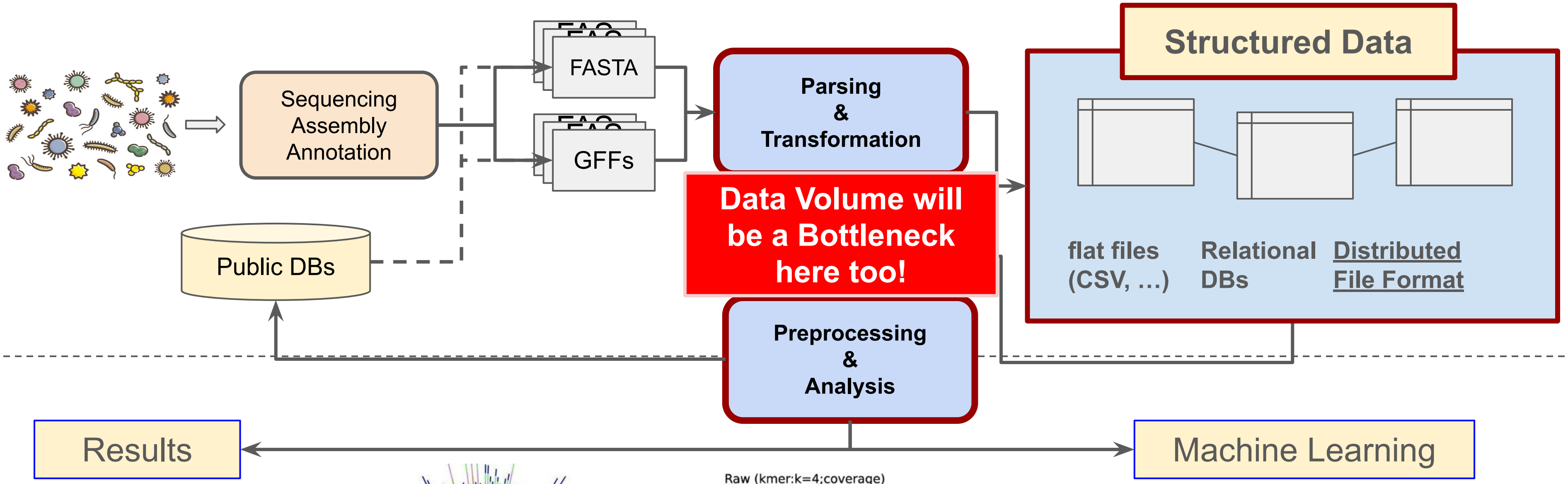
JGI JOINT GENOME INSTITUTE
A DOE OFFICE OF SCIENCE USER FACILITY

Scalable [Meta]genomic Data Analysis on Apache Spark

Satria Kautsar
Genome Analysis R&D scientist

October 4, 2024

What's after assembly & annotation?



Sequencing
Assembly
Annotation

FAA

FASTA

FAA

GFFs

Public DBs

Parsing
&
Transformation

Data Volume will be a Bottleneck here too!

Structured Data

flat files
(CSV, …)

