Cloud Testing for Bioinformatics Recommendations Report

June 28, 2024

Report Prepared By

Starlo Innovation

Cloud Testing By

Starlo Innovation

NOAA Fisheries, NOAA OAR

Table of Contents

Executive Summary

Background: This document details a pilot project aimed at evaluating the cost and effort of running 'Omics analyses in a cloud environment compared to on-premise high performance computing clusters (HPCCs). The subsequent report explores the feasibility of using commercial cloud environments for bioinformatics workflows.

Approach: We benchmarked four workflows, three written in the workflow language Snakemake, and one in BASH. Snakemake is the primary workflow manager used within NOAA Fisheries. We tested a wide range of cloud-based execution environments including single virtual machines, batch executors, and cloud-based HPCC using both Verily Workbench and Google Cloud Platform.

Findings: SLURM HPCC in the cloud requires the lowest level of effort, and while more expensive than the other methods we tested, costs could be managed using alternative storage strategies. Single virtual machines scaled poorly to production size workloads, and Batch execution with the Life Science API or Google Batch would require a very high level of effort to get running successfully. Platform as a Service options were a poor fit for the workflows run by NOAA Fisheries.

Future Considerations: For the bioinformatic workflows run by NOAA Fisheries, deploying a centrally managed SLURM cluster in the cloud will provide the most flexibility and lowest level of effort to get running. Google Batch should be revisited as Snakemake development progresses. We recommend SLURM HPCC over Verily Workbench or other Platform as a Service (PaaS) offerings due to the flexibility to run Snakemake and other workflows.

Collaboration and Support: Ongoing collaboration between cloud engineers and bioinformaticians is necessary to ensure cloud environments are optimized for a wide variety of workflows. Ongoing and centralized support is essential as bioinformatic tools and cloud infrastructure changes, and as bioinformatics scientists of all levels of experience are engaged in analyses to make the most efficient use of compute and storage resources.

Background

The NOAA 'Omics Strategic Plan prioritizes the generation and analysis of large molecular and chemical data. On-premises servers or High-Performance Computing (HPC) environments are currently utilized and appropriate for bioinformatics workflows, but as demand increases and as infrastructure ages, commercial Cloud environments may provide necessary infrastructure for bioinformatics processing into the future. Under a NOAA High-Performance Computing and Communications (HPCC) funded project (funding in FY23), NOAA partnered with Starlo, the Broad Institute, Verily, and Google Cloud to conduct a pilot project to test the feasibility of running 'Omics analyses in a cloud environment and compare the cost and effort with those of on-premise HPC. In particular the original proposed work sought to:

- Objective 1: Evaluate Terra as a Platform for 'Omics Analysis.
- Objective 2: Compare Cost and Effort Required in GCP vs. Local Servers.
- Objective 3: Make Recommendations for NOAA Cloud Computing.

The results of this pilot project and collaboration between NOAA and Starlo will provide documentation on required storage space, processing power, and time spent on installation and troubleshooting to provide a robust framework for establishing a cost-benefit analysis of cloud vs. on-premises computing. This will inform long-term planning for effective future storage and analysis of 'Omics datasets.

The initial proposal sought to achieve these aims by benchmarking several workflows on Terra. In early planning meetings, it was decided that our team would instead benchmark on Verily Workbench (VWB) and a native Google Cloud console (with focus on SLURM) because a majority of NOAA workflows are either written in Snakemake or Bash. Neither of these workflow types are well supported in Terra (see below), which would have required rewriting workflows in a different language. As such, this study expanded from the initial proposal to test additional questions. Given the current preference for a) Snakemake and b) on-premise SLURM:

1) Should NOAA adopt a Platform as a Service (PaaS) solution, or build a custom solution directly on Google Cloud Console?
2) Is the SLURM offering from Google Cloud a sufficient replacement for existing HPCC resources on Sedna?
3) Are current NOAA Snakemake and Bash workflows cloud-ready?

Our benchmarking began in VWB, then moved to a Google Cloud Console in the latter half of the project.

Using Cloud-Native Solution vs. Platform as a Service (PaaS) Offering to Execute Bioinformatic Workflows

A bioinformatic workflow refers to a structured sequence of data analysis that involves the use of various computational tools and methods to process data and produce meaningful insights such as species composition or amount of genomic variation. Modern workflows are primarily written using workflow languages such as Snakemake, Nextflow or Workflow Description Language (WDL), although many legacy and ad hoc workflows still use Bash, Python or similar language.

In the cloud, bioinformaticians can execute workflows using either the native cloud console or Platform as a Service (PaaS) platforms. Cloud consoles offer more flexibility and control but require much higher up front effort to build the infrastructure and need to comply with NOAA security policies. Building a custom solution on the cloud console also requires a minimum level of cloud engineering expertise. PaaS solutions abstract away much of the setup and challenges from bioinformaticians but provide less flexibility. PaaS platforms also run best using standardized workflows and reference datasets, which is inefficient for many routine bioinformatic analyses conducted within NOAA, and would require substantial rewriting of existing workflows. PaaS platforms support workflow languages based on perceived demand; most platforms currently support Nextflow, WDL, and Common Workflow Language (CWL) rather than Snakemake. In deploying cloud solutions, organizations are faced with important decisions to balance ease of use with the flexibility of the platform for diverse workflows, and the time, effort, and available expertise to design workflows in a specific workflow language (Figure 1).

