External PGAP

Freeing the Prokaryotic Genome Annotation Pipeline

Douglas Slotta

Bethesda, MD
Sept 12, 2019

History

- Steve Turner, CTO of PacBio asked for PGAP.
- Ben Busby was super enthused (of course).
- We had already been thinking of externalizing pipelines.
- Creation of an Industry/Government Consortium proposed.
- Pilot Project Approved.

Goals

Primary Goal

Running NCBI pipelines should not require NCBI.

- Ability of outside users to process genomes, including proprietary genomes (a perennial user request).
- Development of the NCBI staff skill set in externalizing GPipe software packages.
- Reduced burden on NCBI farm (submitters run PGAP, then submit results).
- External and Internal PGAP should produce identical results.

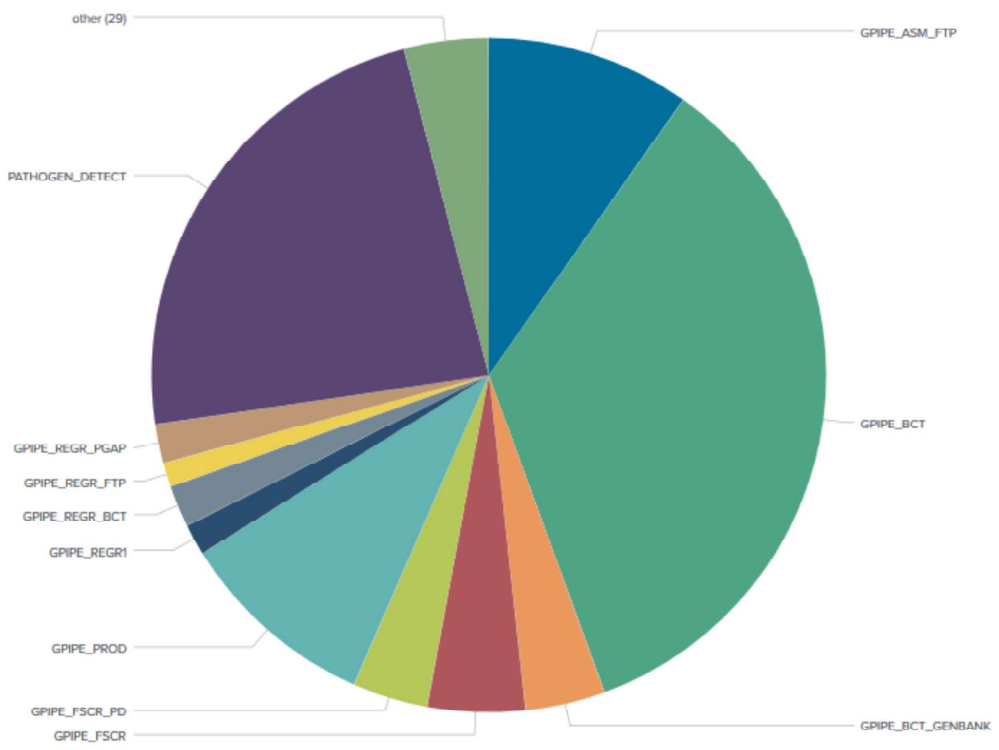
GPipe Usage of UGE Compute Farm by Service

Splunk pie chart of CPU for GPipe usage in the past 90 days

- PGAP at 35% ($13K/month)
- Pathogen at 23% ($9K/month)
- Total: $38K/month (excluding developer runs)

```
index=gridengine
sourcetype="uge:accounting"
owner="gp*" | eval
cost_per_month_usd=cpu*0.0128/3600/3
| chart sum(cost_per_month_usd) by
SERVER
```

U.S. National Library of Medicine
National Center for Biotechnology Information



Why not GPipe?

Needs:

- PanFS,
- MSSQL Server,
- SGE,
- NCBI internal databases,
- NCBI internal data files.

Why Common Workflow Language (CWL)?

- Emerging standard
- Understandable description (declarative, balance of complexity)
- Implemented by multiple platforms.
- Actively supported.
- Built-in container support.
- Models dataflows, not just workflows

Why not CWL?

- An *emerging standard* is not an *actual standard*.
- Tooling: not nearly as mature as those we currently use.
  - Retries
  - Monitoring
  - Scalability
- Optimization: GPipe is highly optimized for speed on NCBI systems (which is part of the problem).
- Simplicity of the model: dataflow but coarse-grained e.g. not stream oriented; application + input/output may be naive for parallel distributed computing (see e.g. MPI/Hadoop/Spark/Charm++).

Describing Common Workflow Language

Language syntax: YAML & JSON + JavaScript

- YAML for programmers who like significant whitespace; JSON for those who hate it.
  - YAML is superset of JSON:
    Any reasonable JSON file is a valid YAML file.
  - JSON is sufficient for CWL:
    YAML extensions are not needed.
- JavaScript for limited points of extension.