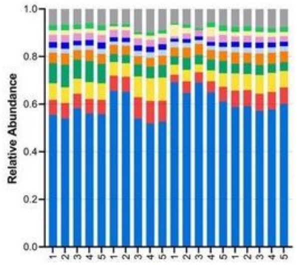
Relational
DBs

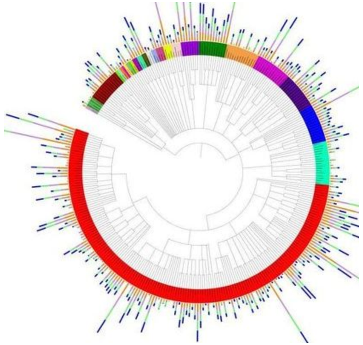
Distributed
File Format

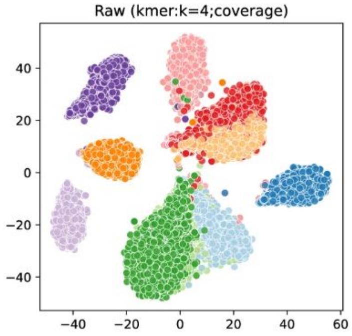
Preprocessing
&
Analysis

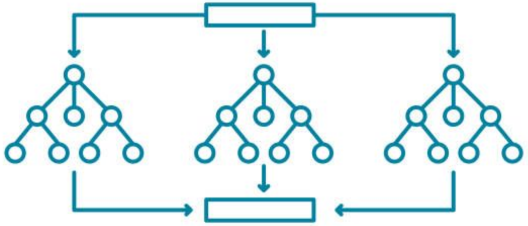
Results

Machine Learning

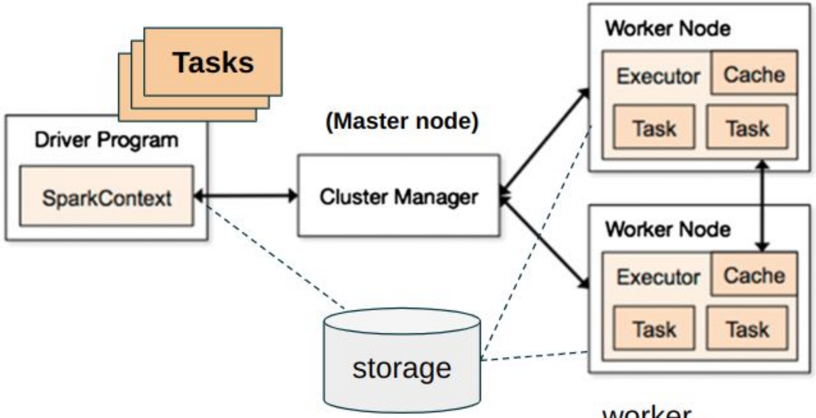






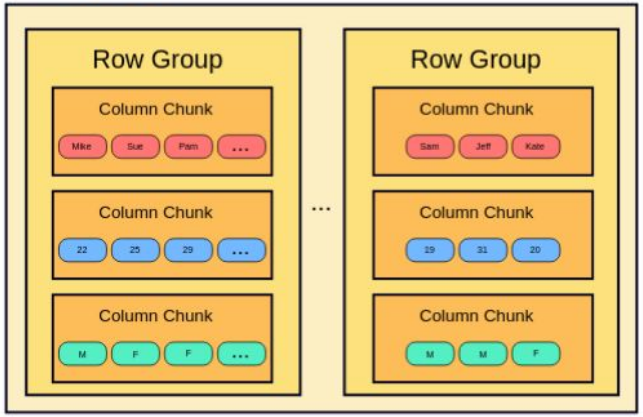


The solution: process & data scalability



worker$_3$,
worker$_4$,
…
worker$_n$

> **on-the-fly** scalability
> **fault tolerance**
> automated task optimization



> data is **partitioned** by **row groups,** then sorted by **column chunks**
> each chunk is **indexed** (including min, max values)
> resulted in a **distributable, compressible** data

The thing is…

Working with large-scale genomics data can be **messy**

- many **data formats**
- non-computer-friendly data formats
- non-compliant data entry

- data is increasing in an **exponential** fashion

Writing and troubleshooting a genome analysis pipeline can become **a nightmare**

Axolotl

https://github.com/zhongwang/axolotl

Genome Analysis **library** written in **Python** (on top of the **pySpark** library)

Defines a collection of **Standardized Tabular Data Structures** (i.e., Dataframes with schema) for all sorts of genomic data types (not file formats): NuclSeqDF, ProtSeqDF, cdsDF, etc…

Defines a collection of **Parser Classes** to process raw genomic files (FASTA, GFF, etc) into data tables (stored as Parquet)

Defines commonly used **functions** and **workflows** to preprocess, query, and analyze genomic data

Demonstrating Axolotl

SMC

SECONDARY METABOLISM COLLABORATORY

Dataset:

1,222,123 **genomic** (FASTA + GFF) files (bacteria, fungi, archaea) [8/22/2023]

