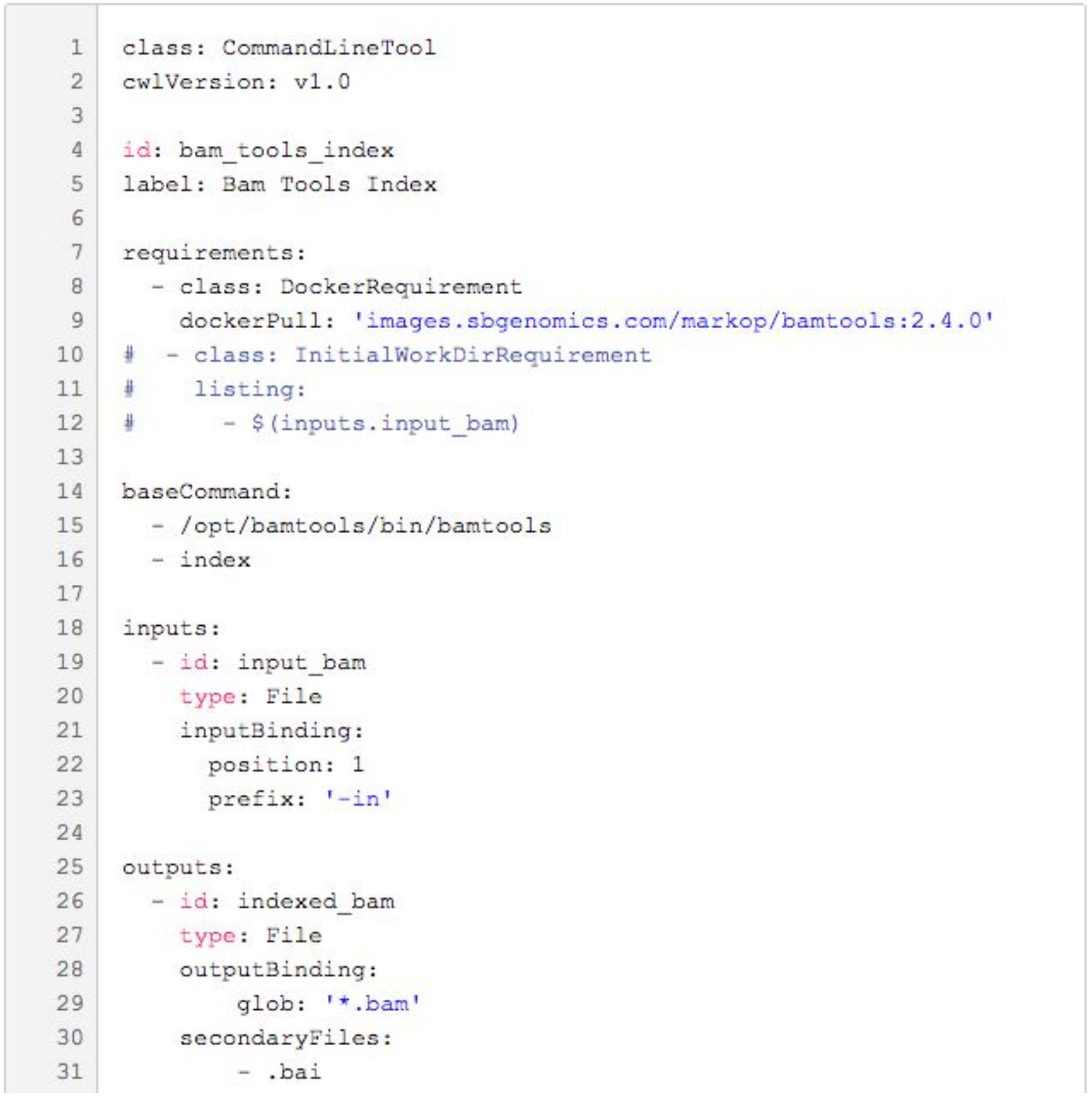


```
message: Hello world!
```

Used by CWL executor



CWL: Simple instructions for reproducible analyses



```
class: CommandLineTool
cwlVersion: v1.0

id: bam_tools_index
label: Bam Tools Index

requirements:
  - class: DockerRequirement
    dockerPull: 'images.sbgenomics.com/markop/bamtools:2.4.0'
#  - class: InitialWorkDirRequirement
#    listing:
#      - $(inputs.input_bam)

baseCommand:
  - /opt/bamtools/bin/bamtools
  - index

inputs:
  - id: input_bam
    type: File
    inputBinding:
      position: 1
      prefix: '-in'

outputs:
  - id: indexed_bam
    type: File
    outputBinding:
        glob: '*.bam'
    secondaryFiles:
        - .bai
```

Text in YAML or JSON format.

Describes the tools and workflows.

Easier and faster to deploy tools

Wide adoption by 40+ institutes/research groups

Avoids lock-in to a given system

produces the command line

```
/opt/bamtools/bin/bamtools index -in input_bam.
```

How do I learn CWL?

You can learn the syntax: **CWL User Guide**

BUT you don't have to!

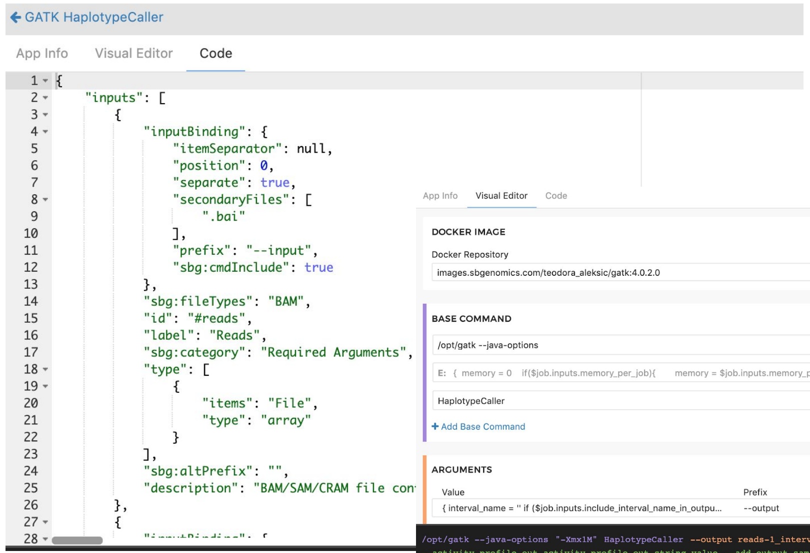
With the Seven Bridges **Software Development Kit** (Tools/Workflow Editor & Rabix Composer), you can easily create tools and chain them into workflows interactively and without any programming experience.

The Seven Bridges SDK will create the CWL code for you
so you can get your tool up and running on the platform more quickly and easily.

Rabix Composer

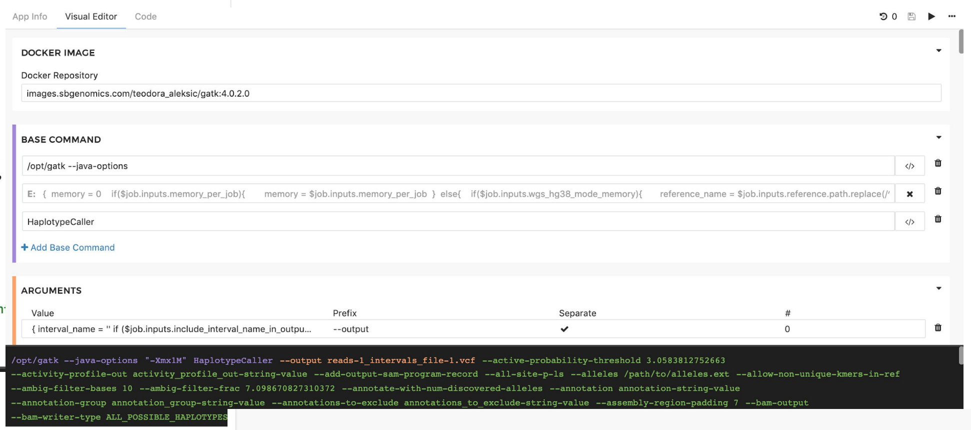
An Integrated Development Environment for CWL developers

- **Compatible** with different versions of CWL
- **Version history**
- **Graphical editors**
- **In-line documentation**
- Support for popular **scripting** languages
- **Desktop Version** - local testing
- **Web Composer**



```
{
    "inputs": [
        {
            "inputBinding": {
                "itemSeparator": null,
                "position": 0,
                "separate": true,
                "secondaryFiles": [
                    ".bai"
                ],
                "prefix": "--input",
                "sbg:cmdInclude": true
            },
            "sbg:fileTypes": "BAM",
            "id": "#reads",
            "label": "Reads",
            "sbg:category": "Required Arguments",
            "type": [
                {
                    "items": "File",
                    "type": "array"
                }
            ],
            "sbg:altPrefix": "",
            "description": "BAM/SAM/CRAM file con
        },
        {
```

CWL Text Editor



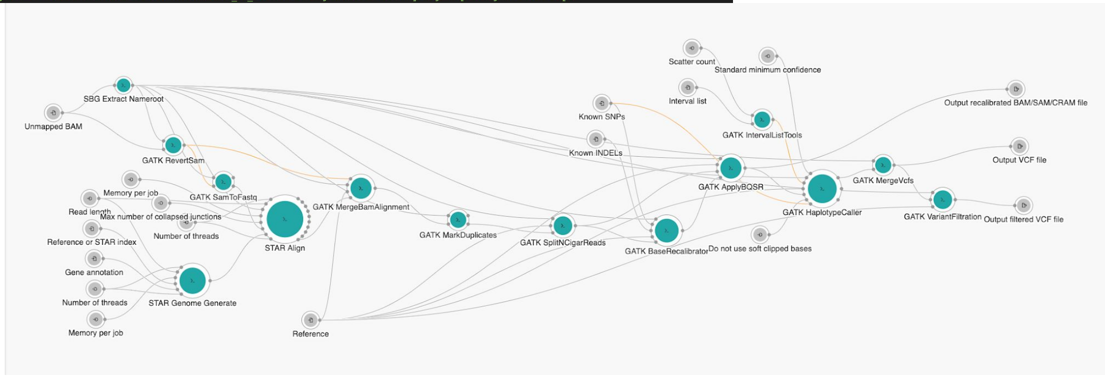
Docker Repository: images.sbgenomics.com/teodora_aleksic/gatk:4.0.2.0

```
/opt/gatk --java-options
E:  {  memory = 0    if($job.inputs.memory_per_job){      memory = $job.inputs.memory_per_job  }  else{    if($job.inputs.wgs_hg38_mode_memory){      reference_name = $job.inputs.reference.path.replace(/'
HaplotypeCaller
```

Value	Prefix	Separate	#
{ interval_name = '' if ($job.inputs.include_interval_name_in_outpu...	--output	✓	0

```
/opt/gatk --java-options "-Xmx1M" HaplotypeCaller --output reads-1_intervals_file-1.vcf --active-probability-threshold 3.0583812752663
--activity-profile-out activity_profile_out-string-value --add-output-sam-program-record --all-site-p-ls --alleles /path/to/alleles.ext --allow-non-unique-kmers-in-ref
--ambig-filter-bases 10 --ambig-filter-frac 7.098670827310372 --annotate-with-num-discovered-alleles --annotation annotation-string-value
--annotation-group annotation_group-string-value --annotations-to-exclude annotations_to_exclude-string-value --assembly-region-padding 7 --bam-output
--bam-writer-type ALL_POSSIBLE_HAPLOTYPES
```

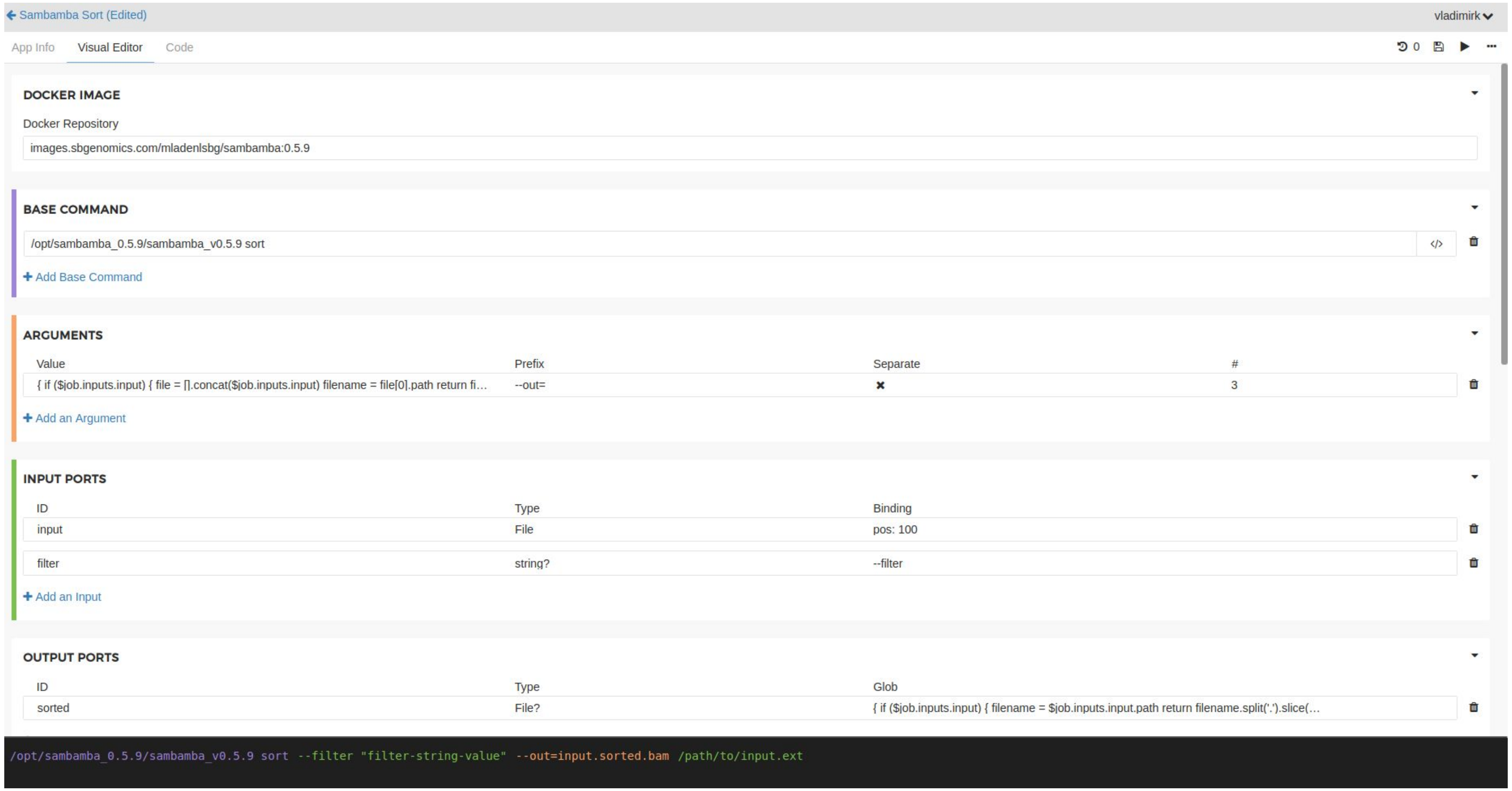
Tool Editor



Workflow Editor

Rabix Composer

An Integrated Development Environment for CWL developers



Sambamba Sort (Edited)

vladimirk

App Info | Visual Editor | Code

DOCKER IMAGE

Docker Repository

images.sbgenomics.com/mladenlsbg/sambamba:0.5.9

BASE COMMAND

/opt/sambamba_0.5.9/sambamba_v0.5.9 sort

+ Add Base Command

ARGUMENTS

Value	Prefix	Separate	#
{ if ($job.inputs.input) { file = [].concat($job.inputs.input) filename = file[0].path return fi…	--out=	✖	3