Taxonomic classification:

- **Syntax:** Explicit, DAG
- **Paradigm:** Configuration
- **Interaction:** N/A
  Varies by implementation
- **Task Granularity:** Coarse

Semantics and workflow concepts:

- Input, Output **files** via IRIs (Internationalized URIs), e.g. file://, http://, s3://
- **Data types** via IRIs, e.g. EDAM Ontology
- **Command Line Tools** with Arguments (parameters mapped to named and positional args)
- Workflow, **Nested Workflow**; DAGs without loops or conditionals (yet).
- **Resources** (CPU, memory, disk), coarse grained
- **Scatter/Gather** for data partitioning and batch processing

COMMON
WORKFLOW
LANGUAGE

Example Tool

```
cwlVersion: v1.0
class: CommandLineTool
hints:
  DockerRequirement:
    dockerPull: ncbi/amr:18.06
baseCommand: blastp
stdout: blastp.out
inputs:
  query:
    type: File
    inputBinding:
      prefix: -query
  db:
    type: Directory
    inputBinding:
      prefix: -db
      valueFrom: $(self.path)/$(self.basename)
outputs:
  - id: output
    type: File
    outputBinding:
      glob: "blastp.out"
```

Example Tool – Command

```
cwlVersion: v1.0
class: CommandLineTool
hints:
  DockerRequirement:
    dockerPull: ncbi/amr:18.06
baseCommand: blastp
stdout: blastp.out
inputs:
  query:
    type: File
    inputBinding:
      prefix: -query
  db:
    type: Directory
    inputBinding:
      prefix: -db
      valueFrom: $(self.path)/$(self.basename)
outputs:
  - id: output
    type: File
    outputBinding:
      glob: "blastp.out"
```

Example Tool – Input & Output

```
cwlVersion: v1.0
class: CommandLineTool
hints:
  DockerRequirement:
    dockerPull: ncbi/amr:18.06
baseCommand: blastp
stdout: blastp.out
inputs:
  query:
    type: File
    inputBinding:
      prefix: -query
  db:
    type: Directory
    inputBinding:
      prefix: -db
      valueFrom: $(self.path)/$(self.basename)
outputs:
  - id: output
    type: File
    outputBinding:
      glob: "blastp.out"
```

Example Workflow

```
cwlVersion: v1.0
class: Workflow
requirements:
  - class: SubworkflowFeatureRequirement
inputs:
  query: File
  fasta: File
outputs:
  result:
    type: File
    outputSource: blastp/output
steps:
  makeblastdb:
    run: wf_makeblastdb.cwl
    in:
      fasta: fasta
    out:
      [blastdb]
  blastp:
    run: blastp.cwl
    in:
      query: query
      db: makeblastdb/blastdb
    out:
      [output]
```

Example Workflow – Inputs

```
cwlVersion: v1.0
class: Workflow
requirements:
  - class: SubworkflowFeatureRequirement
inputs:
  query : File
  fasta : File
outputs:
  result:
    type: File
    outputSource: blastp/output
steps:
  makeblastdb:
    run: wf_makeblastdb.cwl
    in:
      fasta: fasta
    out:
      [blastdb]
  blastp:
    run: blastp.cwl
    in:
      query: query
      db: makeblastdb/blastdb
    out:
      [output]
```