- Originally sourced from NCBI (Genbank+Refseq) and IMG
- GFFs include CDS, **BGC** (antiSMASH + EMERALD) and other annotations
- Total size: FASTA = **5,017** GiB; GFFs = **8,285** GiB

>11,000,000 BGCs

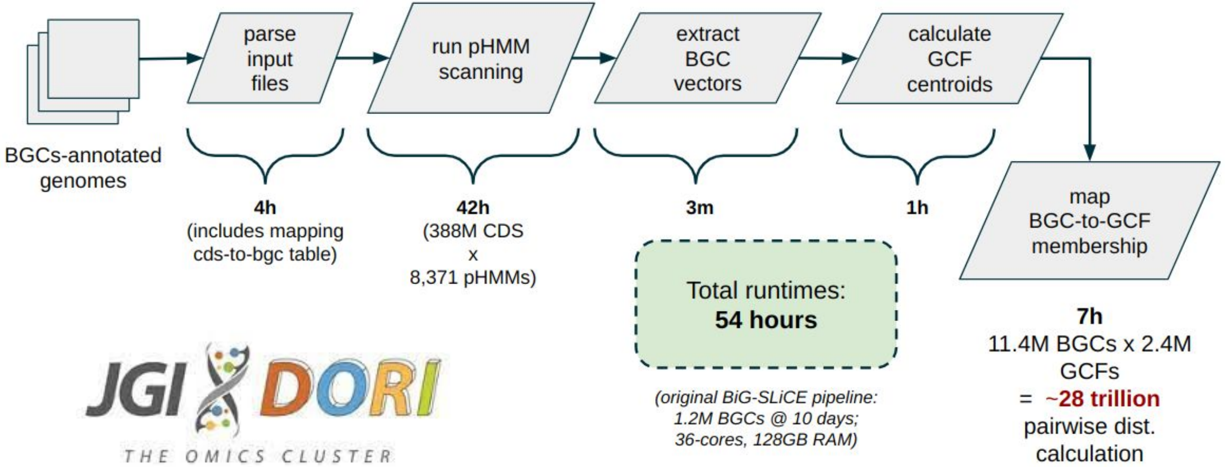
(**10 times** the size of state-of-the-art global analysis [Gavriilidou, 2022])

Tasks:

- **Parse** the raw input files and save as **Axolotl Tables** (stored in Parquet)
  - contigs
  - features => CDS, BGC
- Perform **clustering analysis** on the BGCs by replicating the "BiG-SLiCE" pipeline
- **Query** the Parquet "database" and generate some statistical summaries

Running the full "BiG-SLiCE" pipeline

Total resources used: <u>16 normal Dori nodes</u>
- 1x 16-cores, 32GB Node [Master & Driver]
- 63x 16-cores, 128GB Nodes [Workers]



BGCs-annotated genomes

parse input files

run pHMM scanning

extract BGC vectors

calculate GCF centroids

map BGC-to-GCF membership

4h
(includes mapping cds-to-bgc table)

42h
(388M CDS x 8,371 pHMMs)