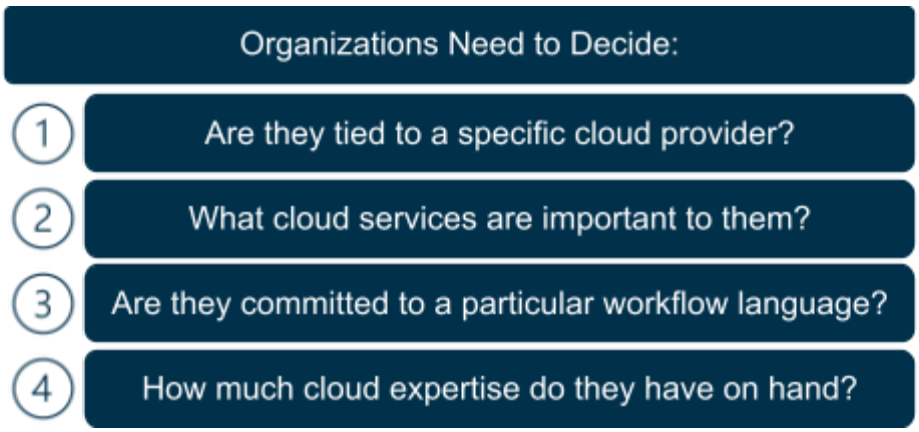

Figure 1: Decisions NOAA needs to make regarding adopting a cloud console vs. PaaS solution.

Currently, the three big Cloud Service Providers (CSPs) are Amazon Web Services (AWS), Google Cloud, and Microsoft Azure. Only AWS has its own dedicated genomics platform, AWS HealthOmics, which supports Nextflow, WDL, and CWL. AWS only supports Snakemake (see description below) via AWS Genomics CLI.

PaaS stacks include Terra, Verily Workbench (VWB), Velsera (formerly Seven Bridges), DNAnexus, Nextflow Tower, and others. Each PaaS stack is supported on different cloud providers and has different levels of FedRAMP approvals. For example, Terra is supported on both GCP and Azure, while VWB is only hosted on GCP. VWB supports a wider range of services than Terra, including Nextflow and WDL support.

Terra is a cloud-native platform for biomedical researchers to access data, run analyses, and collaborate in shared workspaces. Terra natively runs WDL workflows and is hosted on GCP and Azure. Terra comes preloaded with many datasets, but these are primarily biomedical focused. Terra plans to support Nextflow in the future, but Snakemake is not on the roadmap for support. Terra also has a well-developed Command Line Interface (CLI) that enables interacting with cloud resources from the command line, including creating new virtual machines, submitting jobs to the batch APIs, and interacting with containers from the Artifact Registry.

Verily Workbench is a secure PaaS solution designed for governing and analyzing global multimodal biomedical data; it supports the end-to-end lifecycle of data and enables secure, collaborative research by integrating multiple data sources and providing tools for diverse user expertise levels. VWB is hosted on GCP and uses the Terra CLI to orchestrate tasks, meaning users can still use Terra commands with VWB. Verily also developed their own CLI for orchestrating Verily Workbench-specific processes on GCP. We go into more detail on VWB below.

Seven Bridges Platform (owned by Velsera) is a precision medicine platform that supports drug discovery and clinical trial optimization with a secure, scalable cloud environment, fostering collaboration and reducing data silos. Seven Bridges supports multiple workflow languages, but Common Workflow Language (CWL) is the best-supported. Seven Bridges is cloud agnostic and allows users to submit to several cloud platforms.

DNAnexus is a fast and versatile platform for bioinformatics, supporting multiple workflow languages and providing both command-line and intuitive user interface options. It accelerates pipeline deployment and analysis with pre-built tools and apps, catering to the needs of both technical and non-technical users in wet and dry lab environments. DNAnexus is hosted in AWS, although the user does not interact with the underlying cloud infrastructure.

Seqera Platform (formerly Nextflow Tower) is a centralized platform for managing and scaling data analysis pipelines across various environments, from on-premises clusters to public clouds. It supports flexible pipeline execution, secure collaboration, and seamless integration with cloud storage. Seqera Platform only supports Nextflow execution, but can be deployed on any cloud provider and allows users to submit to and manage jobs on any of the three cloud providers as well as on-premises clusters.

Overview of Snakemake in the Cloud

Snakemake is a versatile workflow management system designed to facilitate reproducible and scalable data analyses, and is the most commonly used

workflow language at NOAA Fisheries. It allows workflows to be described using a human-readable, Python-based language which can be scaled across various computing environments—including servers, clusters, and clouds—without requiring modifications to the workflow definition. This feature makes Snakemake particularly suitable for large-scale data analyses in bioinformatics, particularly for routine workflows or processes. However, Snakemake is not well supported by cloud platforms or cloud-based PaaS stacks.

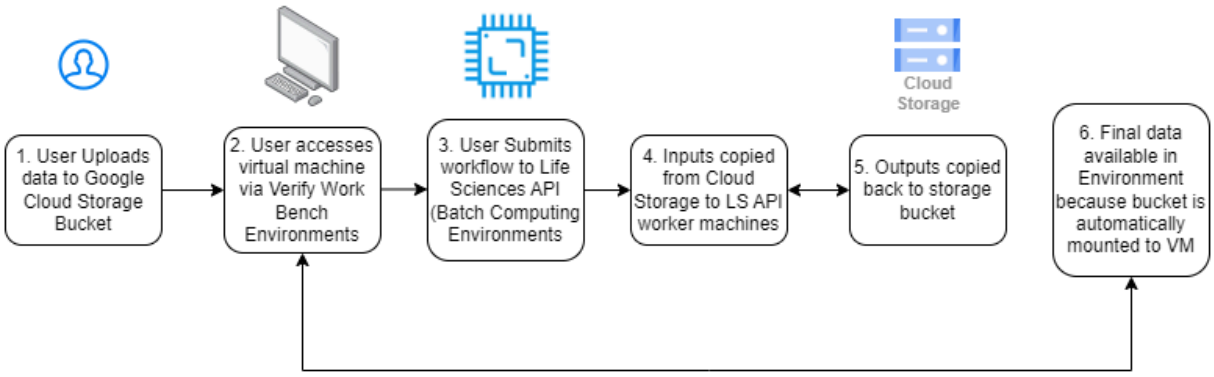

Figure 2: Example Snakemake workflow execution in Verily WorkBench using the Google Life Sciences API.