Example Workflow – Output

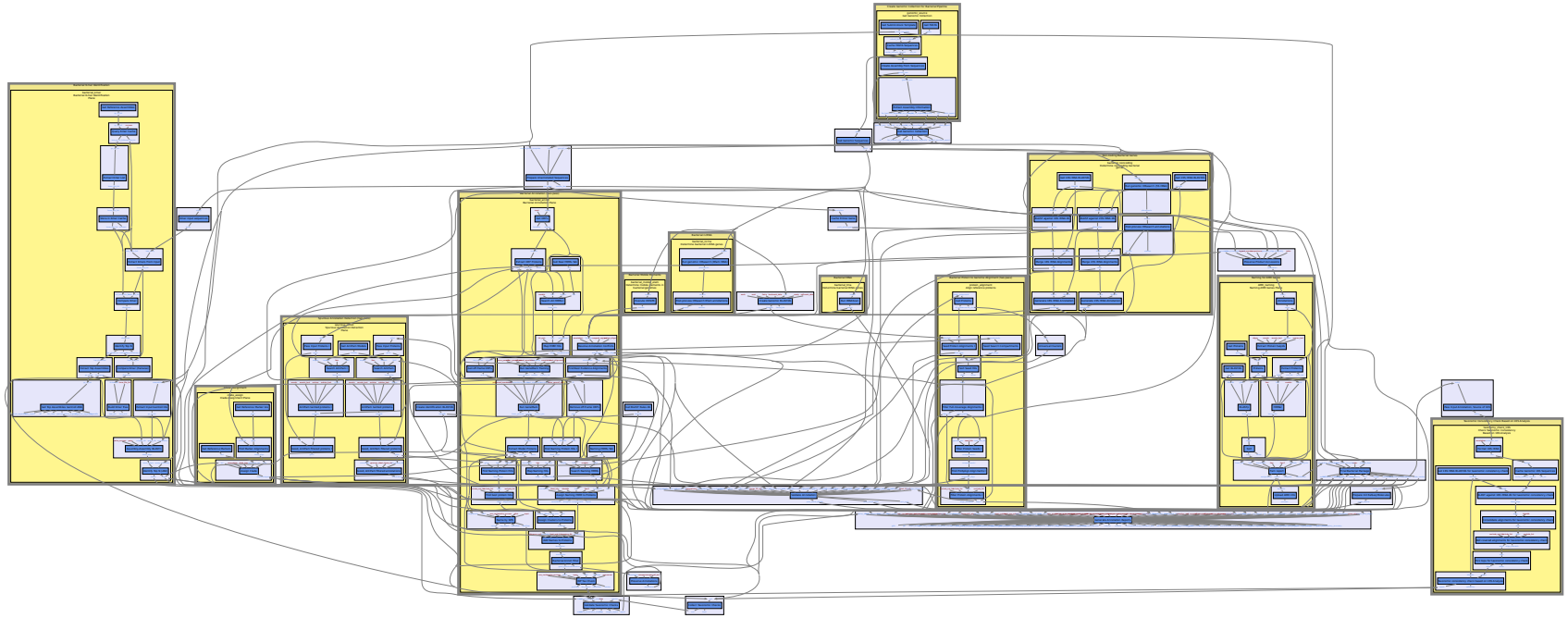
```
cwlVersion: v1.0
class: Workflow
requirements:
  - class: SubworkflowFeatureRequirement
inputs:
  query: File
  fasta: File
outputs:
  result:
    type: File
    outputSource: blastp / output
steps:
  makeblastdb:
    run: wf_makeblastdb.cwl
    in:
      fasta: fasta
    out:
      [blastdb]
  blastp :
    run: blastp.cwl
    in:
      query: query
      db: makeblastdb/blastdb
    out:
      [ output ]
```

Example Workflow – Link Steps

```
cwlVersion: v1.0
class: Workflow
requirements:
  - class: SubworkflowFeatureRequirement
inputs:
  query: File
  fasta: File
outputs:
  result:
    type: File
    outputSource: blastp/output
steps:
  makeblastdb:
    run: wf_makeblastdb.cwl
    in:
      fasta: fasta
    out:
      [ blastdb ]
  blastp:
    run: blastp.cwl
    in:
      query: query
      db: makeblastdb / blastdb
    out:
      [output]
```

Prokaryotic Genome Annotation Pipeline (PGAP)