3m

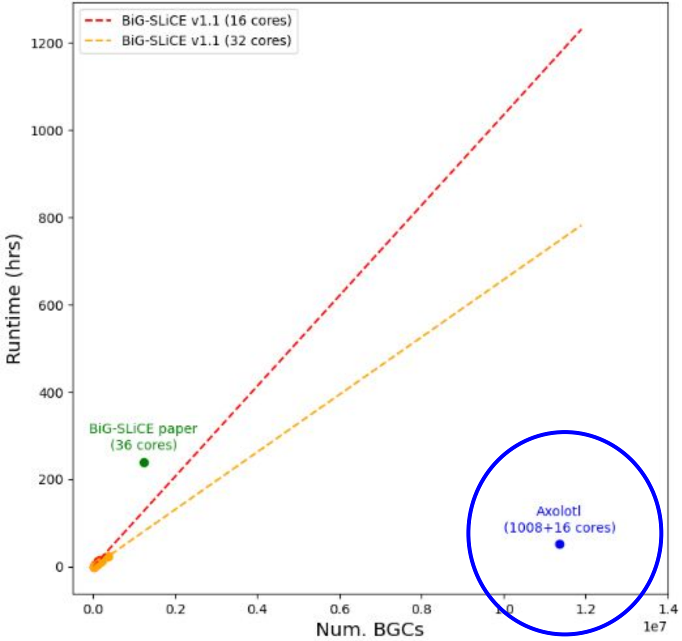
1h

7h
11.4M BGCs x 2.4M GCFs = **~28 trillion** pairwise dist. calculation

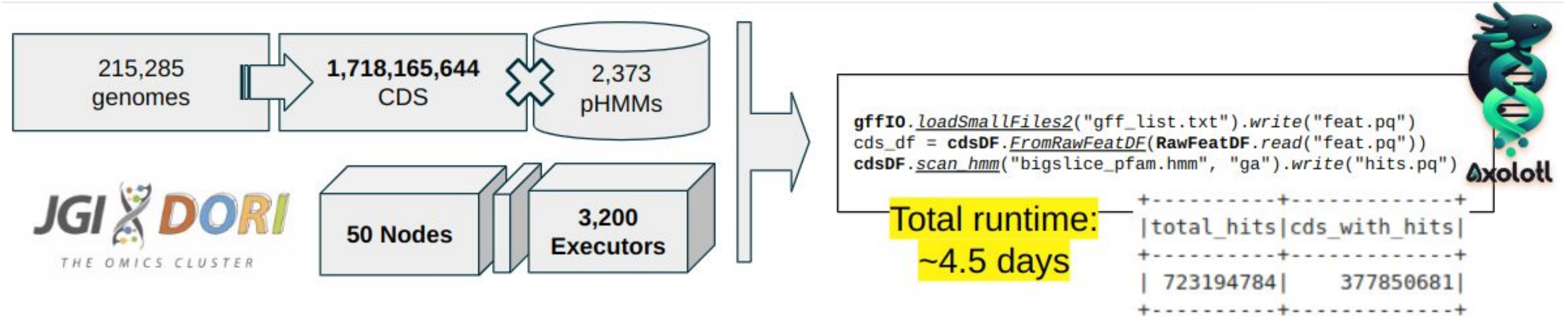
Total runtimes:
54 hours

(original BiG-SLiCE pipeline: 1.2M BGCs @ 10 days; 36-cores, 128GB RAM)

JGI DORI
THE OMICS CLUSTER



We survey genome-wide 'biosynthetic' genes of all RefSeq genomes…



215,285 genomes

1,718,165,644 CDS

2,373 pHMMs

JGI DORI THE OMICS CLUSTER

50 Nodes

3,200 Executors

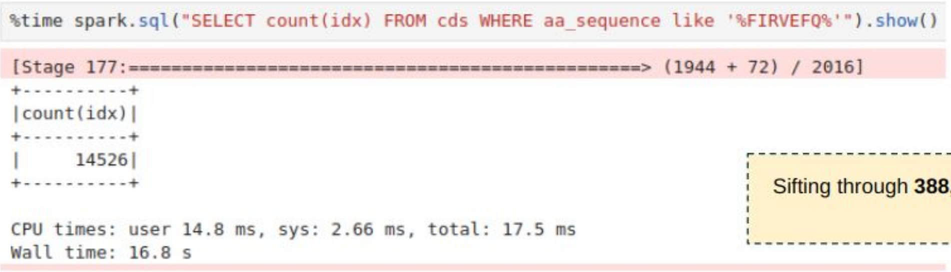
```
gffIO.loadSmallFiles2("gff_list.txt").write("feat.pq")
cds_df = cdsDF.FromRawFeatDF(RawFeatDF.read("feat.pq"))
cdsDF.scan_hmm("bigslice_pfam.hmm", "ga").write("hits.pq")
```

Axolotl

Total runtime: ~4.5 days

```
+----------+------------+
|total_hits|cds_with_hits|
+----------+------------+
| 723194784|   377850681|
+----------+------------+
```