Benefits of Snakemake

One of the main advantages of Snakemake is its portability and flexibility. Unlike many other workflow management systems that require tight integration of tools, Snakemake can interoperate with any installed tool or web service with well-defined input and output formats. This is facilitated by the use of a domain-specific language in a 'Snakefile' that closely mirrors standard Python syntax, making it accessible and easy to use for developers familiar with Python. Snakemake supports automatic scalability, optimizing the number of parallel processes based on available CPU cores and threads, and can operate efficiently on single machines as well as on HPCCs.

Snakemake has made many recent advances with the addition of execution and storage plugins that allow workflows to be more portable across environments, as well as the adoption of containers for workflow execution instead of only conda environments. While benchmarking on Verily

Workbench, our team began with Snakemake v7 which supports submitting to the Life Sciences API but did not yet include plugins. We then moved to Snakemake v.8+ on the Google Cloud Console, thus allowing us to use execution and storage plugins. Snakemake v.8+ no longer supports the Google Life Sciences API (which has been deprecated); thus, our team instead tested Google Batch. Likewise, we tested the SLURM executor plugin in Google Cloud on Google hpc-toolkit.

Limitations of Snakemake

Though the containerization and the introduction of storage and execution plugins allow Snakemake to handle various execution environments more efficiently, it can still be challenging to easily port Snakemake workflows between compute environments. For example, one workflow our team was testing runs well on a local machine or HPC cluster but encounters many problems with remote execution on Google Life Sciences API or Google Batch. Furthermore, while Snakemake supports Google and Azure Batch, it does not support AWS Batch, limiting its usability on AWS unless the user employs the AWS Genomics CLI.

Snakemake Best Practices

Successful cloud migration requires Snakemake workflows to be written following Snakemake best practices. Some of these are outlined in the Snakemake documentation, including the use of Snakemake wrappers, use of config files to define common parameters and profiles to define environment-specific parameters, and following common organization structures such as having a workflow direction containing rules, scripts, environments etc, as well as a config directory containing configuration files.

With the advent of Snakemake version 8, our team recommends a few additional practices, including the use of workflow or rule-specific containers rather than conda environments wherever possible, and the use of executor and storage plugins. Examples of best practices workflows can be found on the Snakemake standardized workflows page. The workflows we benchmarked used a mix of these best practices, and found that how well a workflow follows these practices was a strong predictor of success for cloud migration.

Overview of Various Compute Environments Benchmarked in this Study

Verily Workbench

Verily Workbench allows users to run analyses using a simple VM with mounted buckets as well as using batch execution through the Google Life Sciences API. Google Batch integration is on the platform's roadmap but is not yet fully supported. Verily Workbench uses much of the same underlying infrastructure as Terra and as such offers many similar features.

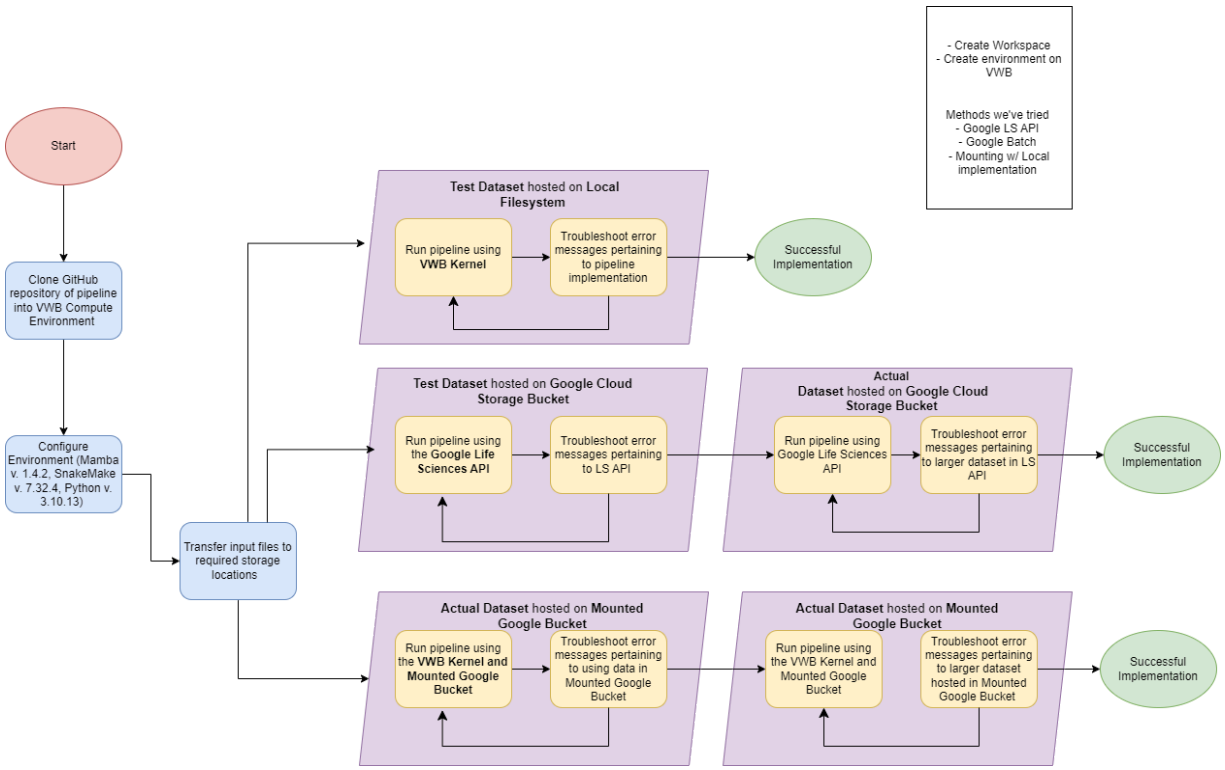

Figure 3: Verily Workbench (VWB) Implementation Flowchart

Google Cloud Console

Using the Google Cloud console offers more flexibility, but also requires more effort to stand up and maintain the environment. Users can run bioinformatic

workflows using a variety of methods, three of which we benchmarked in this study:

- Using a simple VM in Compute Engine or in a notebook via VertexAI Workbench.
- Using remote batch execution with Google Life Sciences API (deprecated) or Google Batch (not yet FedRamp-approved).
- Via SLURM cluster, which can be spun up using Terraform code from the Google Cloud hpc-toolkit.
- Via Kubernetes cluster, which Snakemake supports via plugin but was not tested in this project because it offers a similar offering to SLURM.