https://github.com/ncbi-gpipe/pgap



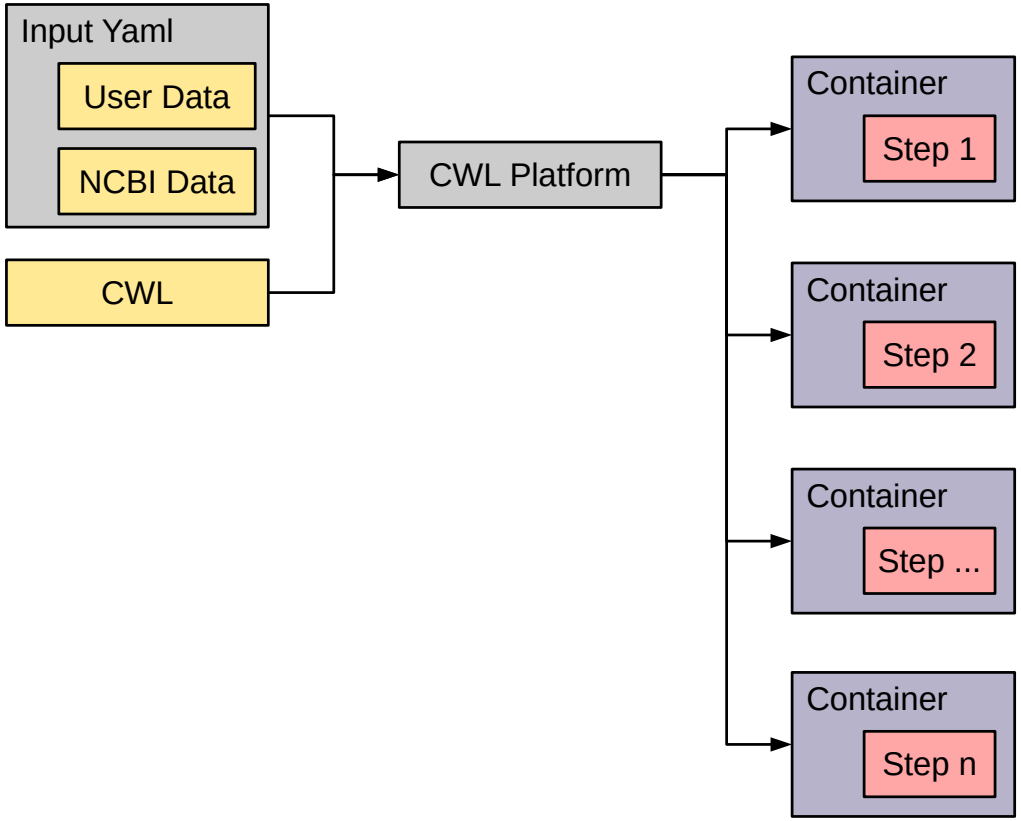
External PGAP Status

- 233 cwl files (on GitHub)
  - CommandLineTools: 144
  - Workflows: 83
  - ExpressionTools: 4
- Docker image is 6.4GB (on DockerHub)
- 307 supplemental datafiles, 30GB total (on S3)
- Builds automated via TeamCity

Implementation Timeline

- **February 2018** Steve Turner visits NCBI
- **September 2018** Proof of concept. PGAP executed by PacBio. Results conforming with internal PGAP
- **February 2019** Packaging of cwltool with the pipeline
- **June 2019** PGAP produces GenBank-ready files
- **August 2019** Annotation of 12,000 Enterococcus genomes on GCP

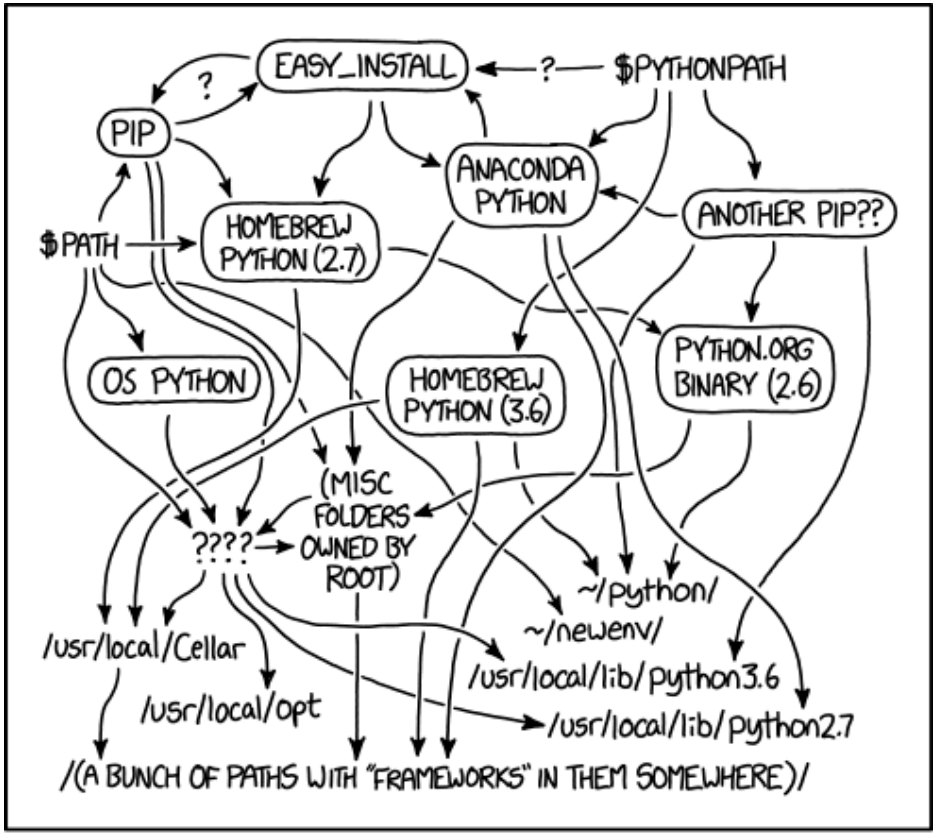
Running CWL


Input Yaml
User Data
NCBI Data
CWL
CWL Platform
Container
Step 1
Container
Step 2
Container
Step ...
Container
Step n