+ Add an Argument

INPUT PORTS

ID	Type	Binding
input	File	pos: 100
filter	string?	--filter

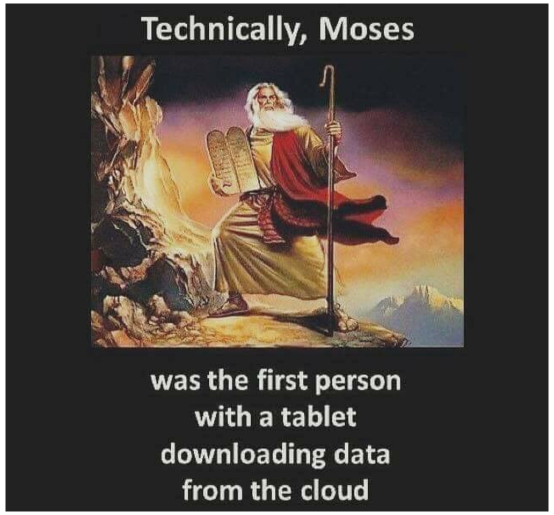
+ Add an Input

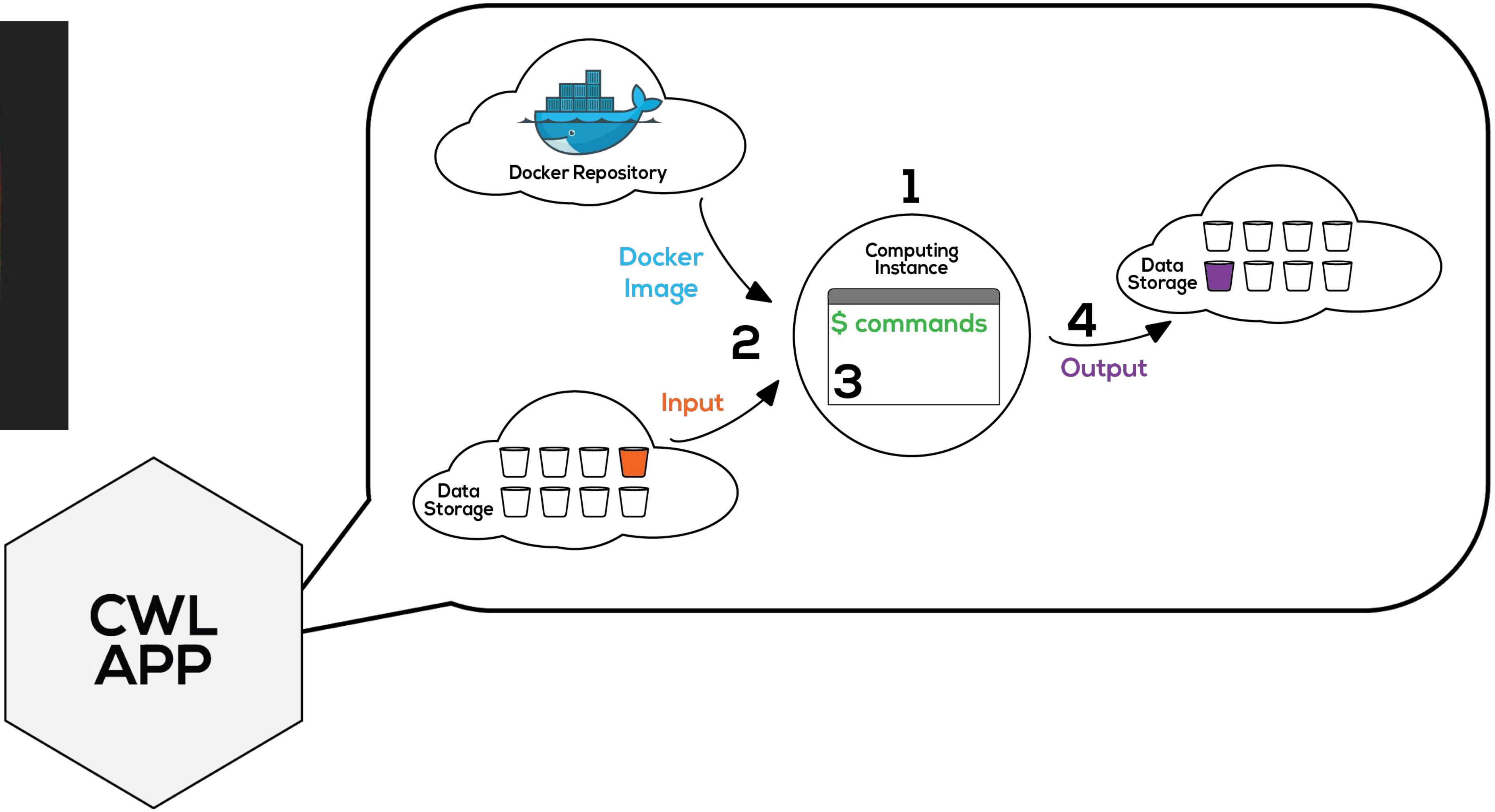
OUTPUT PORTS

ID	Type	Glob
sorted	File?	{ if ($job.inputs.input) { filename = $job.inputs.input.path return filename.split('.').slice(…

```
/opt/sambamba_0.5.9/sambamba_v0.5.9 sort --filter "filter-string-value" --out=input.sorted.bam /path/to/input.ext
```

CWL @ Cloud

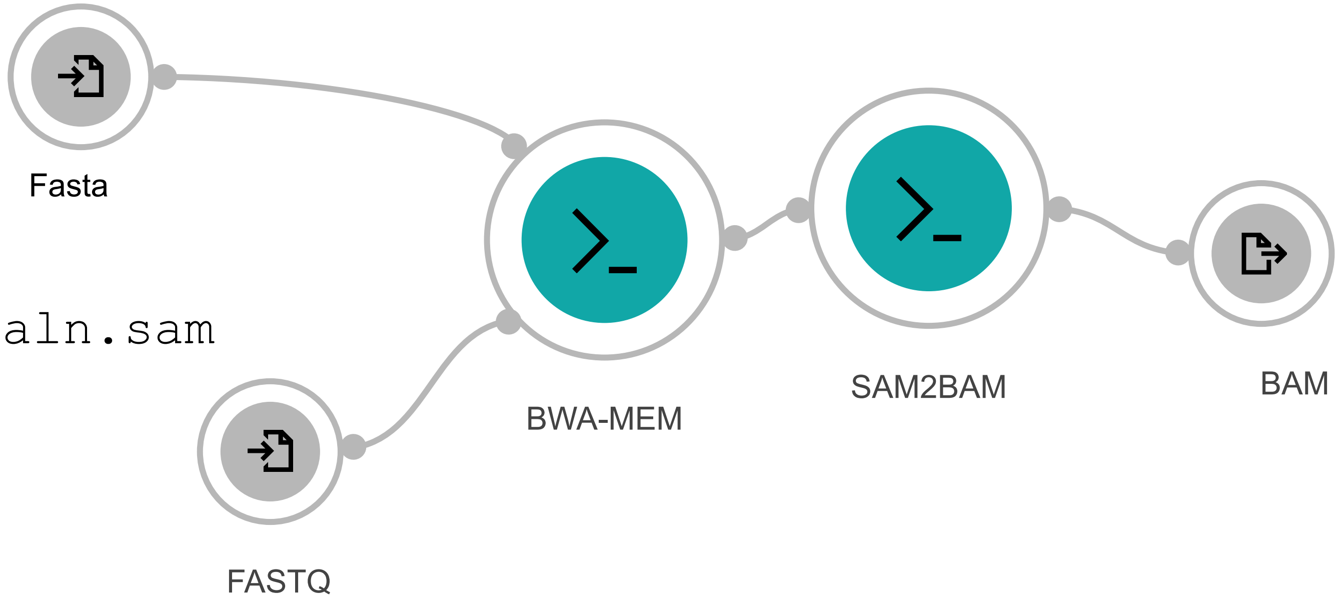

Technically, Moses
was the first person
with a tablet
downloading data
from the cloud


Docker Repository
Docker Image
1
Computing Instance
$ commands
3
2
Input
Data Storage
4
Output
Data Storage
CWL APP

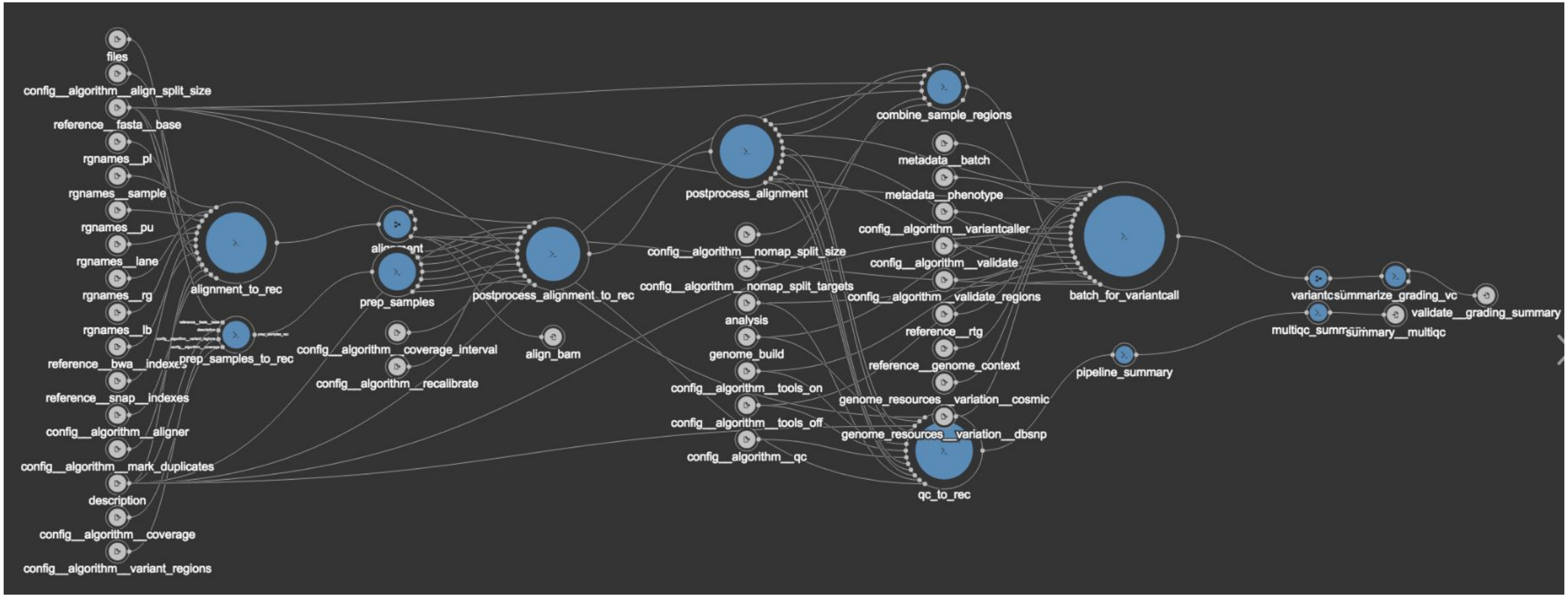
What is a CWL workflow?

- Acyclic graph of tools connected to perform some analysis
- Workflow's nodes are:
  - Inputs (file or parameter)
  - Tools
  - Outputs
  - Workflow

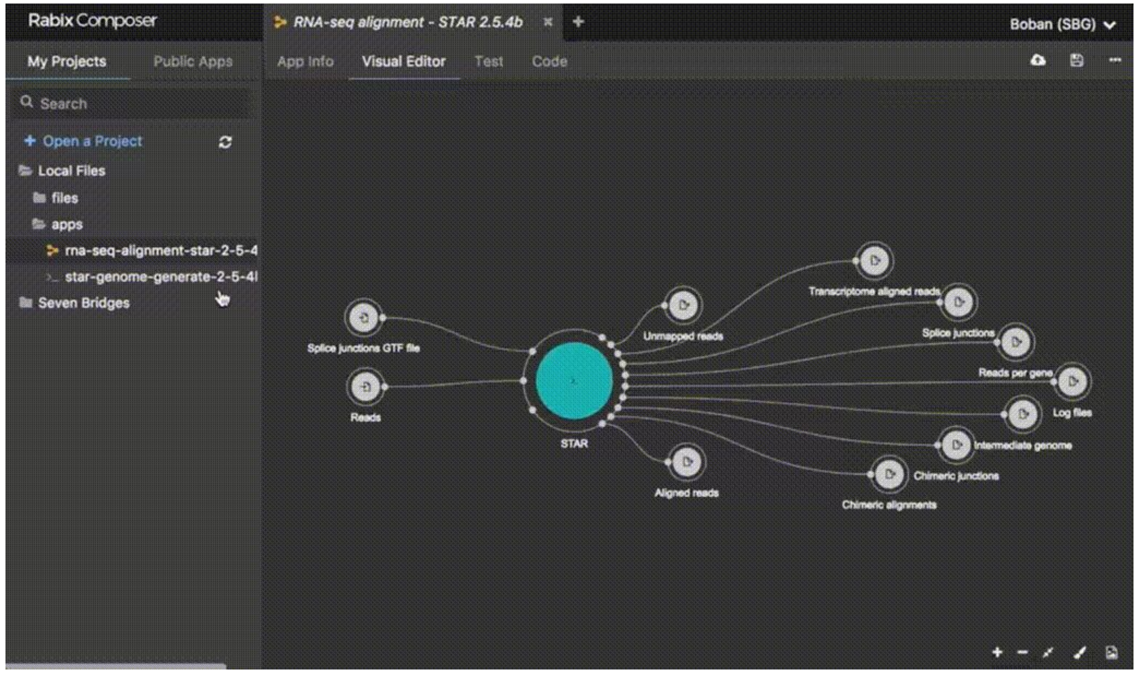
```
bwa mem ref.fa read1.fq read2.fq > aln.sam
sam2bam aln.sam > aln.bam
```