SLURM on GCP

SLURM is a free and open-source job scheduler widely used in supercomputers and computer clusters. It facilitates resource allocation, job execution, and workload management without requiring kernel modifications. Cloud providers like AWS, Azure, and Google offer SLURM through implementations such as ParallelCluster, CycleCloud/AzHop, and the HPC Toolkit, respectively. While Google's SLURM setup may entail configuration challenges in NOAA environments, SLURM's scalability and portability make it valuable for migrating from on-premise bioinformatics analyses to cloud environments. SLURM offers efficient job scheduling and resource allocation and can seamlessly run both Snakemake and Bash workflows.

Benchmarking Materials and Methods

Overview of Bioinformatic Workflows

Our team evaluated four bioinformatic workflows in the Google Cloud selected for their variation in CPU, RAM, and storage requirements. All scripts used for analysis can be found at our GitHub repository. Our four workflows included:

- Non-Model Whole Genome Variant Calling is a Snakemake workflow written by Eric Anderson that uses the common GATK variant calling pipeline. This workflow requires low CPUs and RAM to execute hundreds of processes. This workflow exemplifies many Snakemake best practices, including modularity and the use of compute profiles. The author is currently in the process of upgrading many of the components from Snakemake v.7 to v.8+ as many new changes have been integrated. This workflow includes test data that can run in minutes on a local laptop. The full test dataset included 43 whole rockfish Illumina short read genomes.

- Tourmaline is a Snakemake workflow to conduct DNA metabarcoding with Illumina sequencing amplicon data. The workflow uses QIIME 2 and the software packages it wraps. Tourmaline has moderate CPU (8 CPU) and RAM requirements but has a large number of long-running processes. DNA metabarcoding is a common use case within NOAA. In addition to the test data distributed with Tourmaline, our team benchmarked DNA metabarcoding data from eDNA collected from the Gulf of Alaska (GoA) amplified using 12S rRNA targeted Illumina short read sequencing.

- Amethyst is a Snakemake workflow to conduct metagenomic analysis using whole genome shotgun sequencing with Illumina data, and is used to characterize microbial communities in an environmental sample. It has moderate CPU and high memory requirements. It was written during this project by Rachael Storo and Luke Thompson. The dataset our team benchmarked includes 88 seawater metagenomes from the North Pacific, Southern California, and Florida, collected as part of the Bio-GO-SHIP project and sequenced using Illumina short reads.

- Trinity is a program that conducts transcript assembly from Illumina RNA sequencing data. Trinity has very high CPU (20+) and RAM (200 GB) requirements and was run using a simple Bash script rather than using Snakemake. Many (if not most) workflows currently run by NOAA scientists are not written using workflow languages; thus, our team wanted to benchmark a simple Bash example in addition to the

Snakemake workflows. The dataset used for this benchmarking consists of 20 Illumina short read steelhead trout RNA samples.

Objective 1: Evaluate Terra as a Platform for Omics Analysis

Choice of Verily Workbench as a PaaS Stack

Terra was initially proposed for use in this project. However, Terra as a platform only supports WDL workflows and would have required re-writing or configuring workflows into a different workflow language for our cloud testing. After initial discussion among NOAA stakeholders, it was decided that Snakemake and Bash were the primary workflow languages used within the NOAA 'omics community. As such, the team decided to investigate Verily Workbench as an alternative PaaS stack because VWB supports running Snakemake via the Google Life Sciences API as well as locally in Jupyter notebook instances. Because VWB supports the Terra CLI, we were able to test some of the same functionality as on Terra.

Evaluation Methods

Although VWB supports running workflows natively, it only supports WDL and Nextflow using this feature. Because our team was focusing primarily on evaluating workflows that had been previously designed in Snakemake, we tested the following features in VWB:

- Provisioning workspaces: Workspaces are project-specific areas that organize all resources and workflows for that project. Workspaces include Resources, Environments, and Workflows. Our team limited our workspaces to a NOAA users group, although they can be open to all users on the platform and can enable collaboration with any user within a walled environment.
- Resources are the VWB way of organizing Cloud Storage Buckets. Our team added our data to Resources from Google Drive using Rclone. Once data was uploaded to the cloud, our team was able to move it between compute environments and even between projects using the Gcloud CLI.

- Environments include computing workspaces. Our team used Environments to create JupyterLab instances with various machine sizes, allowing us to compute on a single Virtual Machine and interact with other services such as Google Life Sciences API, Artifact Registry, or BigQuery. Environments allow users to spin up JupyterLab, JupyterLab Spark, RStudio, and Visual Studio Code instances. Users can launch Environments from preloaded machine images or from custom machine images. For example, our team launched our environments from Vertex AI machine images because the default machine image had Python 3.7 and we wanted a newer version of Python to ensure compatibility with the newer Snakemake tools.

- Once in our Jupyter environments, our team submitted jobs to the Google Life Sciences API and tested running Snakemake workflows locally on the Environment's virtual machine. This allowed us to test performance on a single machine with numbers of CPUs ranging from 4 to 32.

- Environments are launched from a startup script as compute engine instances. This startup script pre-installs many helpful tools such as the Terra and Verily CLIs, but it could be modified to pre-install Snakemake or other CLI tools. Further, Environments mount selected bucket paths and can clone in GitHub repositories upon launching, allowing for quick and easy access to data without having to copy back and forth from Cloud Storage.

- Finally, Workflows allow users to run WDL and Nextflow workflows in a managed environment. Our team did not test workflows in this project because benchmarking focused on Snakemake and Bash which are not supported by the Workflows feature.