Running CWL

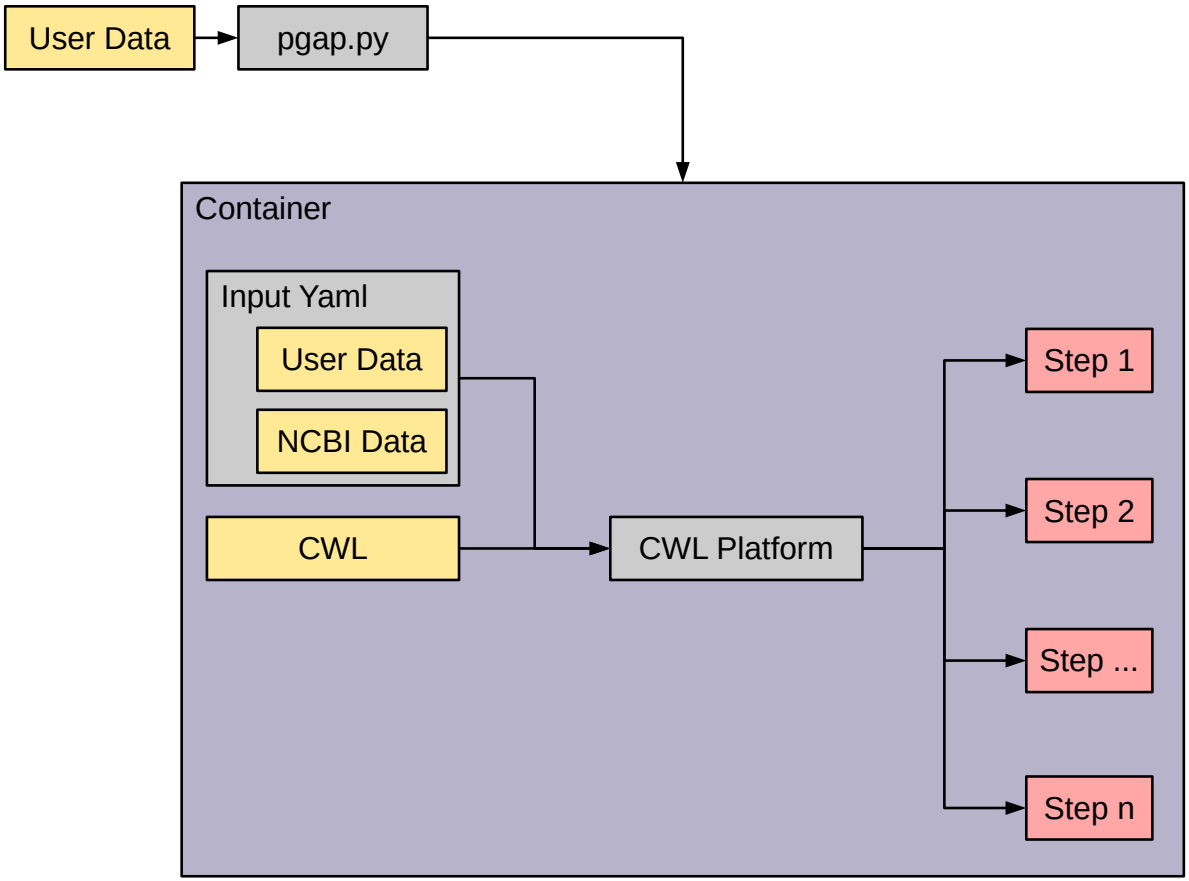
- The Platform (cwltool) is given CWL code and YAML file describing inputs.
- Executes each task in a separate container based on pgap-utils image.

Python is Easy!



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

PGAP CWL with Convenience Script



Running CWL the Easy Way

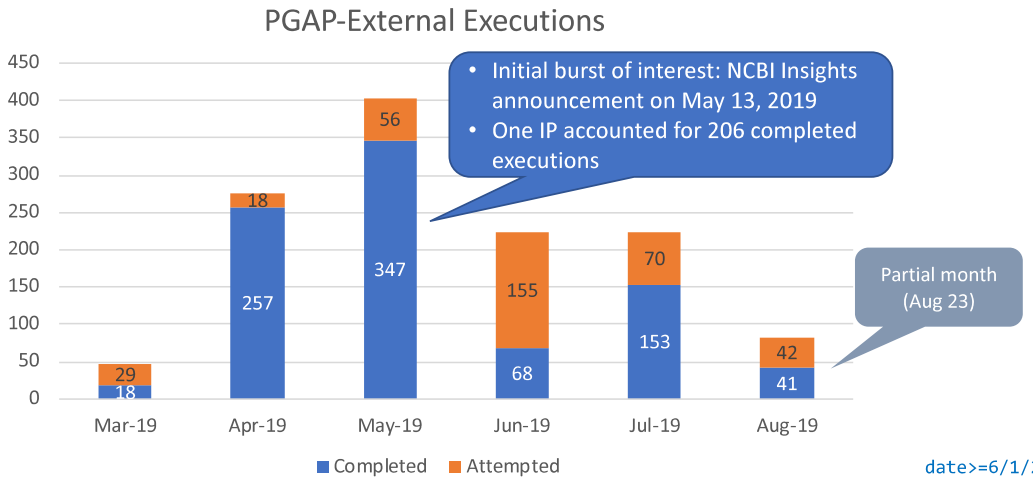
- User downloads pgap.py, which is generic Python 3
- User has access to Docker
- User runs pgap.py:
  - installs standalone docker image containing cwltool, CWL, and pgap-utils,
  - downloads and extracts supplemental data,
  - maps the paths to the standalone container,
  - creates the input.yaml file,
  - launches the pipeline.