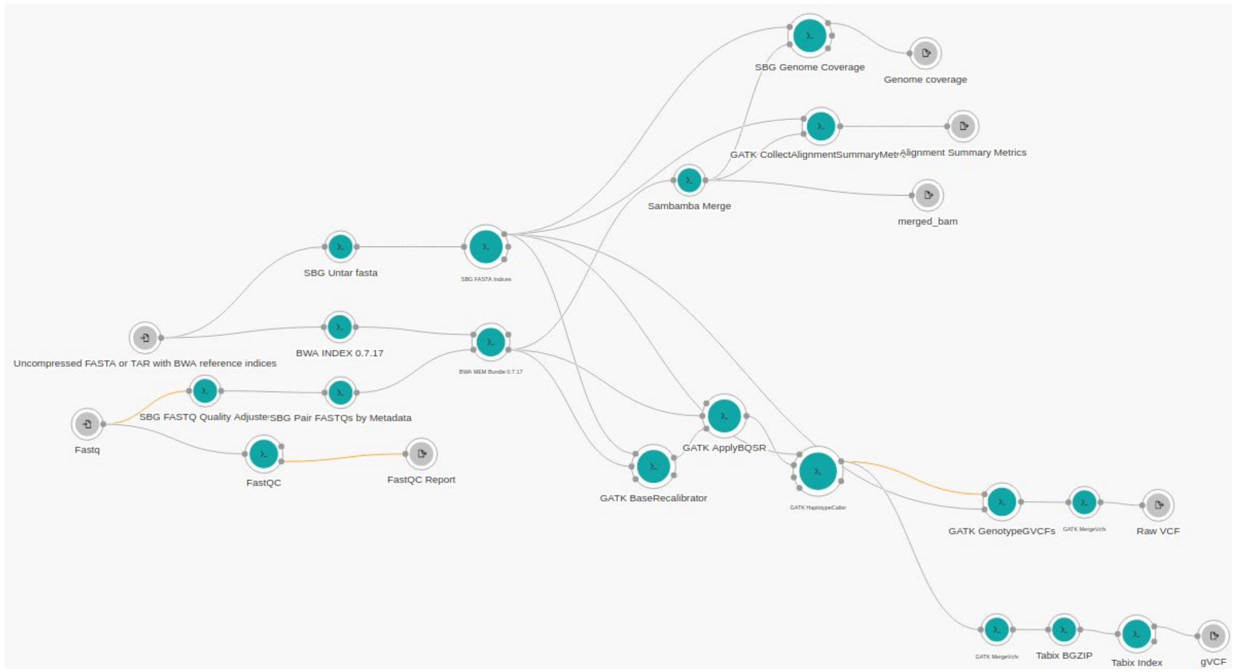


Why we need a workflow?


files
config__algorithm__align_split_size
reference__fasta__base
rgnames__pl
rgnames__sample
rgnames__pu
rgnames__lane
rgnames__rg
rgnames__lb
reference__bwa__indexes
reference__snap__indexes
config__algorithm__aligner
config__algorithm__mark_duplicates
description
config__algorithm__coverage
config__algorithm__variant_regions
alignment_to_rec
prep_samples_to_rec
alignment
prep_samples
config__algorithm__coverage_interval
config__algorithm__recalibrate
postprocess_alignment_to_rec
align_bam
postprocess_alignment
config__algorithm__nomap_split_size
config__algorithm__nomap_split_targets
analysis
genome_build
config__algorithm__tools_on
config__algorithm__tools_off
config__algorithm__qc
combine_sample_regions
metadata__batch
metadata__phenotype
config__algorithm__variantcaller
config__algorithm__validate
config__algorithm__validate_regions
reference__rtg
reference__genome_context
genome_resources__variation__cosmic
genome_resources__variation__dbsnp
qc_to_rec
batch_for_variantcall
pipeline_summary
variantcall
summarize_grading_vc
validate__grading_summary
multiqc_summary
summary__multiqc

How to build a workflow?





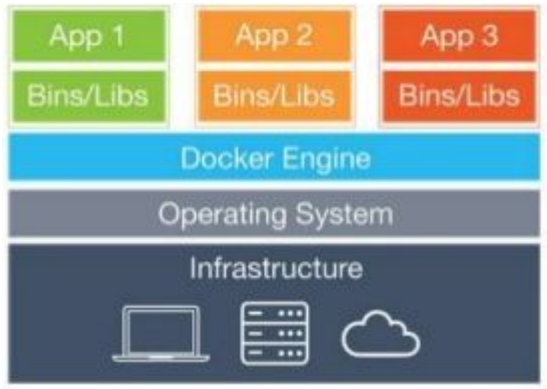
[Desktop CWL composer](#)

[Web CWL composer](#)

2. Docker

What is Docker?

- Docker is a light-weight virtual environment
- Allows you to package the tool (e.g. Python script or some C program) with all of its dependencies into the standardized unit for software development
- Docker containers run on any computer, on any infrastructure
- Layered container structure
- Can directly access resources of host operating system

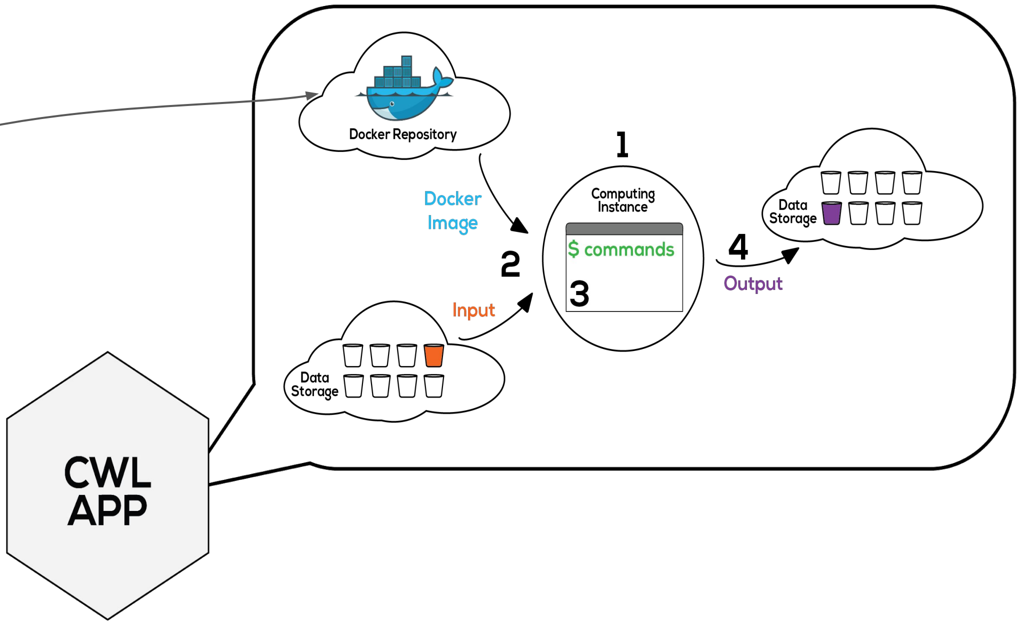


How to create Docker image?

```
FROM ubuntu:16.04
MAINTAINER vladimir.kovacevic@sbgenomics.com
RUN apt-get update && apt-get install -y wget \
make \
gcc \
zlib1g-dev
WORKDIR /opt
RUN wget
https://github.com/bwa/releases/bwa-0.7.15.tar.bz
2
RUN tar xfj bwa-0.7.15.tar.bz2
WORKDIR /opt/bwa-0.7.15
RUN make
COPY Dockerfile /opt/Dockerfile
```

Docker
file

```
# Build image from Dockerfile and push to docker repo

docker build -t images.sbgenomics.com/vladimirk/bwa:0.7.15 .

docker push images.sbgenomics.com/vladimirk/bwa:0.7.15
```



Best practice: Using a Dockerfile

A Dockerfile is a text file that stores commands to create a Docker image

- Uses a domain-specific language to describe how to build an image
- The Docker tool automates the building of an image from a Dockerfile
- Docker reads commands and executes in succession

Benefits:

- Stores whole procedure of image creation
- Helps facilitate and automate the process of maintaining tools that are wrapped for the platform
  - Automate builds
  - Can be used as the source of documentation at failure points and can restart failed builds
  - Transparency
  - Easy to share on GitHub/DockerHub

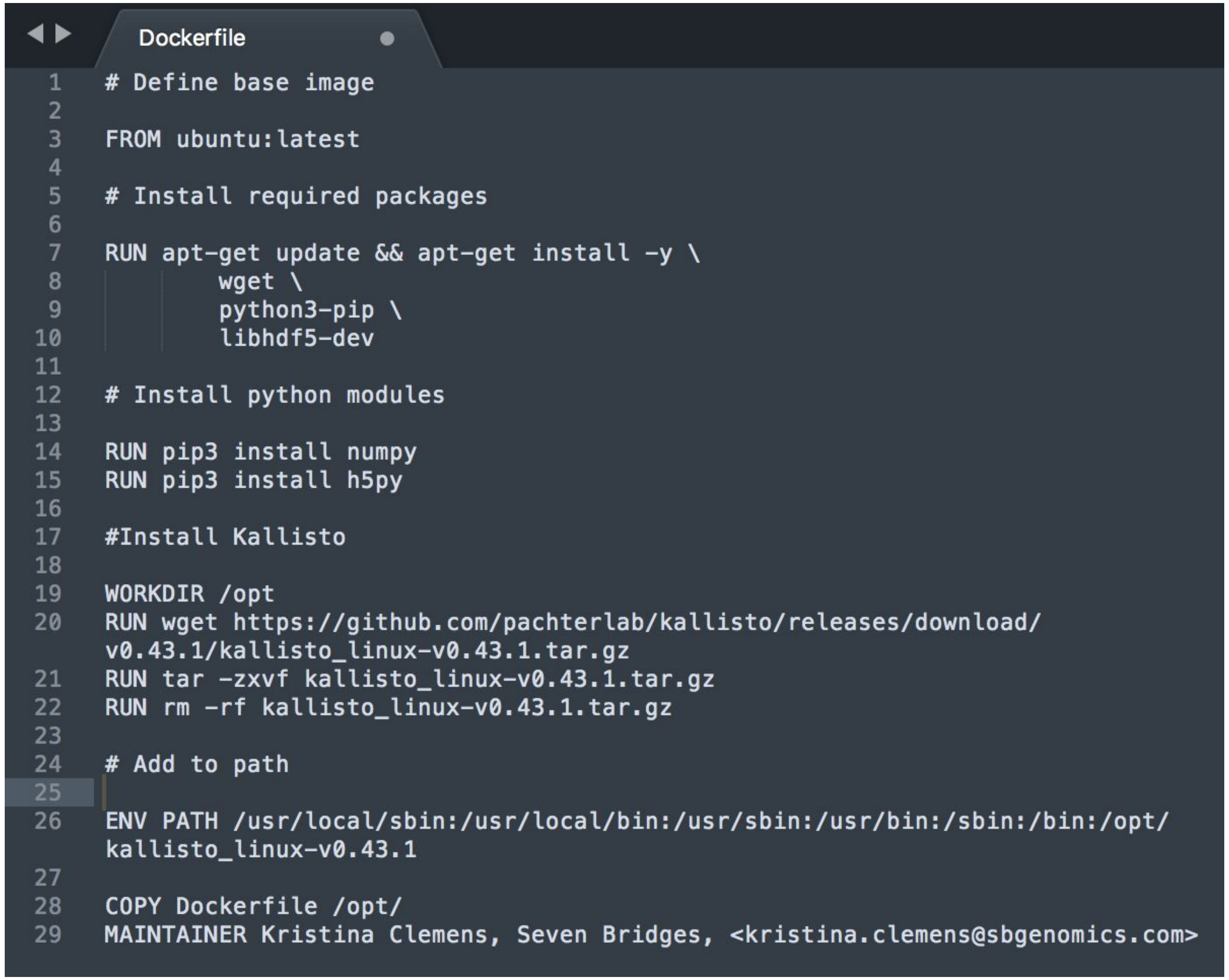
A Dockerfile consists of **Instructions** followed by **arguments** and comments:

```
#Comment
INSTRUCTION arguments
```

Dockerfile Instructions

FROM	● Initializes new build stage and sets Base Image (“pulling”)
RUN	● Executes the command of argument during build process ● Execution results are committed to current image and resulting image is used for next instruction ● Chain multiple commands with && and \ for a line break
CMD	● Provides default command, which is executed inside container when it’s created based on image ● Need to use argument [“/bin/bash”] , as that is how the container is invoked during task execution for SB Platform
ADD	● Used to copy files, directories, or remote file URLs from original location <source> to container destination path <destination> ● You can only specify those source paths that are within context directory
COPY	● Used to copy files or directories to container at specified path ● Unlike ADD, doesn’t take URL as <source> and will not unpack archived file as <source>
WORKDIR	● Used to set default working directory for container. Instructions will be executed in the defined working directory

Use a Dockerfile to build an image

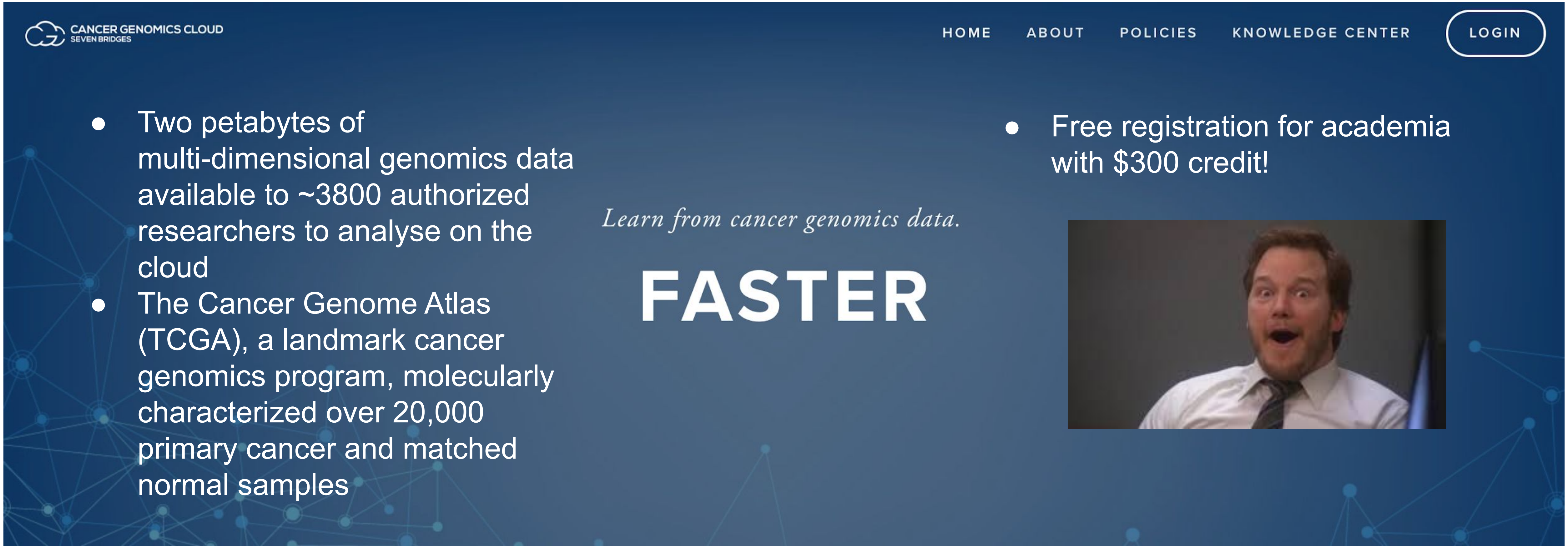


Dockerfile

```
# Define base image

FROM ubuntu:latest

# Install required packages

RUN apt-get update && apt-get install -y \
        wget \
        python3-pip \
        libhdf5-dev

# Install python modules

RUN pip3 install numpy
RUN pip3 install h5py

#Install Kallisto

WORKDIR /opt
RUN wget https://github.com/pachterlab/kallisto/releases/download/
v0.43.1/kallisto_linux-v0.43.1.tar.gz
RUN tar -zxvf kallisto_linux-v0.43.1.tar.gz
RUN rm -rf kallisto_linux-v0.43.1.tar.gz

# Add to path

ENV PATH /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/
kallisto_linux-v0.43.1

COPY Dockerfile /opt/
MAINTAINER Kristina Clemens, Seven Bridges, <kristina.clemens@sbgenomics.com>
```