Verily has a host of other features that were not rigorously tested in this project because they were not immediately necessary for our benchmarking effort. However, they may have applications for other NOAA use cases. Verily includes its own CLI, as well as the Terra CLI, neither of which our team rigorously evaluated because for our benchmarking we were able to conduct all necessary tasks using the GUI. Further, our team did not use Data

Collections or Data from Catalogs but instead stored all project data in a single Cloud Storage bucket. If NOAA were to adopt Verily, these data collections could store shared resources such as reference genomes or test data files that would easily be accessed by many users.

Benefits of Verily

Verily offers numerous benefits, including quick access for users looking to streamline their workflows without needing to set up Google Projects from scratch. With a well-developed Command Line Interface (CLI) that builds on Terra and Google Cloud tools, users can efficiently manage infrastructure and cloud resources. Moreover, Verily provides the flexibility to leverage containers and much of the functionality of Terra and Google Cloud Platform (GCP), ensuring compatibility and versatility. The option to mount buckets and GitHub repositories to virtual machines adds convenience, enhancing data accessibility and management. Additionally, Verily simplifies the setup process with a pre-configured startup script that installs a wealth of handy tools, further optimizing user experience and productivity. Our team chose not to modify this startup script because it had most of what we already needed, but a user could add additional tools to the startup script such as Mamba or Snakemake.

Limitations of Verily

Though offering certain advantages such as mounted buckets, a user-friendly interface, and shared workspaces, Verily also presents several limitations. Firstly, cloud costs were not clearly presented, making it challenging for users to accurately assess and manage expenses. Our team had to go directly to the Google Cloud console to view our billing report. In addition, Verily requires an annual licensing fee of $75,000–$250,000 depending on the number of users and level of support, meaning that annual costs of the platform are higher than a standard Google Cloud Project. Secondly, the range of services is limited as it is subject to the discretion of Verily, restricting users' flexibility and autonomy. For example, VWB currently supports Google Life Sciences API but not Google Batch.

In addition, Verily's compatibility is primarily tailored towards Workflow Description Language (WDL) and Nextflow (NF) users and thus is not optimized for those utilizing Snakemake or bioinformatics workflows not written in these languages. Particularly for Snakemake v.8+, the environment required a lot of custom configuration to get things running correctly because of the version of Python installed on the Vertex AI images (Python 8) due to version conflicts with other pre-installed tools.

Our team focused much of our benchmarking efforts on getting workflows to run successfully using the Google Life Sciences API. Jobs submitted via Snakemake to the Google Life Sciences API require a base container that is used for each virtual machine. A Cloud Storage Bucket is mounted and then data required for a task is copied from the Bucket to the virtual machine to execute the task, then files are copied back to the Bucket when the task completes and the virtual machine is spun down. Our team encountered a few challenges with this remote execution setup:

- For workflows that did not have existing containers our team often encountered version conflicts between task-specific bioinformatic software and the base container. To give an example, the version of Samtools specified by a workflow may be incompatible with some software within the Snakemake 7.1.1 container, causing the workflow to crash.
- When the bucket was mounted, the new data path became /data/BUCKETNAME. Many of the workflows our team tested assumed hard-coded paths from a local file system such as /home/data/file.R1.fastq/gz. To use the remote execution, our team had to change the paths in the configs to point to the expected paths of the Google Life Sciences API, and inconsistencies in naming often led to workflows crashing prematurely. Snakemake can now support reading from Google Cloud Storage using the new storage plugin, but it does not yet allow users to point config files directly to gs://PATHS and this new storage plugin is only compatible with Google Batch, not the Life Sciences API.

Summary of Verily Workbench

Verily excels at simplifying the cloud experience while still providing powerful CLI capabilities and supporting native workflow execution. It is still early in development, and many new tools will likely be rolled out in the future. Verily includes many helpful features such as auto-mounting buckets to compute environments and allowing users to interact with GCP services like Artifact Registry, BigQuery, and Google Life Sciences API without having to navigate the console. Although it shares many features with Terra, it also supports the launching of virtual machines with Juptyer, RStudio or VS Code, and it supports the native execution of both WDL and Nextflow Workflows. Despite these advantages, Verily is not natively configured for Snakemake or Bash workflows and requires custom configurations or very specific versions of Snakemake to execute correctly. Finally, our team outlined above several challenges with Snakemake and the Google Life Sciences API, suggesting that for many Snakemake workflows, the effort required to get the workflow running on the Google Life Sciences API may not be worth the attempt.

Objective 2: Compare Cost and Effort Required in GCP vs. Local Servers

Most bioinformatic analysis currently conducted within NOAA occurs using on-premises SLURM clusters. Needs and tools for bioinformatics analyses are broad within NOAA, ranging from applications to population genomics and assessment of genetic diversity in whole re-sequenced genomes, to DNA metabarcoding of environmental samples, to evaluation of gene expression in response to environmental or experimental variables. As such, our team decided to also test Google's implementation of SLURM in the cloud using the Google Cloud HPC Toolkit. In addition, our team explored the use of Google Batch for an alternative to the Google Life Sciences API. Google deprecated the Life Sciences API and will no longer make it available after 2025 to instead only support Google Batch. At the time of writing, Google Batch was under evaluation for FedRAMP compliance. For this portion of benchmarking, our team used a sandbox Google Cloud Project without the normal NOAA connectivity or NOAA security and regulatory protocols. Our team chose this route to enable rapid benchmarking and prototyping but emphasizes that

bringing these console-native solutions to production within the NOAA environment would require additional custom configurations.

SLURM on Google Cloud

Our team deployed a SLURM cluster to Google Cloud Platform using the hpc-toolkit with the simplest configuration. More complex examples add additional machine types as partitions and additional storage configurations that were not needed for this benchmarking effort. Our team deployed our configuration using Terraform following the hpc-toolkit recommendations. This deployed a login and control node, as well as three partitions:

- A debug partition that includes up to four nodes with n2-standard-2 machine types (2 CPUs and 8 GB RAM).
- A compute partition which includes up to 20 c2-standard-30 (30 CPUs, 120 GB RAM) or c2-standard-60 (60 CPUs, 240 GB RAM) machines.
- An h3 partition that uses AMD rather than Intel machines.