PGAP-External Operational KPI

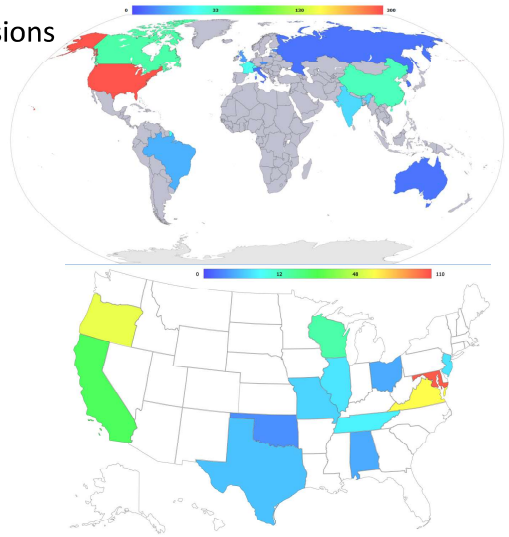
	March 2019	May 2019	August 2019
GitHub Forks	9	14	18
DockerHub Downloads			2.6K

< $0.38/genome
for *Enterococcus*

2 GenBank submissions
1 submitter



Geography
(June-Present)



`date>=6/1/2019 and app=stat and ncbi_app=pgapx and state=start`

Product

Microbial Virulence Hackathon

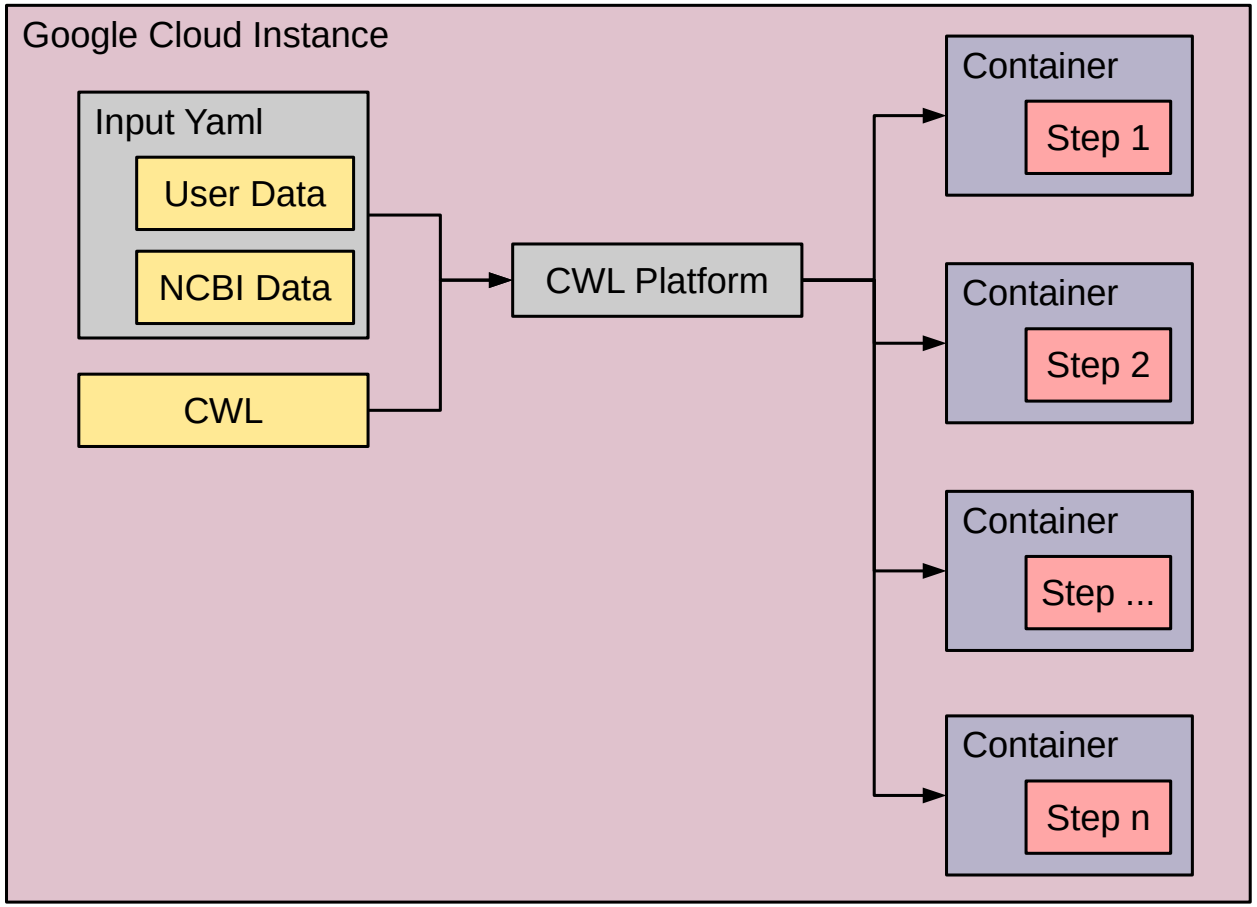
Task

Need to run approximately 15,000 Enterococcus assemblies through PGAP within a week.