3. Constructing and running portable and reproducible bioinformatics analysis

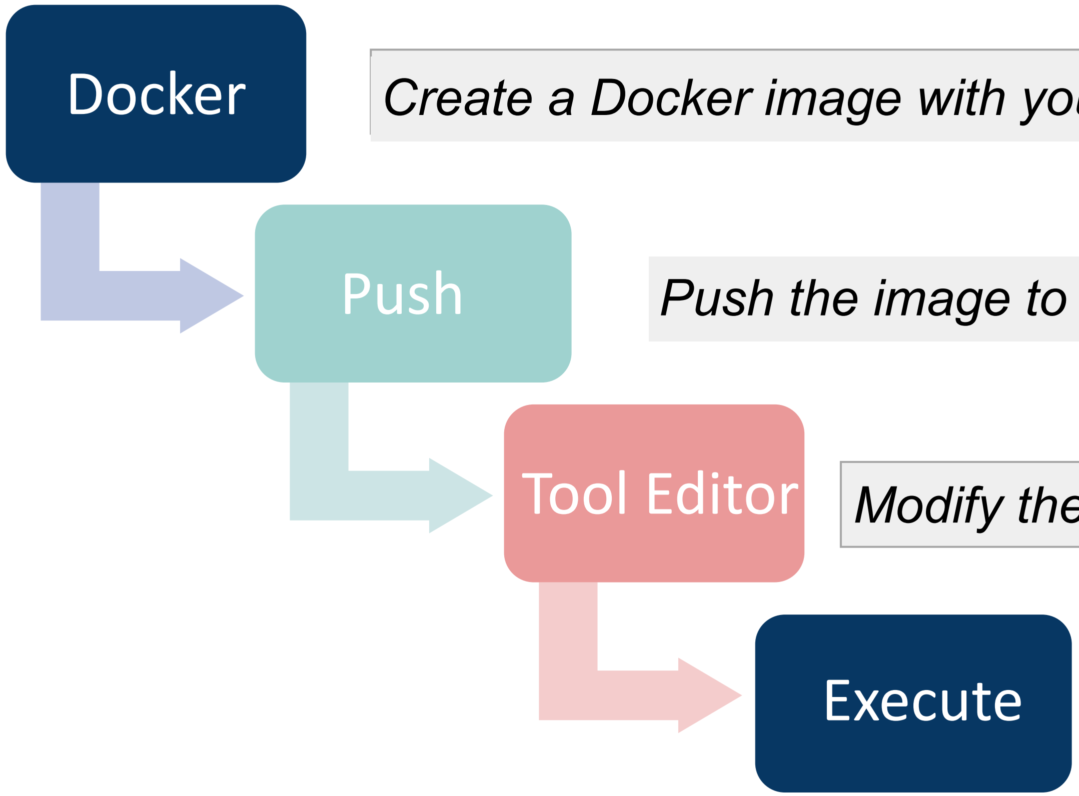
Cancer Genomics Cloud platform



- Two petabytes of multi-dimensional genomics data available to ~3800 authorized researchers to analyse on the cloud
- The Cancer Genome Atlas (TCGA), a landmark cancer genomics program, molecularly characterized over 20,000 primary cancer and matched normal samples

- Free registration for academia with $300 credit!

Bringing your own tools to the Platform



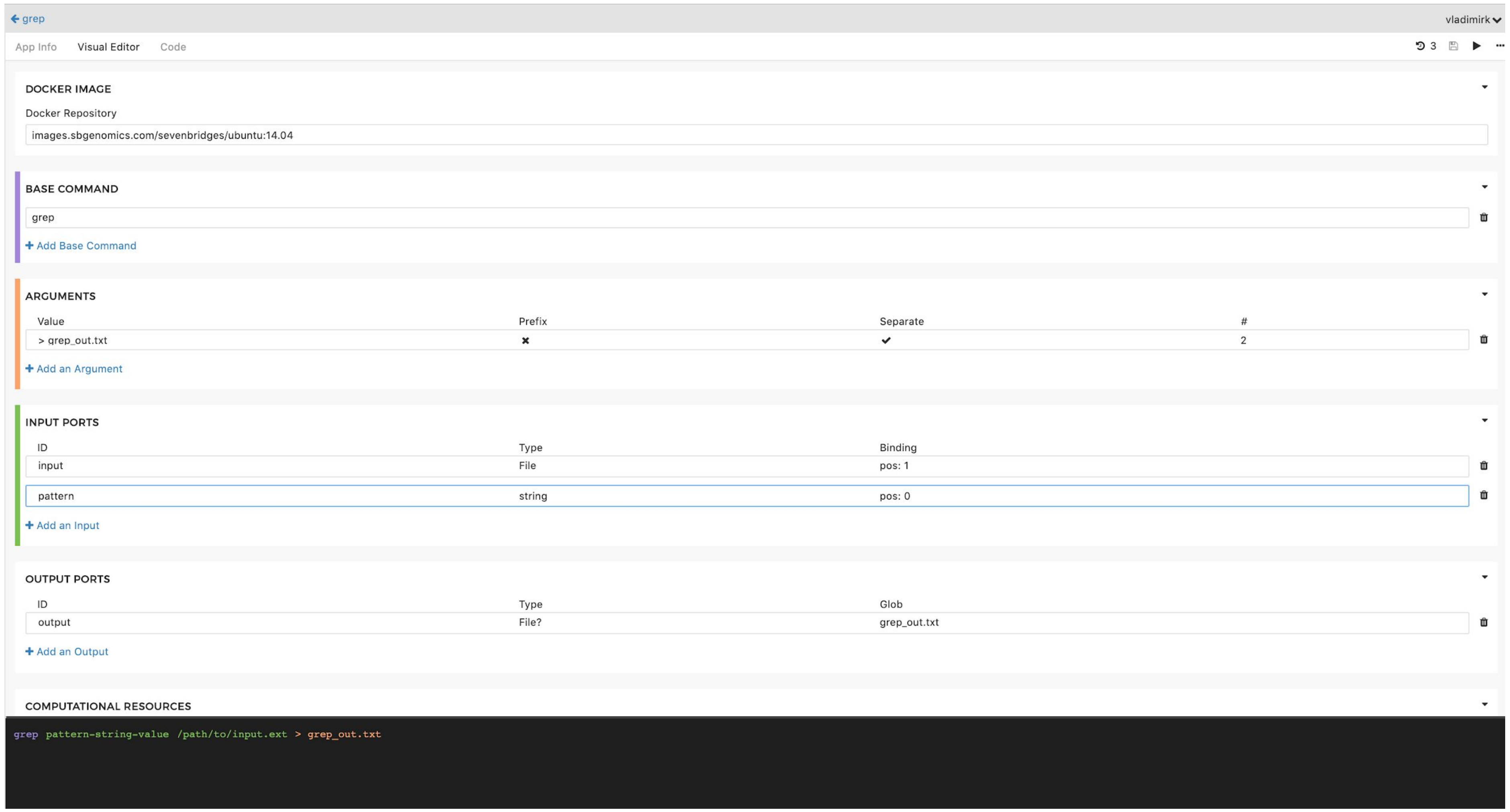
Create a Docker image with your tool

Push the image to a SB repository

Modify the tool behavior using Web Composer

Version & Execute on Platform

Let’s build some tool!



DOCKER IMAGE

Docker Repository

images.sbgenomics.com/sevenbridges/ubuntu:14.04

BASE COMMAND

grep

+ Add Base Command

ARGUMENTS

Value	Prefix	Separate	#
> grep_out.txt	✖	✓	2

+ Add an Argument

INPUT PORTS

ID	Type	Binding
input	File	pos: 1
pattern	string	pos: 0

+ Add an Input

OUTPUT PORTS

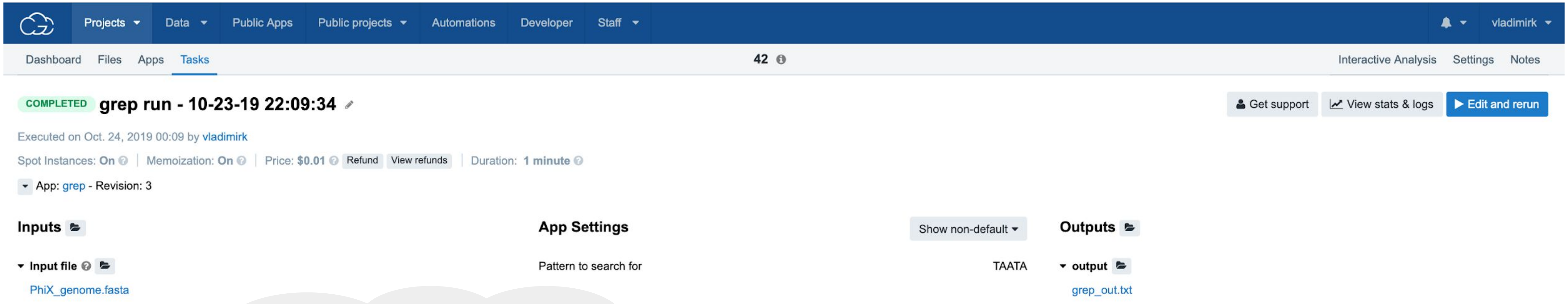
ID	Type	Glob
output	File?	grep_out.txt

+ Add an Output

COMPUTATIONAL RESOURCES

```
grep pattern-string-value /path/to/input.ext > grep_out.txt
```

...and run it!



COMPLETED grep run - 10-23-19 22:09:34

Executed on Oct. 24, 2019 00:09 by vladimirk

Spot Instances: On | Memoization: On | Price: $0.01 | Duration: 1 minute

App: grep - Revision: 3

Inputs

Input file

PhiX_genome.fasta

App Settings

Pattern to search for TAATA

Outputs

output

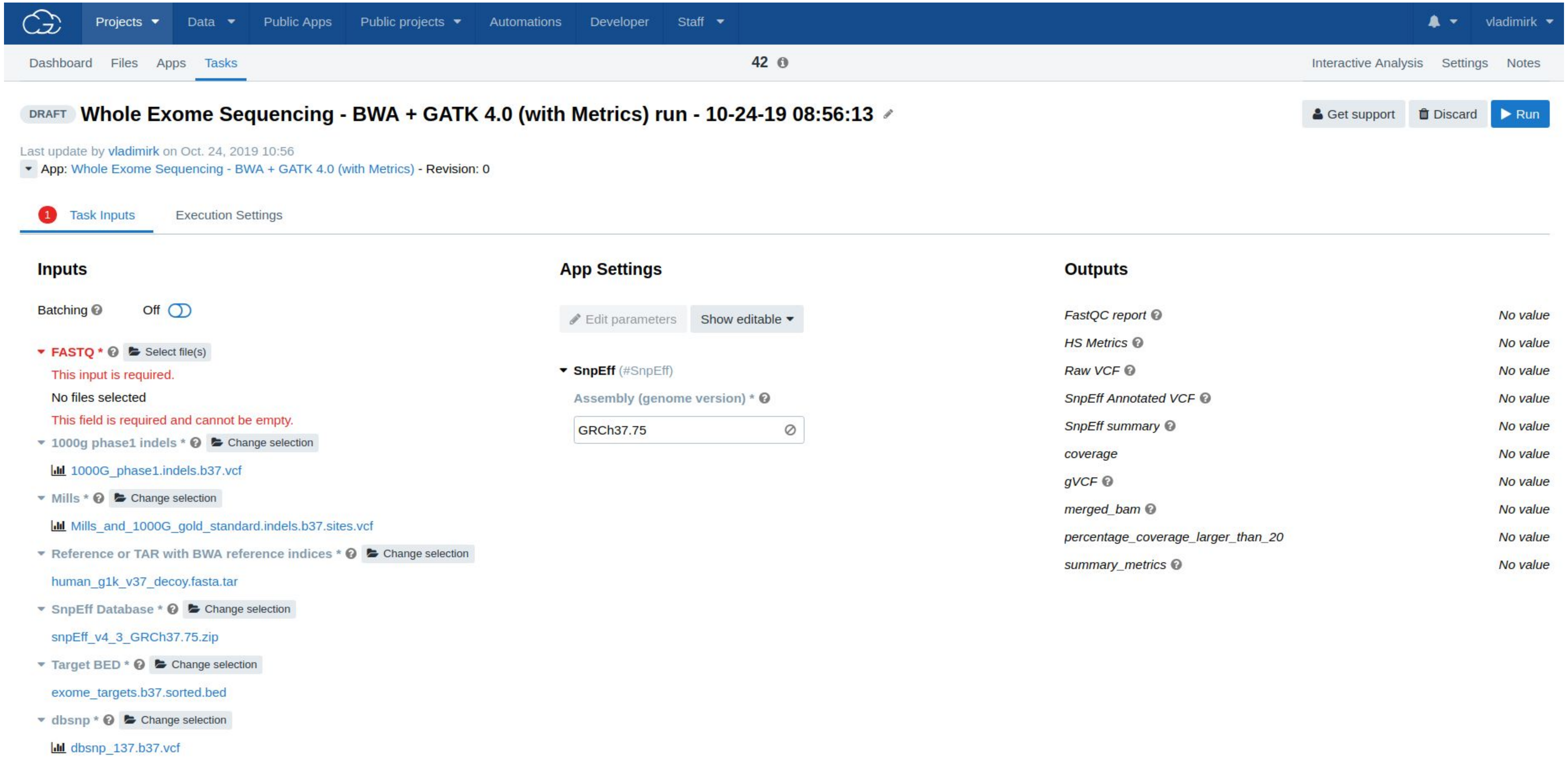
grep_out.txt

PhiX is an icosahedral, nontailed bacteriophage with a single-stranded DNA. It has a tiny **genome** with 5386 nucleotides and was the first DNA **genome** to be sequenced by Fred Sanger. Due to its small, well-defined **genome** sequence, **PhiX** has been commonly used as a control for Illumina sequencing runs.

So, what just happened?

- Request for default (c4.2xlarge) instance sent to aws
- Initialize instance
- cwl.job.json created from task inputs and parameters
- Together with cwl.app.json sent to initialized aws instance
- Download input files to the aws instance
- Download of docker image(s) of the tool(s)
- Run the tool inside docker container
- Collect marked outputs and upload them to the cloud storage attached to our platform’s project

What about some real data?



DRAFT **Whole Exome Sequencing - BWA + GATK 4.0 (with Metrics) run - 10-24-19 08:56:13**

Last update by vladimirk on Oct. 24, 2019 10:56

App: Whole Exome Sequencing - BWA + GATK 4.0 (with Metrics) - Revision: 0

Inputs

Batching Off

FASTQ * Select file(s)

This input is required.

No files selected

This field is required and cannot be empty.