Our team submitted all jobs to the compute partition. Our SLURM job scripts are given in the GitHub repository here. Our configuration created a Cloud Filestore, which is a high performance file system that is mounted directly to the cluster.

Our team also mounted a Google Storage bucket using the deployment scripts allowing jobs to read and write directly to Cloud Storage without having to store files long term on the Filestore. This helps to reduce costs, but it does lower I/O performance. Depending on the workflow, our team either submitted our jobs to a single compute node (Trinity) or attempted to spin up several nodes to accommodate all jobs (WGS Variant Calling).

Several configuration changes would be needed to deploy to a NOAA production environment. For example, the cluster would need to use a custom service account rather than the default compute engine service account. The cluster would also need to be deployed into an existing Virtual Private Cloud (VPC: a secure isolated network to house resources within a public cloud account) rather than create a new VPC for each SLURM deployment. Both of

these changes can be accommodated by more advanced terraform stacks such as Google's Enterprise SLURM environment.

Snakemake version 8.0 and above adds many new features to Snakemake's overall functionality, including the use of executor plugins which increase portability by allowing users to execute the same workflow across cloud and HPCC environments such as SLURM clusters without having to define parameters through different profiles. Snakemake workflows written for version 7 and below used profiles rather than these plugins to define execution and environment parameters. Plugins can be installed via pip and allow for users to mix and match compute and storage configurations, thereby increasing the portability of workflows. Our team found that Snakemake workflows written for version 7 (including those with profiles) did not always execute well with plugins and may require some refactoring to run well with Snakemake v8+.

Google Batch

Our team also tested Google Batch as a Snakemake executor using the Google Batch plugin in Snakemake v8+. Although Google Batch is not yet FedRAMP compliant, our team wanted to explore this service for three reasons. First, Snakemake v.8+ no longer supports the Life Sciences API, and benchmarking with version 7 did not allow us to use any of the other new Snakemake features including executor and storage plugins and containers. Second, the Life Sciences API will no longer be available after 2025, and Google will only support Batch executions. Finally, our team encountered many dependency conflicts with the Life Sciences API, as well as mount path issues outlined above. As such, our team hoped that Google Batch would serve as a more seamless executor because it allows for finer-grained parameter control, including a mount path.

Workflow Benchmarking Results

Variant Calling from Low Coverage Whole Genome Sequencing (lcWGS)

Background

This whole genome variant calling pipeline is a great example of many Snakemake best practices: it is modular, it uses wrappers when available, and it uses profiles to make the pipeline more portable between compute environments. It has been run and benchmarked on several on-premises SLURM environments as well as on Microsoft Azure's implementation of SLURM (AzHOP) using Snakemake v.7.

Our team initially benchmarked the variant calling pipeline on VWB using the Google Life Sciences API, but our team encountered two sources of error. First, the pipeline failed several times due to version conflicts between the launch container and conda environments. Our team was able to make necessary modifications to executable versions, but the pipeline failed due to Bash commands within Snakemake rules such as mkdir. These Bash commands execute as expected in a local or SLURM environment, but the pipeline crashed when run via Life Sciences API.

Because our team was unable to run the pipeline using the Life Sciences API (after 20+ hours of troubleshooting by two cloud-trained bioinformaticians), we ended up benchmarking the pipeline using a VWB Jupyter Environment. We ran Snakemake v.7.12 within a Jupyter notebook, and read and wrote data from the mounted Google Storage bucket using an n2-standard-30 machine. We found that the workflow did not scale well on a single virtual machine, and thus we tested two methods on the Google Cloud Console.

First, our team explored using our SLURM cluster. Following established protocols for this workflow, we ran the workflow end to end using Snakemake v.7.32.4 with the Sedna profile distributed with the GitHub repository. To make this work, we had to copy the data to the local home directory, requiring a large capacity high performance drive to be mounted which increases the overall cost of analysis. We also tested Snakemake v.8.14.0 using the Slurm executor without the Sedna profile, as well as the Google Cloud Storage plugin. The advantage with this method is that the files can be staged in a cloud storage bucket, saving money on expensive locally mounted disk space. Although this method will likely work well in the future, Snakemake took several hours to stage the data from Cloud Storage to the compute nodes. As

such, we only present results here for the Snakemake v.7 + Sedna profile method.

In a separate study, Eric Anderson benchmarked this workflow using Snakemake v.7 on Microsoft Azure using az-hop with a similar configuration of 20 compute nodes with a similar machine type using Intel Ice Lake processors with 16 CPUs and 64GB of RAM. Results are showing in the table below. He used slightly different samples of the same species thereby preventing perfect apples to apples comparison, but we include his results as a general reference.

Finally, our team explored using Google Batch for benchmarking of this workflow using the Batch executor plugin. Using the test data, each rule should take less than a second to execute on a laptop, but on Google Batch the same rules ran for over six minutes, then crashed. We were unable to locate any error logs, and we had to modify the source code to get it to run (indicating persistent bugs). Due to the support for containers and the absence of need for expensive persistent storage we feel Google Batch may be a great solution for NOAA Fisheries in the future, but that Snakemake development is not yet far enough along for this feature to work well for diverse workflows.