Problem

Could not use the farm; it was already overloaded.

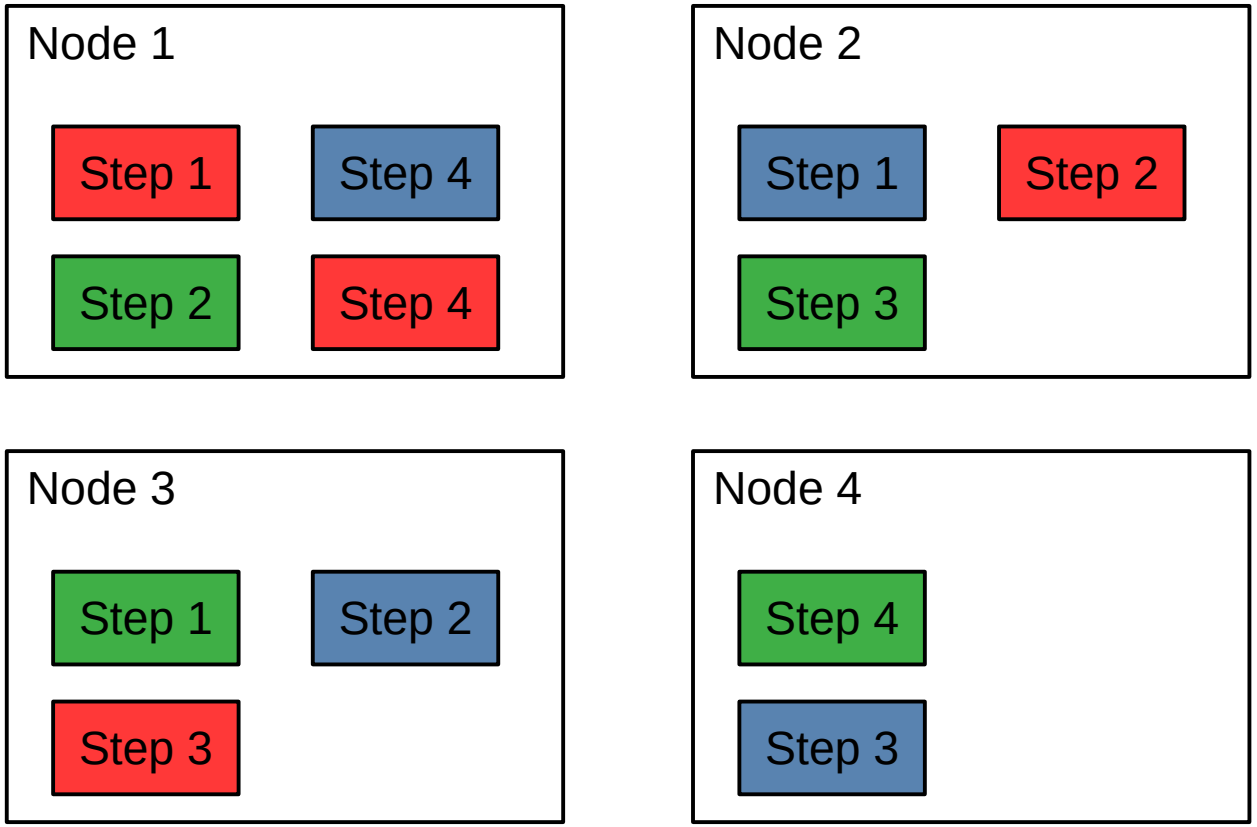
PGAP CWL on Google Cloud Platform



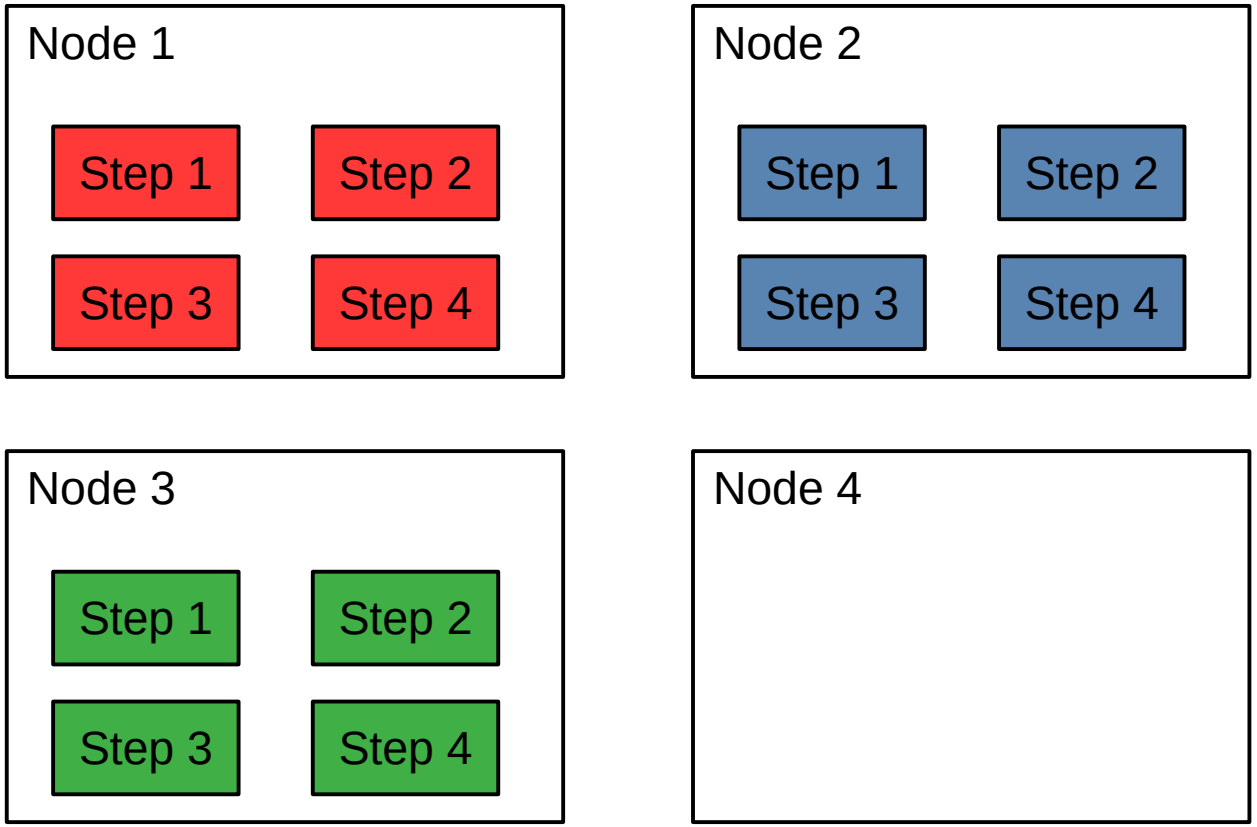
Running CWL for GCP

- Image created using packer which contains cwltool, cwl, pgap-utils, and supplementary data.
- Instance launched with startup script.
  - Pulls genome from Google storage based on metadata argument.
  - Run Pipeline
  - Log process in Google Stackdriver
  - Store result in Google storage
  - Halt instance (regardless).

Step Distribution on NCBI's Farm



Step Distribution on Google Cloud Platform



Cost

Persistent instances on the Google Cloud.

CPUs	Disk	Mem	per hour	avg hours	est. cost	x15,000
4	hdd	$0.137	15G	7.454475	$1.02	$15,319
4	ssd	$0.152	15G	7.318611	$1.11	$16,686
8	hdd	$0.270	30G	3.889568	$1.05	$15,753
8	ssd	$0.285	30G	3.844972	$1.10	$16,437

Preemptible instances on the Google Cloud.

CPUs	Disk	Mem	per hour	avg hours	est. cost	x15,000
4	hdd	$0.137	15G	7.454475	$0.33	$4,920
4	ssd	$0.152	15G	7.318611	$0.43	$6,477