1000g phase1 indels * Change selection

1000G_phase1.indels.b37.vcf

Mills * Change selection

Mills_and_1000G_gold_standard.indels.b37.sites.vcf

Reference or TAR with BWA reference indices * Change selection

human_g1k_v37_decoy.fasta.tar

SnpEff Database * Change selection

snpEff_v4_3_GRCh37.75.zip

Target BED * Change selection

exome_targets.b37.sorted.bed

dbsnp * Change selection

dbsnp_137.b37.vcf

App Settings

SnpEff (#SnpEff)

Assembly (genome version) *

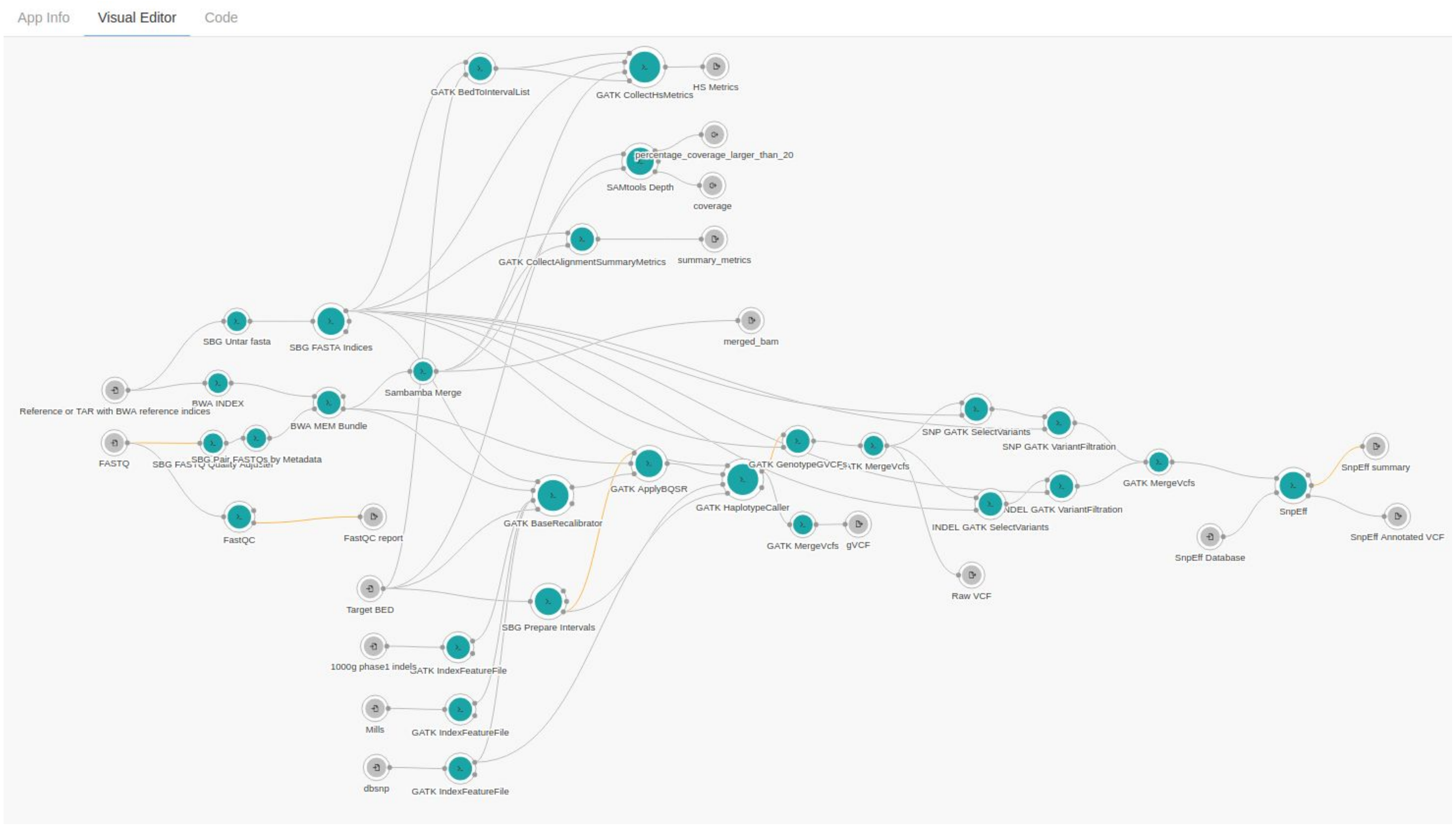
GRCh37.75

Outputs

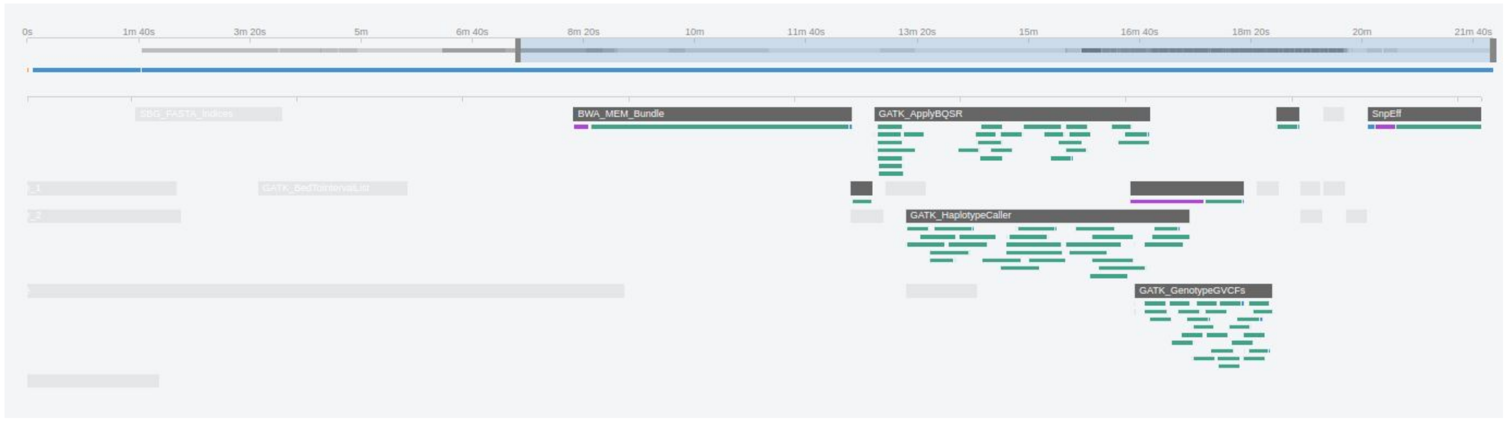
FastQC report	*No value*
HS Metrics	*No value*
Raw VCF	*No value*
SnpEff Annotated VCF	*No value*
SnpEff summary	*No value*
coverage	*No value*
gVCF	*No value*
merged_bam	*No value*
percentage_coverage_larger_than_20	*No value*
summary_metrics	*No value*

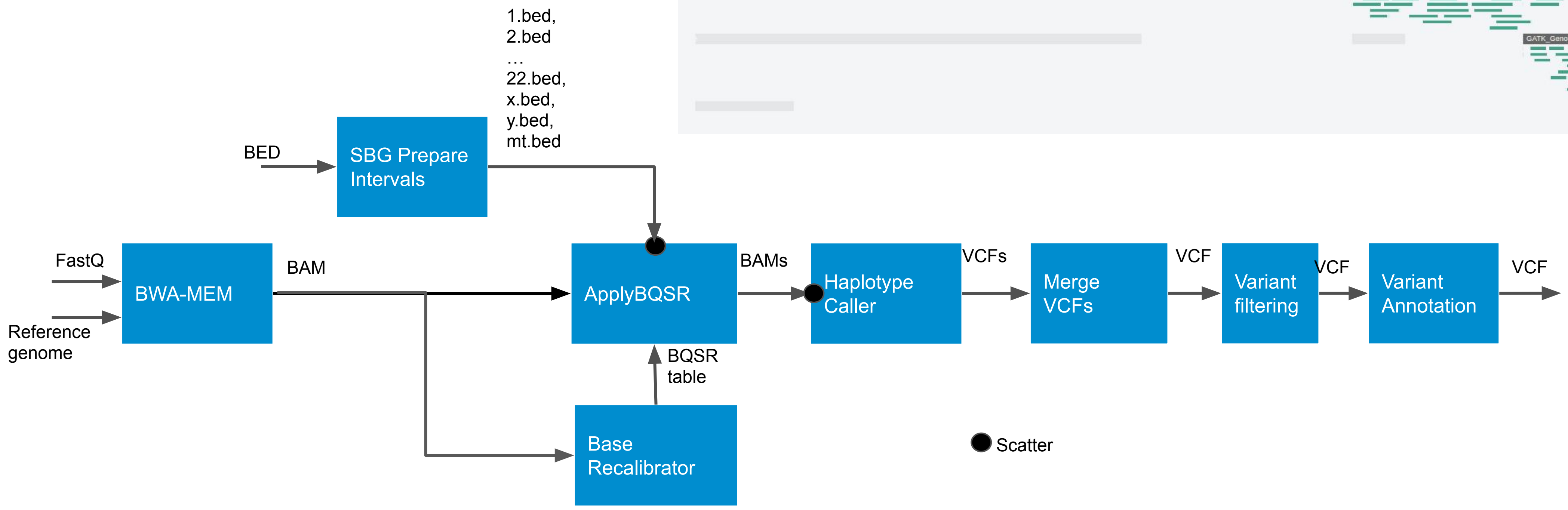
WES

...with real analysis!

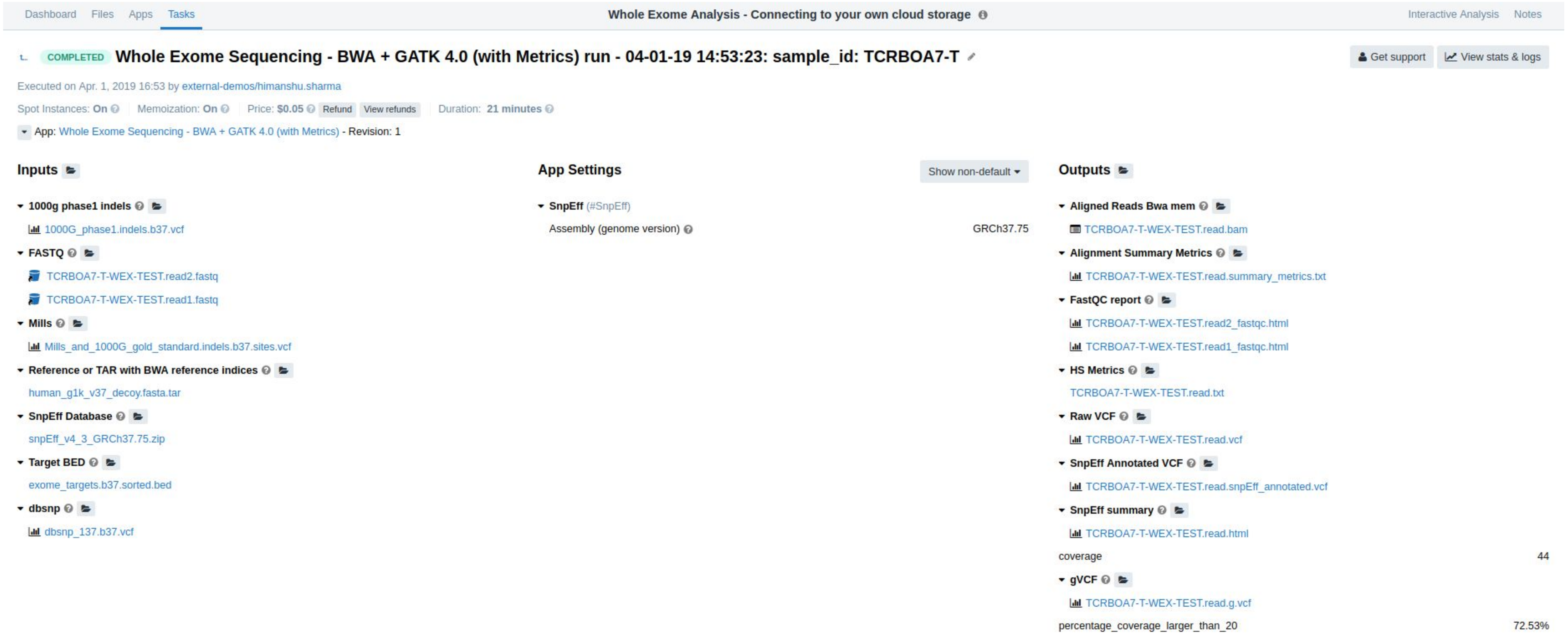


...with real analysis!





Whole exome sequencing execution



Whole Exome Analysis - Connecting to your own cloud storage

COMPLETED **Whole Exome Sequencing - BWA + GATK 4.0 (with Metrics) run - 04-01-19 14:53:23: sample_id: TCRBOA7-T**

Executed on Apr. 1, 2019 16:53 by external-demos/himanshu.sharma

Spot Instances: **On** | Memoization: **On** | Price: **$0.05** | Duration: **21 minutes**

App: Whole Exome Sequencing - BWA + GATK 4.0 (with Metrics) - Revision: 1

Inputs

- **1000g phase1 indels**
  - 1000G_phase1.indels.b37.vcf
- **FASTQ**
  - TCRBOA7-T-WEX-TEST.read2.fastq
  - TCRBOA7-T-WEX-TEST.read1.fastq
- **Mills**
  - Mills_and_1000G_gold_standard.indels.b37.sites.vcf
- **Reference or TAR with BWA reference indices**
  - human_g1k_v37_decoy.fasta.tar
- **SnpEff Database**
  - snpEff_v4_3_GRCh37.75.zip
- **Target BED**
  - exome_targets.b37.sorted.bed
- **dbsnp**
  - dbsnp_137.b37.vcf

App Settings

- **SnpEff** (#SnpEff)

Assembly (genome version)	GRCh37.75

Outputs

- **Aligned Reads Bwa mem**
  - TCRBOA7-T-WEX-TEST.read.bam
- **Alignment Summary Metrics**
  - TCRBOA7-T-WEX-TEST.read.summary_metrics.txt
- **FastQC report**
  - TCRBOA7-T-WEX-TEST.read2_fastqc.html
  - TCRBOA7-T-WEX-TEST.read1_fastqc.html
- **HS Metrics**
  - TCRBOA7-T-WEX-TEST.read.txt
- **Raw VCF**
  - TCRBOA7-T-WEX-TEST.read.vcf
- **SnpEff Annotated VCF**
  - TCRBOA7-T-WEX-TEST.read.snpEff_annotated.vcf
- **SnpEff summary**
  - TCRBOA7-T-WEX-TEST.read.html

coverage	44

- **gVCF**
  - TCRBOA7-T-WEX-TEST.read.g.vcf

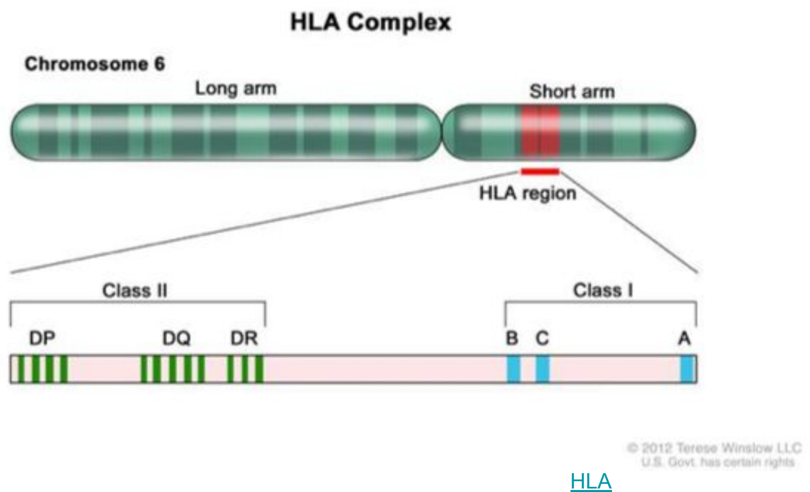
percentage_coverage_larger_than_20	72.53%