Once you got your tables, the real fun begins…

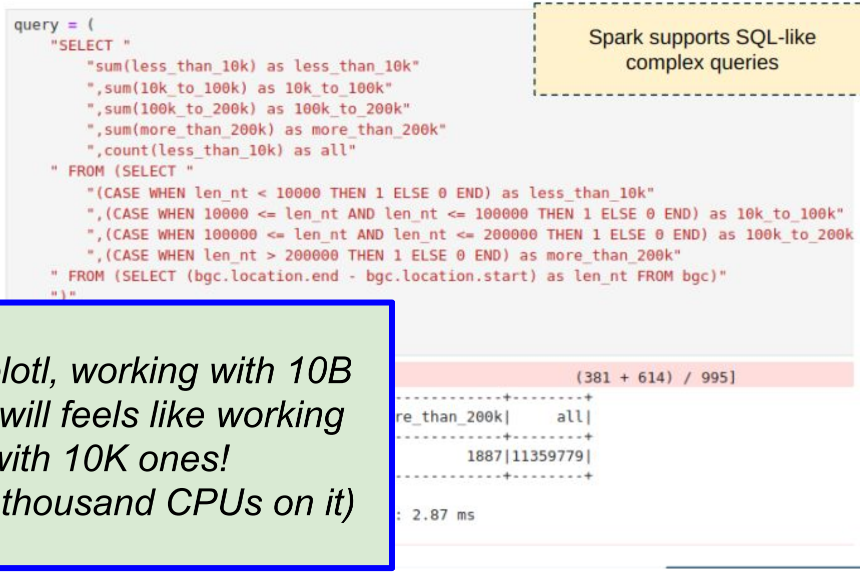


```
%time spark.sql("SELECT count(idx) FROM cds WHERE aa_sequence like '%FIRVEFQ%'").show()

[Stage 177:=================================================> (1944 + 72) / 2016]
+----------+
|count(idx)|
+----------+
|     14526|
+----------+

CPU times: user 14.8 ms, sys: 2.66 ms, total: 17.5 ms
Wall time: 16.8 s
```

Sifting through **388,075,840 BGC's CDS** and perform an AA substring match takes **17 seconds**



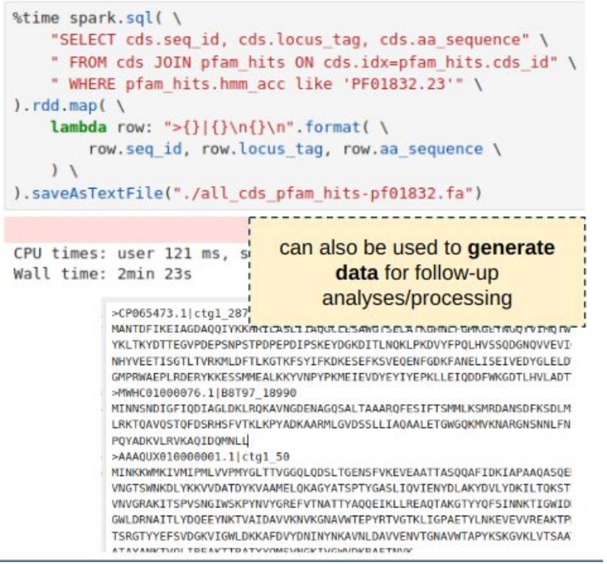
```
query = (
    "SELECT "
        "sum(less_than_10k) as less_than_10k"
        ",sum(10k_to_100k) as 10k_to_100k"
        ",sum(100k_to_200k) as 100k_to_200k"
        ",sum(more_than_200k) as more_than_200k"
        ",count(less_than_10k) as all"
    " FROM (SELECT "
        "(CASE WHEN len_nt < 10000 THEN 1 ELSE 0 END) as less_than_10k"
        ",(CASE WHEN 10000 <= len_nt AND len_nt <= 100000 THEN 1 ELSE 0 END) as 10k_to_100k"
        ",(CASE WHEN 100000 <= len_nt AND len_nt <= 200000 THEN 1 ELSE 0 END) as 100k_to_200k
        ",(CASE WHEN len_nt > 200000 THEN 1 ELSE 0 END) as more_than_200k"
    " FROM (SELECT (bgc.location.end - bgc.location.start) as len_nt FROM bgc)"
    ")"

                                    (381 + 614) / 995]
------------+--------+
re_than_200k|     all|
------------+--------+
        1887|11359779|
------------+--------+

: 2.87 ms
```