8	hdd	$0.270	30G	3.889568	$0.33	$4,901
8	ssd	$0.285	30G	3.844972	$0.38	$5,710

What happened to the Consortium?

- What worked:
  - PacBio tested PGAP with 7 genomes.
  - DNAnexus did a simple CWL code review.
- What didn't work:
  - Other partners never materialized.
  - Unclear how consortium could help.
  - Illumina bought PacBio in November 2018
- Lessons learned?

Future Plans

- Add Averge Nucleotide Identity (ANI)
  - As another task in PGAP
  - Standalone version
- Support other containers, such as Singularity
- Integrate Fire-and-forget instances with GPipe
  - Allow choice of where to run
- Externalize FOOSH (16S)
- Externalize Foreign Contamination (FSCR)

Acknowledgments

Teamwork!

Teamwork, I personally do a tremendous amount of teamwork. . .

GPipe Group

- Azat Badretdin
- Slava Chetvernin
- ~~Rob Cohen~~
- Wratko Hlavina
- Boris Kiryutin
- Alex Kotliarov
- Eyal Mozes
- Daniel Soren
- Françoise Thibaud-Nissen

Cheerleaders

- Ben Busby
- ~~Lewis Geer~~
- Jim Ostell
- Kim Pruitt
- Steve Sherry
- Eugene Yaschenko
- Steve Turner