Exercise 1: Wrap FastQC tool

- Complete the tutorial
- Add cgc user pedjao to the project
- Send the link to the executed task at your CGC project to pedjao@etf.bg.ac.rs, together with name and number of index, the mail subject should be “GI2021_ DZ1”
- Do it before the next lesson
- 10 (easy) points :)

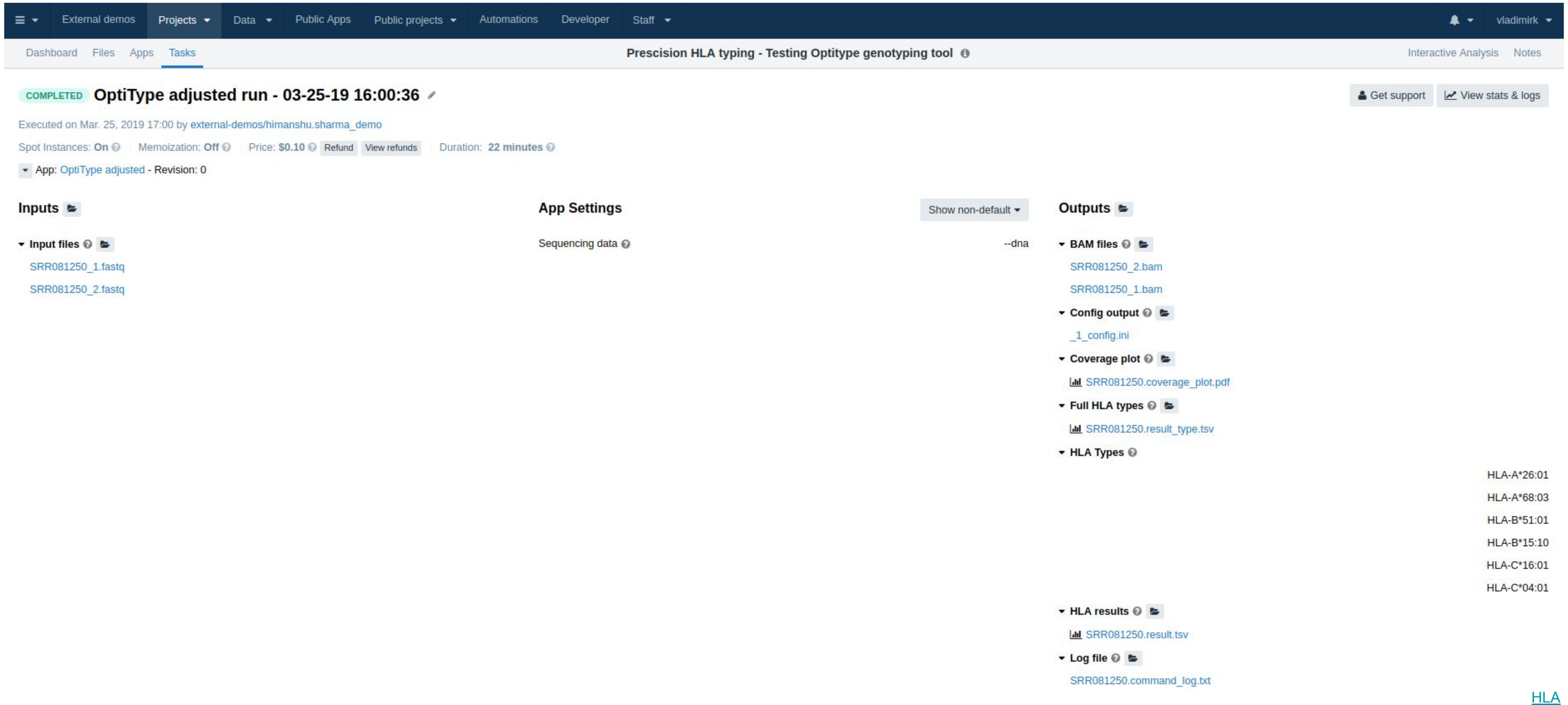
HLA Typing

- The HLA gene family provides instructions for making a group of related proteins known as the human leukocyte antigen (HLA) complex.
- The HLA complex helps the immune system distinguish the body's proteins from proteins made by foreign invaders such as viruses and bacteria.
- HLA typing has been widely used for reducing the risk of organ rejection
- Specific HLA variants are associated with both autoimmune (e.g. type 1 diabetes, rheumatoid arthritis) and infectious (e.g. HIV, Hepatitis C) diseases



HLA

HLA Typing



Prescision HLA typing - Testing Optitype genotyping tool

COMPLETED OptiType adjusted run - 03-25-19 16:00:36

Executed on Mar. 25, 2019 17:00 by external-demos/himanshu.sharma_demo

Spot Instances: On | Memoization: Off | Price: $0.10 | Duration: 22 minutes

App: OptiType adjusted - Revision: 0

Inputs

Input files
- SRR081250_1.fastq
- SRR081250_2.fastq

App Settings

Sequencing data --dna

Outputs

BAM files
- SRR081250_2.bam
- SRR081250_1.bam

Config output
- _1_config.ini

Coverage plot
- SRR081250.coverage_plot.pdf

Full HLA types
- SRR081250.result_type.tsv

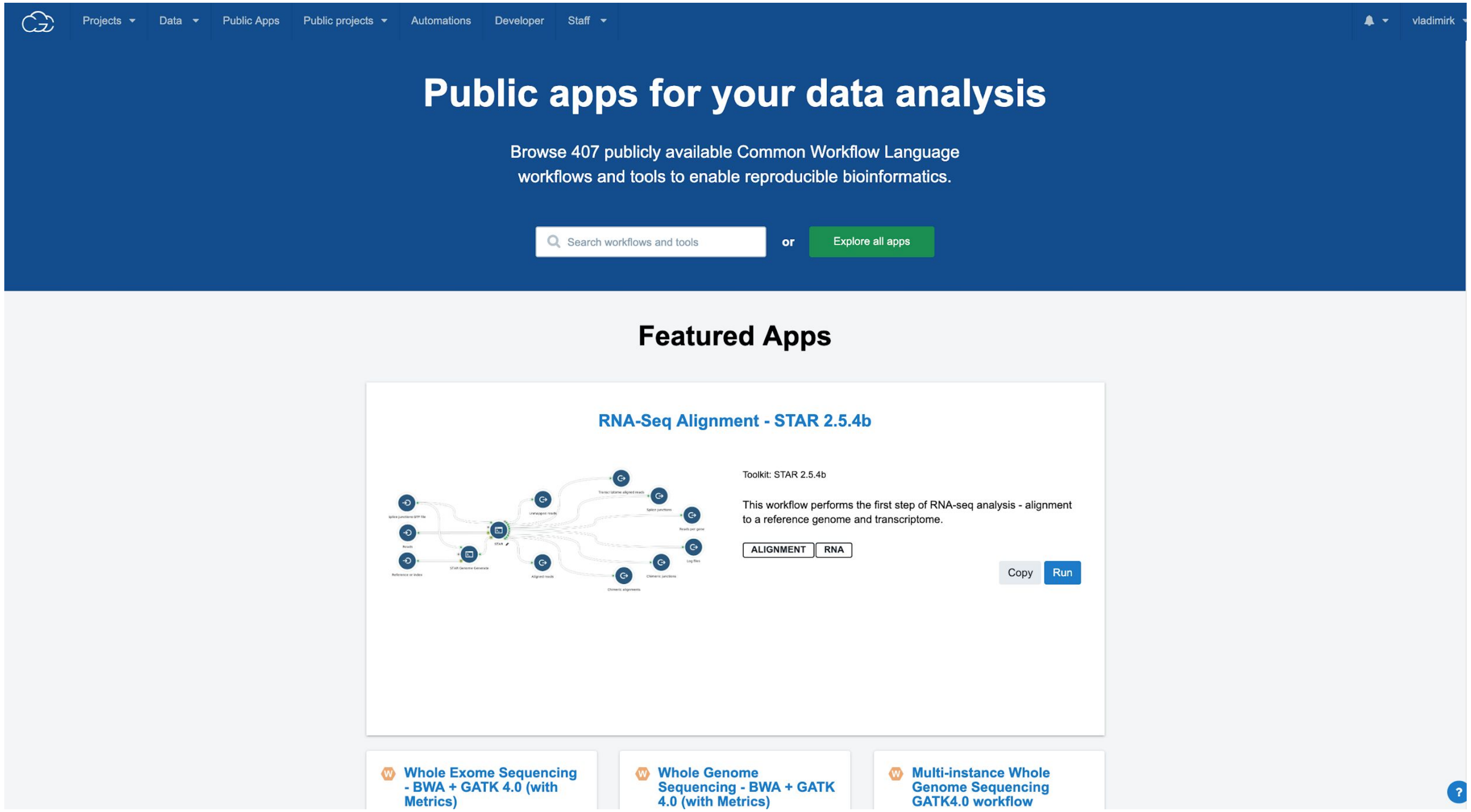
HLA Types
- HLA-A*26:01
- HLA-A*68:03
- HLA-B*51:01
- HLA-B*15:10
- HLA-C*16:01
- HLA-C*04:01

HLA results
- SRR081250.result.tsv

Log file
- SRR081250.command_log.txt

HLA

Public App Gallery



Public apps for your data analysis

Browse 407 publicly available Common Workflow Language workflows and tools to enable reproducible bioinformatics.

Featured Apps

RNA-Seq Alignment - STAR 2.5.4b

Toolkit: STAR 2.5.4b

This workflow performs the first step of RNA-seq analysis - alignment to a reference genome and transcriptome.

Whole Exome Sequencing - BWA + GATK 4.0 (with Metrics)

Whole Genome Sequencing - BWA + GATK 4.0 (with Metrics)

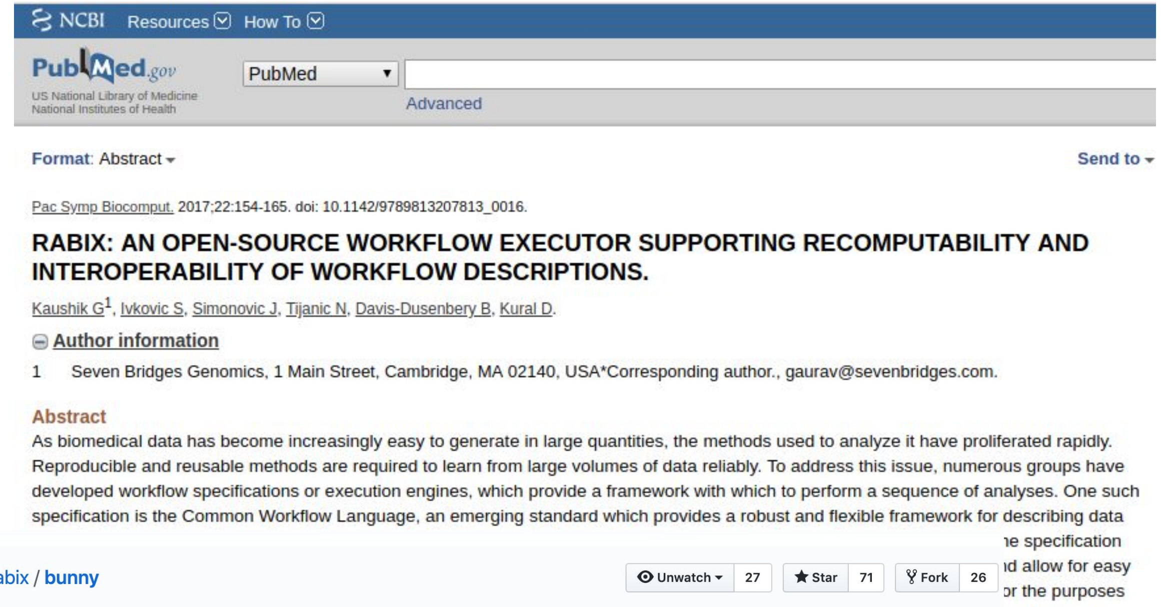
Multi-instance Whole Genome Sequencing GATK4.0 workflow

Local executor

Runnable from the command line
Suitable for local testing and development

```
./rabix [OPTIONS] <app> <inputs>
```

rabix.io
https://github.com/rabix/bunny



Pac Symp Biocomput. 2017;22:154-165. doi: 10.1142/9789813207813_0016.

RABIX: AN OPEN-SOURCE WORKFLOW EXECUTOR SUPPORTING RECOMPUTABILITY AND INTEROPERABILITY OF WORKFLOW DESCRIPTIONS.

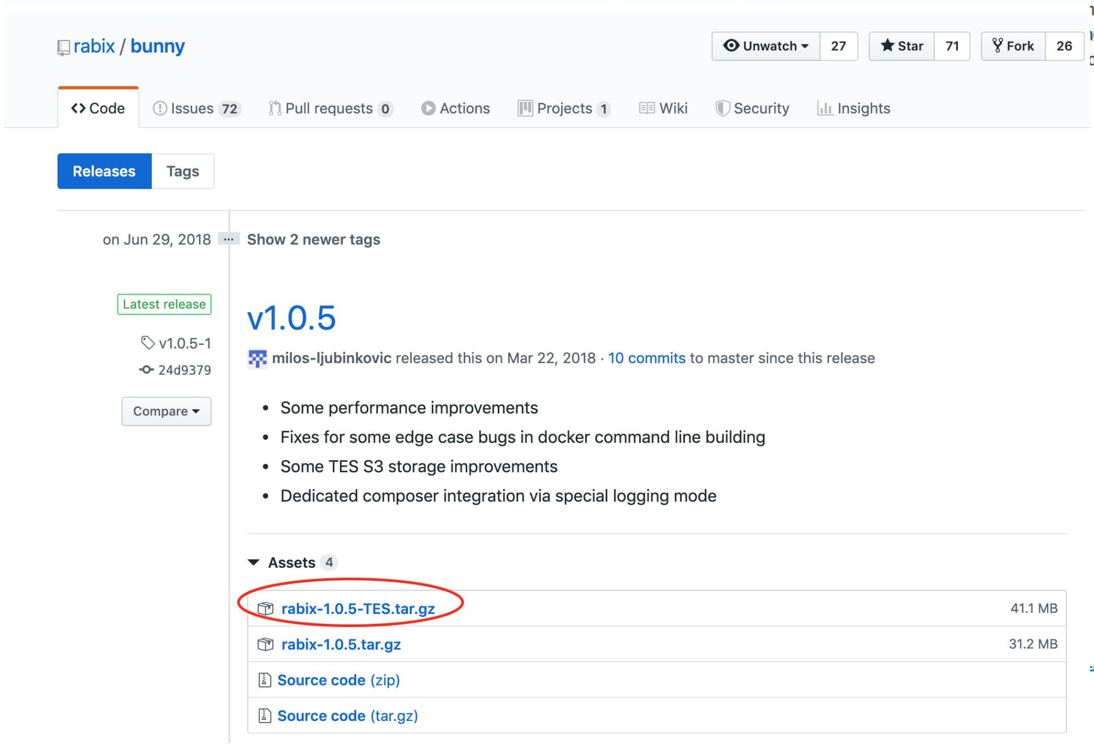
Kaushik G[1], Ivkovic S, Simonovic J, Tijanic N, Davis-Dusenbery B, Kural D.