Results

Variant Calling					
Platform	Verily Workbench	Verily Workbench	Google Cloud	Azure	Sedna
Dataset	Test Data	Low Coverage WGS Rockfish	Low Coverage WGS Rockfish	Low Coverage WGS Rockfish	Low Coverage WGS Rockfish
Number Samples	18	5	43	43	43
Data Format	GZIP FASTQ	GZIP FASTQ	GZIP FASTQ	GZIP FASTQ	GZIP FASTQ
Mean Size	0.2 MB	1.9 GB	1.9 GB	1.9 GB	1.9 GB

Environment	VWB Jupyter Environment	VWB Jupyter Environment	SLURM, hpc-toolkit	SLURM, AZHop	SLURM
Method	Snakemake	Snakemake	Snakemake	Snakemake	Snakemake
Machine Used, CPU, RAM, Processors	n2-standard-32, 32 CPU, 128 GB, Intel Ice Lake/ \Cascade Lake	n2-standard-32, 32 CPU, 128 GB, Intel Ice Lake/ Cascade Lake	20 nodes of c2-standard-30, 30 CPUs, 120 GB, Intel Cascade Lake	20 nodes of Standard_D16ds_v5, 16 CPUs, 64 GB, Intel Ice Lake	20 CPUs, 192 GB
Number of Snakemake Processes	269	881	2687	2658	2658
Total Time (hrs)	2.16	137	6.92	5.1	6.4
Time/Sample (min)	7.2	1620	9.6	7.1	8.9
Compute Total $$	$4.6	$292.0	$12.55	$5.7	-
Compute $$/Sample	$0.000008	$0.22	$0.29	$0.13	-
Storage $$/Month	$0.046	$0.368	$1.9	$1.7	-

Lessons Learned

This workflow requires a very high number of Snakemake processes running on one or a few CPUs. We found that this workflow was incompatible with Life Sciences API, and scaled poorly on a single Virtual Machine. It also requires many read/write executions with the mounted storage bucket which is much slower than reading and writing from local files. AS such, VWB performed poorly for this workflow. This workflow benefits most from a parallel execution environment such as SLURM or Batch because it requires low CPUs/RAM for hundreds of processes. We found that the best solution at present was using existing Slurm profiles with Snakemake v.7.

DNA Metabarcoding

Background

Tourmaline is a well-established Snakemake workflow developed by the AOML 'Omics program. The workflow has detailed documentation that outlines the setup of a custom conda environment, as well as the installation of several reference databases. The workflow also comes with a pre-built container that includes the conda environments pre-installed. Tourmaline also adds additional functionality to amplicon workflows by running Qiime2 instead of only DADA2. NOAA Fisheries PI Kim Ledger initially benchmarked a similar workflow called dadasnake, which wraps Snakemake to run QC and DADA2. In our preliminary testing, dadasnake only worked with Snakemake v.7 rather than v.8, and dadasnake was not well designed to run using remote batch execution.

Our team attempted to benchmark Tourmaline using several strategies. First, we attempted using the Life Sciences API on VWB but encountered issues with the mount paths of the input files. Tourmaline expects locally mounted data and is not configured well for remote batch execution. Due to this limitation, our team decided to benchmark Tourmaline using our SLURM cluster and a single Compute Engine virtual machine (CE VM). Our team encountered errors when creating the conda environment on both machines (SLURM login node and CE VM), and instead chose to use containers for the software deployment method. Snakemake now supports executing either the whole workflow from within a single container, or specifying a container for each rule and executing using Apptainer/Singularity. Tourmaline already has a publicly available container, but, because the container already has conda environments installed, we found that the best way to run the workflow was with docker interactively: *sudo docker run -v $HOME:/data -it aomlomics/tourmaline /bin/bash -c "sh run_tourmaline"* where the run_tourmaline script contained the necessary snakemake commands to run the workflow with --jobs 30 (the workflow only uses up to 8 CPUs per rule) and conda as software deployment method. Our virtual machine was a n2-standard-16 (16 CPUs, 64 GB RAM) which uses Intel Cascade Lake processors and represents the newest general purpose processors. Although some speed differences may have been observed by using a

compute-optimized machine like the c2 machine family, using the N2 family compared with the older E2 provides substantial performance improvements due to the newer generation of Intel processors.

Results

Tourmaline			dadasnake + BLAST
Platform	Google Cloud	Sedna	Sedna
Dataset	GoA MiFish	GoA MiFish	GoA MiFish
Number Samples	1014	1014	1014
Data Format	GZIP FASTQ	GZIP FASTQ	GZIP FASTQ
Mean Size	2.8 MB	2.8 MB	2.8 MB
Environment	Compute Engine VM	SLURM	SLURM
Method	Snakemake	Snakemake	Snakemake + Bash
Machine Used, CPU, RAM, Processors	n2-standard-16, 16 CPUs, 64 GB, Intel Ice Lake/ Cascade Lake	20 CPUs, 92 GB, Intel(R) Xeon(R) Silver	20 CPUs, 30 GB
Number of Snakemake Processes	44	44	-
Total Time (hrs)	2.5	2.32	1.25
Time/Sample (min)	0.148	0.137	0.075
Compute Total $$	$2.7	-	-
Compute $$/Sample	$0.0027	-	-
Storage $$/Month	$0.07	-	-

Lessons Learned

Tourmaline was written with the intent of running the workflow from a local directory and is not yet configured for running with remote batch processing. Although the provided container ameliorates many of the challenges associated with the creation of custom conda environments, the container still requires the use of conda to run within the container, rather than installing all dependencies in the base environment. Our team recommends instead to point to rule-specific containers or conda environments, and to allow Snakemake to create the conda environments rather than requiring the user to set these up in advance. In addition, making the workflow more modular may help with cloud execution, and allow for the addition of additional modules as new software is developed for these types of analyses.

Metagenomics

Background

Amethyst is a new Snakemake workflow developed for this project by the AOML 'Omics program. The workflow requires the creation of several conda environments as well as very custom database setup procedures.

Our team initially benchmarked Amethyst using the Life Sciences API in VWB, but the dependence on creating multiple conda environments made this implementation difficult due to software version conflicts. After many hours of troubleshooting, our team ended up benchmarking Amethyst using a single n2-standard-30 virtual machine within a VWB environment. This machine has 32 CPUs and 128 GB of RAM with Intel Ice Lake or Cascade Lake processors. Due to slow runtimes (6+ hours for MaxBin step) and high memory requirements, our team limited our benchmarking of Amethyst to five samples. This pipeline was still in development during our benchmarking, so the level of effort was high to get it running end to end.