Spark supports SQL-like complex queries

with Axolotl, working with 10B CDSes will feels like working with 10K ones!
(just put thousand CPUs on it)



```
%time spark.sql( \
    "SELECT cds.seq_id, cds.locus_tag, cds.aa_sequence" \
    " FROM cds JOIN pfam_hits ON cds.idx=pfam_hits.cds_id" \
    " WHERE pfam_hits.hmm_acc like 'PF01832.23'" \
).rdd.map( \
    lambda row: ">{}|{}\n{}\n".format( \
        row.seq_id, row.locus_tag, row.aa_sequence \
    ) \
).saveAsTextFile("./all_cds_pfam_hits-pf01832.fa")

CPU times: user 121 ms, s
Wall time: 2min 23s
```

```
>CP065473.1|ctg1_287
MANTDFIKEIAGDAQQIYKK
YKLTKYDTTEGVPDEPSNPSTPDPEPDIPSKEYDGKDITLNQKLPKDVYFPQLHVSSQDGNQVVEVI
NHYVEETISGTLTVRKMLDFTLKGTKFSYIFKDKESEFKSVEQENFGDKFANELISEIVEDYGLELD
GMPRWAEPLRDERYKKESSMMEALKKYVNPYPKMEIEVDYEYIYEPKLLEIQDDFWKGDTLHVLADT
>MWHC01000076.1|B8T97_18990
MINNSNDIGFIQDIAGLDKLRQKAVNGDENAGQSALTAAARQFESIFTSMMLKSMRDANSDFKSDLM
LRKTQAVQSTQFDSRHSFVTKLKPYADKAARMLGVDSSLLIAQAALETGWGQKMVKNARGNSNNLFN
PQYADKVLRVKAQIDQMNLL
>AAAQUX010000001.1|ctg1_50
MINKKWMKIVMIPMLVVPMYGLTTVGGQLQDSLTGENSFVKEVEAATTASQQAFIDKIAPAAQASQE
VNGTSWNKDLYKKVVDATDYKVAAMELQKAGYATSPTYGASLIQVIENYDLAKYDVLYDKILTQKST
VNVGRAKITSPVSNGIWSKPYNVYGREFVTNATTYAQQEIKLLREAQTAKGTYYQFSINNKTIGWID
GWLDRNAITLYDQEEYNKTVAIDAVVKNVKGNAVWTEPYRTVGTKLIGPAETYLNKEVEVVREAKTP
TSRGTYYEFSVDGKVIGWLDKKAFDVYDNINYNKAVNLDAVVENVTGNAVWTAPYKSKGVKLVTSAA
```

can also be used to **generate data** for follow-up analyses/processing

Acknowledgements

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Zhong Wang
Harrison Ho
Bryce Foster
Fengchen Liu

JGI DORI
THE OMICS CLUSTER

JGI JOINT GENOME INSTITUTE
A DOE OFFICE OF SCIENCE USER FACILITY

Hands-on Workshop on Axolotl Library & Apache Spark

Satria Kautsar
Genome Analysis R&D scientist
October 4, 2024

What to expect:

You will:

- Get introduced to Apache Spark and its primary components
- Walk through example how to parse, process and analyze FASTA+GFFs data using Spark and Axolotl
- Small exercises to keep you engaged
- Chance to explore and ask questions!

Preparation

- Login and create a Dataproc Cluster (covered by Steven)
- Open up the JupyterLab Interface
- Open a new Terminal screen

```
> cd /
> git clone https://github.com/JGI-Bioinformatics/jgi-scalable-toolkit-workshop-24.git
> gsutil cp gs://zw_axolotl/jgi_workshop_2024/data/phmms.zip /phmms.zip
> unzip phmms.zip
```