Author information

1 Seven Bridges Genomics, 1 Main Street, Cambridge, MA 02140, USA*Corresponding author., gaurav@sevenbridges.com.

Abstract

As biomedical data has become increasingly easy to generate in large quantities, the methods used to analyze it have proliferated rapidly. Reproducible and reusable methods are required to learn from large volumes of data reliably. To address this issue, numerous groups have developed workflow specifications or execution engines, which provide a framework with which to perform a sequence of analyses. One such specification is the Common Workflow Language, an emerging standard which provides a robust and flexible framework for describing data
ne specification
nd allow for easy
or the purposes



on Jun 29, 2018 ··· Show 2 newer tags

Latest release
v1.0.5-1
24d9379

v1.0.5

milos-ljubinkovic released this on Mar 22, 2018 · 10 commits to master since this release

- Some performance improvements
- Fixes for some edge case bugs in docker command line building
- Some TES S3 storage improvements
- Dedicated composer integration via special logging mode

Assets 4

rabix-1.0.5-TES.tar.gz	41.1 MB
rabix-1.0.5.tar.gz	31.2 MB
Source code (zip)	
Source code (tar.gz)	

A

Local executor

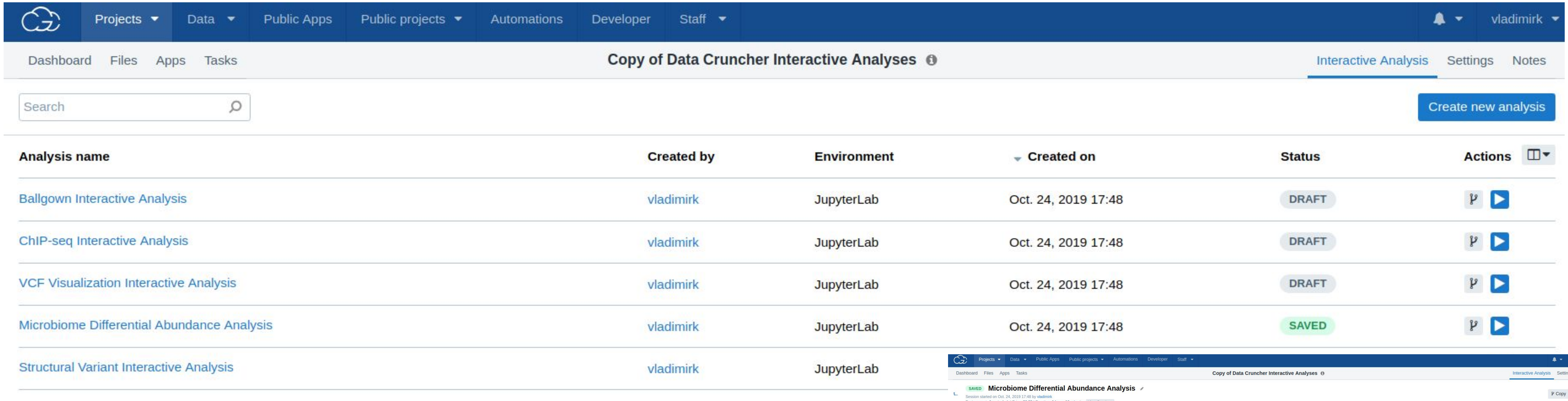
```
# Install docker download and unpack rabix
./rabix -b ./ grep.cwl.json inputs.json
ll grep-2020-02-11-155503.852/root/
-rw-r--r--  1 vladimirk  staff  100 Feb 11 15:55 cmd.log
-rw-r--r--  1 vladimirk  staff  550 Feb 11 15:55 cwl.output.json
-rw-r--r--  1 vladimirk  staff   27 Feb 11 15:55 out.txt
cat grep-2020-02-11-155503.852/root/cwl.output.json
{
  "output" : {
    "basename" : "out.txt",
    "checksum" : "sha1$0a3e8ce4ad3bcd5db0804f28752499adfe2ca5d1",
    "class" : "File",
    "dirname" : "grep-2020-02-11-155503.852/root",
    "location" : "grep-2020-02-11-155503.852/root/out.txt",
    "nameext" : ".txt",
    "nameroot" : "out",
    "path" : "grep-2020-02-11-155503.852/root/out.txt",
    "size" : 27
  }
}
cat grep-2020-02-11-155503.852/root/out.txt
ACTGA
GAGAGAGA
GA
GGGAAAGA
cat grep-2020-02-11-155503.852/root/cmd.log
grep GA dummy.fasta > out.txt
```

```
# grep.json
{
   "class": "CommandLineTool",
   "cwlVersion": "v1.0",
   "$namespaces": {"sbg": "https://sevenbridges.com"},
   "baseCommand": ["grep"],
   "inputs": [
       {  "id": "pattern",
          "type": "string",
          "inputBinding": {"position": 1},
          "label": "Pattern"},
       {  "id": "input",
          "type": "File",
          "inputBinding": {"position": 2}}
   ],
   "outputs": [
       {"id": "output",
          "type": "File?",
          "outputBinding": {
              "glob": "*.txt"}}
   ],
   "arguments": [
       {"position": 3,"prefix": "",
        "valueFrom": "> out.txt"}
   ],
   "requirements": [
       {"class": "ShellCommandRequirement"},
       {"class": "DockerRequirement", "dockerPull": ubuntu:14.04"}}
```

```
# inputs.json
{
   "input" : {
       "path" : "dummy.fasta",
       "class" : "File"
   },
   "pattern" : "GA"
}
```

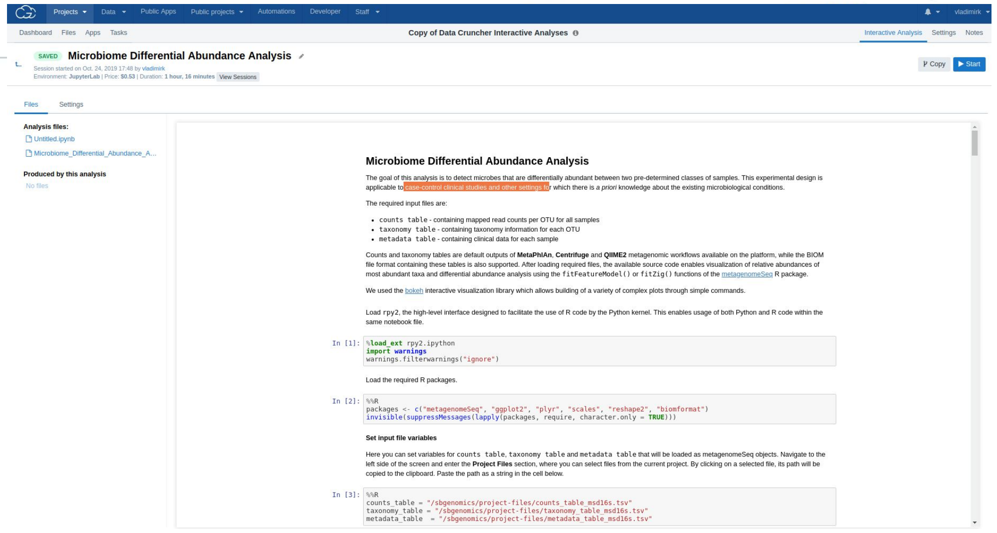
4. Jupyter Notebook bioinformatic analysis on the cloud

Interactive analysis



Copy of Data Cruncher Interactive Analyses

Analysis name	Created by	Environment	Created on	Status	Actions
Ballgown Interactive Analysis	vladimirk	JupyterLab	Oct. 24, 2019 17:48	DRAFT	
ChIP-seq Interactive Analysis	vladimirk	JupyterLab	Oct. 24, 2019 17:48	DRAFT	
VCF Visualization Interactive Analysis	vladimirk	JupyterLab	Oct. 24, 2019 17:48	DRAFT	
Microbiome Differential Abundance Analysis	vladimirk	JupyterLab	Oct. 24, 2019 17:48	SAVED	
Structural Variant Interactive Analysis	vladimirk	JupyterLab			



Copy of Data Cruncher Interactive Analyses

SAVED Microbiome Differential Abundance Analysis

Session started on Oct. 24, 2019 17:48 by vladimirk

Environment: JupyterLab | Price: $0.53 | Duration: 1 hour, 16 minutes View Sessions

Analysis files:
- Untitled.ipynb
- Microbiome_Differential_Abundance_A...

Produced by this analysis

No files

Microbiome Differential Abundance Analysis

The goal of this analysis is to detect microbes that are differentially abundant between two pre-determined classes of samples. This experimental design is applicable to case-control clinical studies and other settings for which there is *a priori* knowledge about the existing microbiological conditions.

The required input files are:

- `counts table` - containing mapped read counts per OTU for all samples
- `taxonomy table` - containing taxonomy information for each OTU
- `metadata table` - containing clinical data for each sample

Counts and taxonomy tables are default outputs of **MetaPhlAn**, **Centrifuge** and **QIIME2** metagenomic workflows available on the platform, while the BIOM file format containing these tables is also supported. After loading required files, the available source code enables visualization of relative abundances of most abundant taxa and differential abundance analysis using the `fitFeatureModel()` or `fitZig()` functions of the metagenomeSeq R package.

We used the bokeh interactive visualization library which allows building of a variety of complex plots through simple commands.

Load `rpy2`, the high-level interface designed to facilitate the use of R code by the Python kernel. This enables usage of both Python and R code within the same notebook file.

```
In [1]: %load_ext rpy2.ipython
        import warnings
        warnings.filterwarnings("ignore")
```

Load the required R packages.

```
In [2]: %%R
        packages <- c("metagenomeSeq", "ggplot2", "plyr", "scales", "reshape2", "biomformat")
        invisible(suppressMessages(lapply(packages, require, character.only = TRUE)))
```

Set input file variables

Here you can set variables for `counts table`, `taxonomy table` and `metadata table` that will be loaded as metagenomeSeq objects. Navigate to the left side of the screen and enter the **Project Files** section, where you can select files from the current project. By clicking on a selected file, its path will be copied to the clipboard. Paste the path as a string in the cell below.

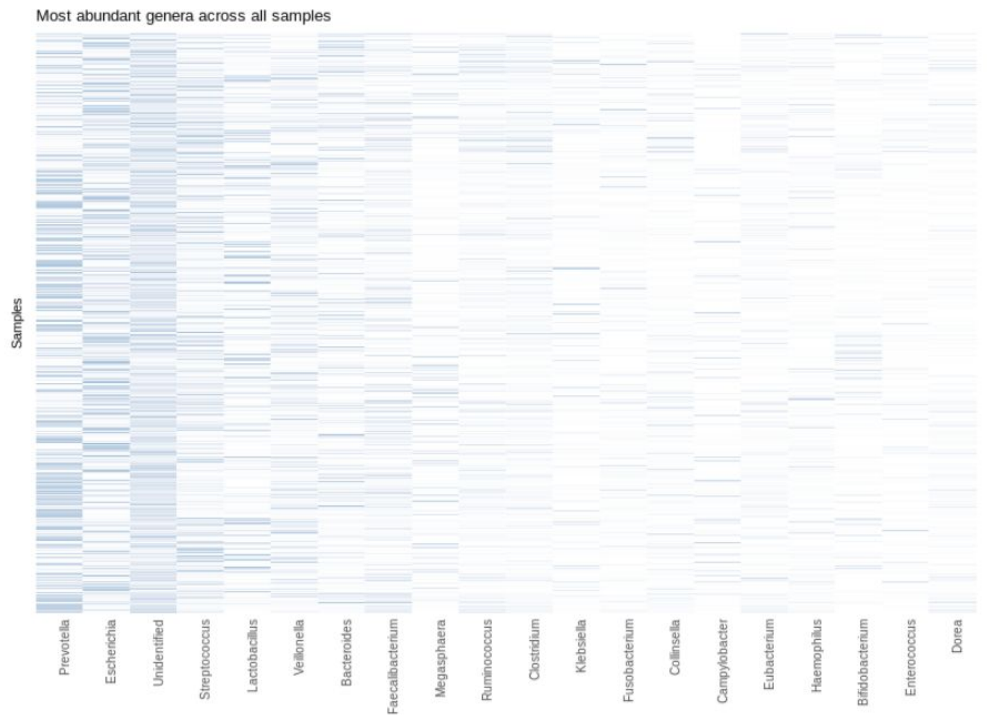
```
In [3]: %%R
        counts_table = "/sbgenomics/project-files/counts_table_msd16s.tsv"
        taxonomy_table = "/sbgenomics/project-files/taxonomy_table_msd16s.tsv"
        metadata_table  = "/sbgenomics/project-files/metadata_table_msd16s.tsv"
```

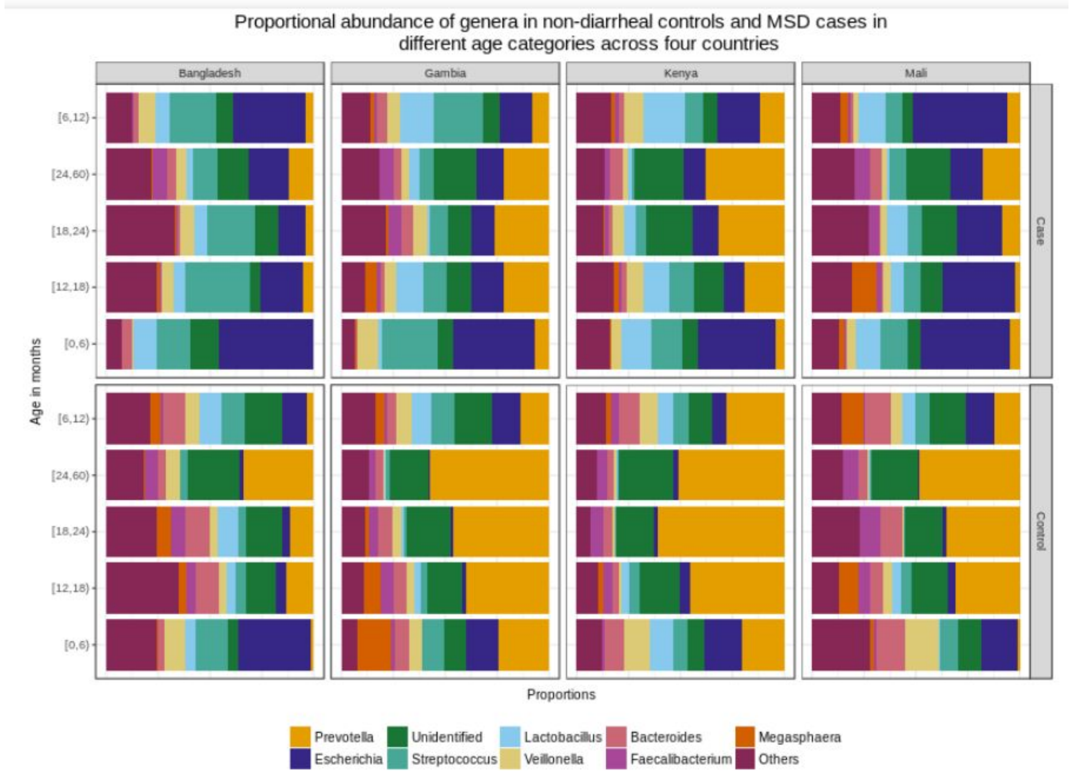
Run python/R Jupyter Notebook on the cloud
Further process outputs from bioinformatics tasks

```
pattern = 'ACCT'
with
open('/sbgenomics/project-files/PhiX_genome.fast
a', 'r') as myfile:
    data=myfile.readlines()
data = ''.join(data).replace('\n', '')
cnt = 0
for i in range(0, len(data) - len(pattern)):
    if data[i:i+len(pattern)] == pattern:
        cnt += 1
        print(cnt, i)
```

Microbiome Differential Abundance Analysis

Detect microbes that are differentially abundant between disease-control (~500 each) samples





HLA

We use Git!

- Created by Linus Torvalds, creator of Linux, in 2005
- Came out of Linux development community
- Designed to do version control on Linux kernel
- Goals of Git:
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects efficiently

(A "git" is a cranky old man. Linus meant himself.)

- Instructions to install Git: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Installing/learning Git!

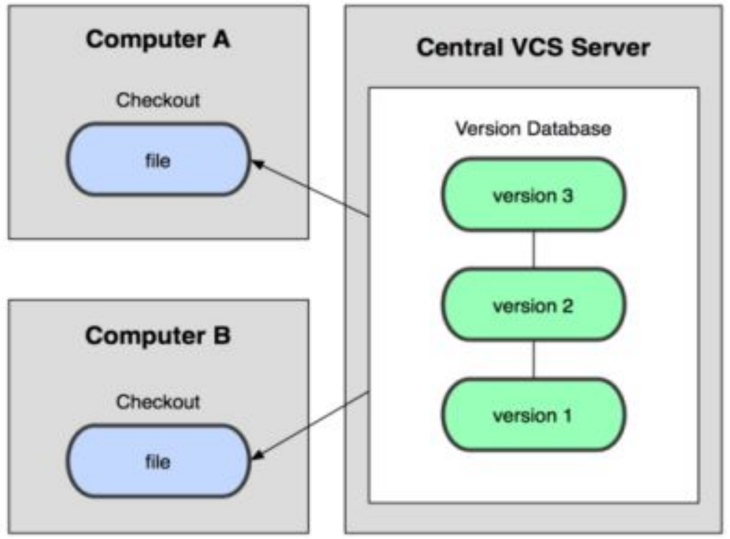
- Git website: http://git-scm.com/
- Free online book: http://git-scm.com/book
- Reference page for Git: http://gitref.org/index.html
- Git tutorial: http://schacon.github.com/git/gittutorial.html
- Git slides: https://courses.cs.washington.edu/courses/cse403/13au/lectures/git.ppt.pdf
- Git for Computer Scientists: http://eagain.net/articles/git-for-computer-scientists
- At command line: (where verb = config, add, commit, etc.)

```
git help verb
```

- Instructions to install Git: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Centralized Versioning Control System

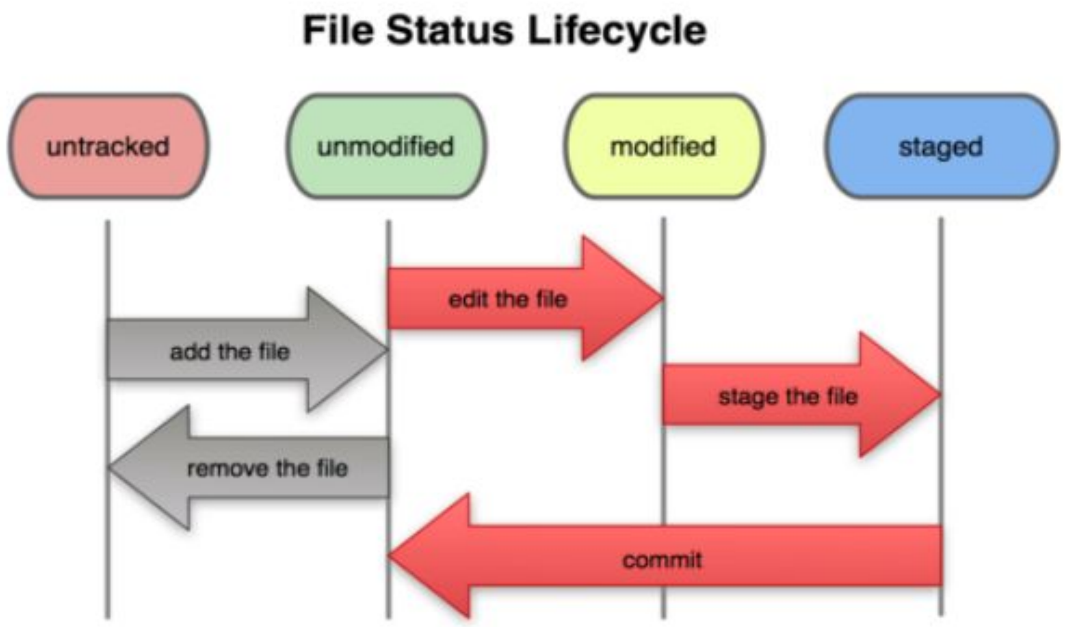
- A central server repository (repo)
  holds the "official copy" of the code
- The server maintains the sole version history of the repo
- You make "checkouts" of it to your local copy
- You make local modifications
- Your changes are not versioned
- When you're done, you "check in" back to the server
- your check in increments the repo's version



Basic Git flow

1. Modify files in your working directory
2. Stage files, adding snapshots of them to your staging area
3. Commit, which takes the files in the staging area
4. Store that snapshot permanently to your Git directory

```
git add file.py
git commit -m "Description of change."
git push origin master
```



Initial Git configuration

Set the name and email for Git to use when you commit:

- git config --global user.name "Bugs Bunny"
- git config --global user.email bugs@gmail.com

You can call git config –list to verify these are set.

Git commands

command	description
`git clone url [dir]`	copy a Git repository so you can add to it
`git add file`	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help [command]`	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
`git checkout filename`	undoes your changes
Others: init, reset, branch, checkout, merge,log, tag	

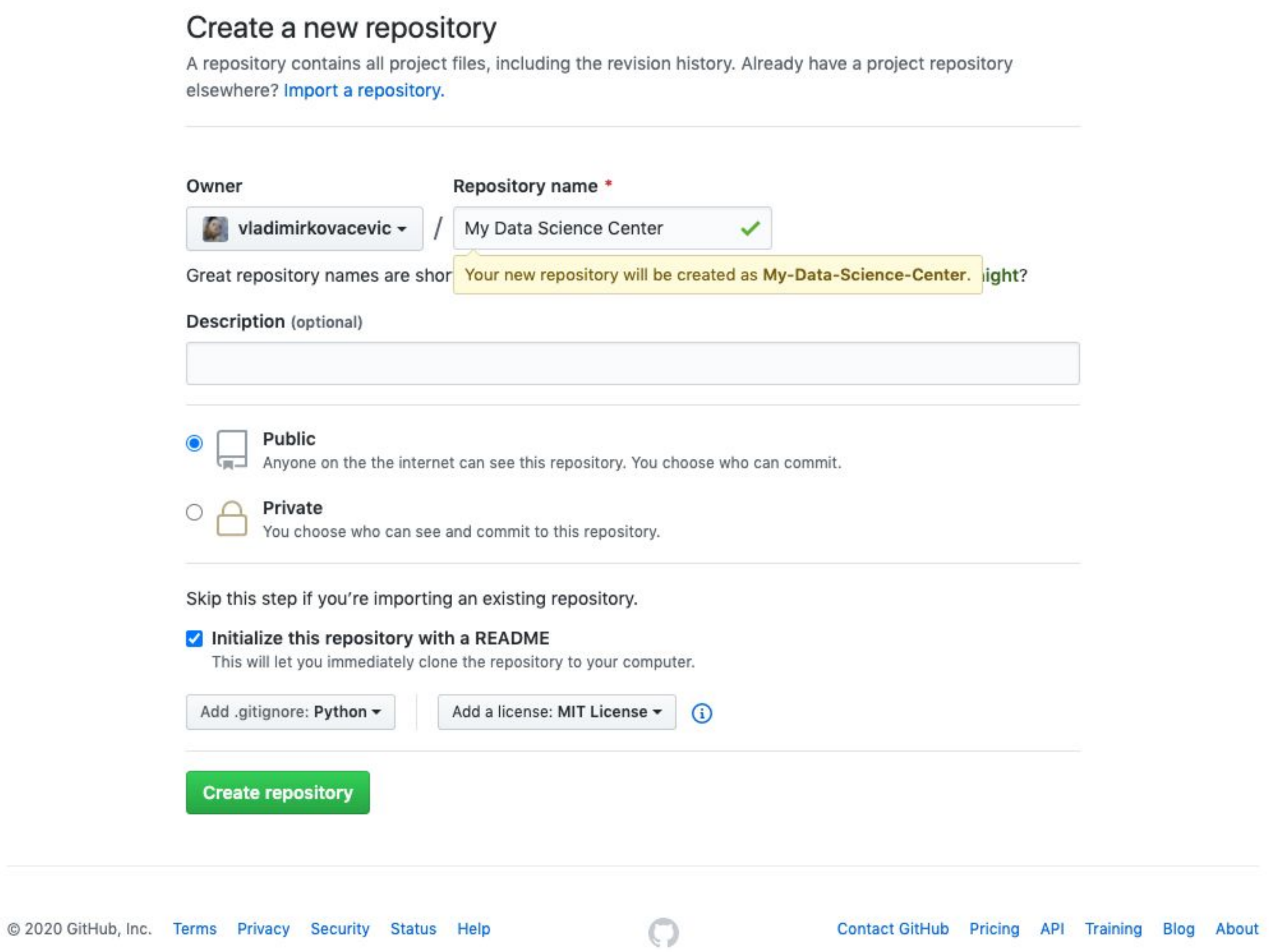
We use Github!

- GitHub.com is a site for online storage of Git repositories.
- You can create a remote repo there and push code to it.
- Many open source projects use it, such as the Linux kernel.
- You can get free space for open source projects, or you can pay for private projects.
- Free private repos for educational use: github.com/edu
- Question: Do I always have to use GitHub to use Git?
  - Answer: No! You can use Git locally for your own purposes.
  - Or you or someone else could set up a server to share files.
  - Or you could share a repo with users on the same file system, as long everyone has the needed file permissions).

Setup Github repo

- Create account on www.github.com
- Set an image :)
- Create repository

  My Data Science Center
  - Initialize with README
  - .gitignore Python
  - MIT License



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: vladimirkovacevic / Repository name *: My Data Science Center

Great repository names are shor Your new repository will be created as **My-Data-Science-Center**. ight?

Description (optional)

(●) **Public** Anyone on the the internet can see this repository. You choose who can commit.

( ) **Private** You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

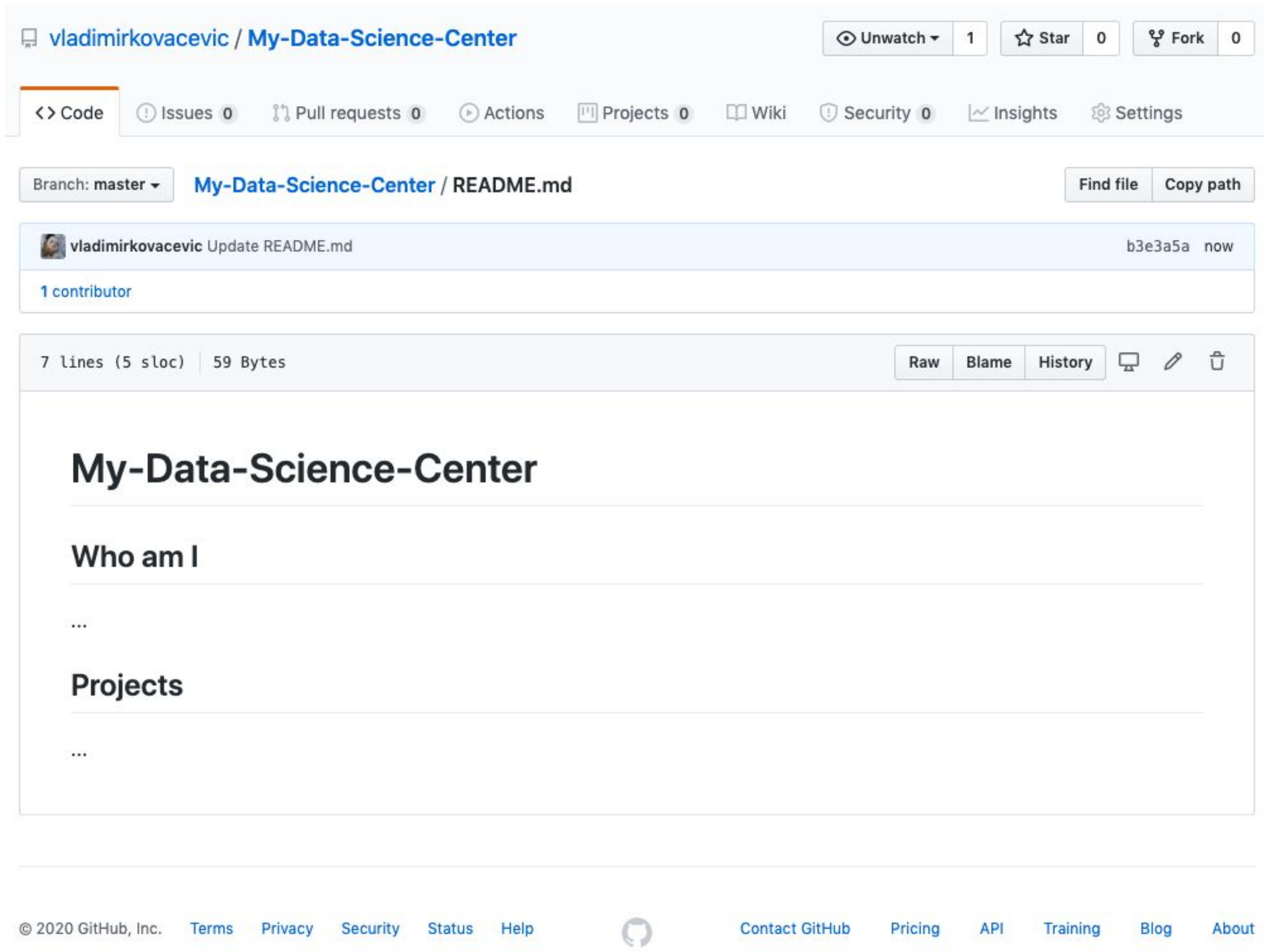
[✓] **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: **Python** Add a license: **MIT License**

Create repository
© 2020 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub Pricing API Training Blog About

Setup Github repo

- Add short biography
- Projects will come on the way



My-Data-Science-Center

Who am I

...

Projects

...

Thank you!

Questions?

@vladimir_bio

HLA