Results

Amethyst				
Platform	Verily Workbench	Verily Workbench	Verily Workbench	Sedna
Dataset	100k reads metagenomes	Seawater metagenomes	Seawater metagenomes	Seawater metagenomes
Number Samples	30	5	10	5
Data Format	GZIP FASTQ	GZIP FASTQ	GZIP FASTQ	GZIP FASTQ
Mean Size	70 MB	3.2 GB	3.2 GB	3.2 GB
Environment	VWB Jupyter Environment	VWB Jupyter Environment	VWB Jupyter Environment	SLURM
Method	Snakemake	Snakemake	Snakemake	Snakemake
Machine Used, CPU, RAM, Processors	n2-standard-32 , 32 CPU, 128 GB, Intel Ice Lake/Cascade Lake	n2-standard-32 , 32 CPU, 128 GB, Intel Ice Lake/Cascade Lake	n2-standard-32 , 32 CPU, 128 GB, Intel Ice Lake/Cascade Lake	20 CPUs, 192 GB
Number of Snakemake Processes	274	49	93	44
Total Time (hrs)	4.7	27.2	88	6.4
Time/Sample (min)	9.4	326	528	76.8
Compute Total $$	$9.9	50.9	164.6	-
Compute $$/Sample	$0.029	10.18	16.5	-
Storage $$/Month	$0.046	$0.368	$0.736	-

Lessons Learned

This workflow was being developed while our team was conducting benchmarking, and many modifications were needed to get it running properly. Namely, our team found that the MaxBin2 step doubled the total runtime, and megahit also required long runtimes due to kmer optimization. Making the workflow more modular, defining rule-specific containers and conda environments (created by Snakemake rather than the user), and optimizing several steps may all enable better performance in the cloud. The workflow ran slowly on Sedna, although it ran much faster than our VWB implementation.

Transcriptome Assembly

Background

Our Trinity workflow ran from a single Trinity command which included quality trimming using trimmomatic and subsequent transcriptome assembly. Our team tested Trinity only using our SLURM environment on Google Cloud submitting to a single compute node. Our team benchmarked Trinity v. 2.15.1, with Snakemake v.8.12.0. Trinity performs best when given all files to work with at once rather than running one by one. Although runtimes are longer as you add samples, Trinity has several steps that are not longer with more samples, meaning our team could not get a per-sample estimate simply by dividing runtime by number of samples. Although our team requested a single node with 30 CPUs, our job only used 20 CPUs and 200 GB RAM to maintain parity with Sedna resources.

Results

Trinity		
Platform	Google Cloud	Sedna
Dataset	Steelhead RNA	Steelhead RNA
Number Samples	20	20

Data Format	GZIP FASTQ	GZIP FASTQ
Mean Size	6.2 GB	6.2 GB
Environment	SLURM	SLURM
Method	Bash	Bash
Machine Used, CPU, RAM, Processors	One node of c2-standard-30, 30 CPUs, 120 GB, Intel Cascade Lake	Himem node, 24 CPUs, 1.5TB (200GB used), Intel(R) Xeon(R) Silver
Number of Snakemake Processes	-	-
Total Time (hrs)	61.6	56.1
Time/Sample (min)	NA	NA
Compute Total $$	$111.7	-
Compute $$/Sample	-	-
Storage $$/Month	$2.8	-

Lessons Learned

This workflow requires a single compute environment with high CPU and memory requirements. It worked very well with our SLURM configuration but required long runtimes but was comparable to Sedna.

Cloud Cost Management

Our tables above show only the costs of compute and cloud storage, but several other factors should be considered when estimating cloud costs. In the case of Verily Workbench, licensing costs would need to be added to any cost estimates. Beyond the license, the Google Project hosting the Verily Workbench project is primarily only charged for compute and storage, with

relatively minor charges for Cloud DNS, Artifact Registry and BigQuery (if used). One downside to the current version of Verily Workbench is that auto shutdown of virtual machines is not yet enabled, whereas in Google Vertex AI auto shutdown is enabled by default. This means that the user needs to remember to stop compute environments in VWB to avoid accruing unnecessary costs.

A Google Cloud console implementation depends on several additional services. First, Cloud Filestore is used for local storage to provide high performance read/write operations compared with a storage bucket. However, this approach requires copying data locally for analysis, rather than leaving raw files in a bucket. Filestores are also expensive; in our case, 4 TB of storage cost $481.00 per month, and real data storage will be orders of magnitude greater. A better, though slightly slower, approach is to mount a bucket to the cluster via the Terraform code and read and write to it. If speed is important, however, one could mount an alternative, but cheaper performant file system such as JuiceFs.

In addition to Compute Engine, notebook instances, Cloud Storage, Artifact Registry, and Cloud Filestore, our Google Cloud Project had monthly charges for Cloud Monitoring ($30.87) and networking ($20.95). For these charges, the best approach to reducing expenses is ensuring that machine types are no larger than needed, shutting down instances not being used, and exploring the use of spot instances. One newer option for 'spot surfing' is the tool offered by Memory Machine Cloud, which allows jobs to jump from spot to spot, potentially saving up to 50% on compute costs.

Finally, costs for cloud engineers and bioinformatics support specialists were covered in-kind or from our NOAA HPCC incubator funding and other partners. NOAA scientists engaged in bioinformatics analyses will need to rely on centralized support for bioinformatics computing in the cloud in an ongoing manner, as the needs and software supporting analyses are dynamic across the Agency.

Findings Summary

Verily Workbench: Verily Workbench provides quick and easy access to compute environments and tools without the user needing to provision cloud resources. However, it is better suited to users of Nextflow and WDL than to Snakemake or BASH. Many of the legacy and ad hoc analyses conducted at NOAA Fisheries are not written in workflow languages.

Execution Environments:

- Local Jupyter notebook or local Virtual Machine execution scales poorly for production scale Snakemake workflows.
- Google Life Sciences API and Google Batch required a high level of effort to get running and were difficult to get workflows to run end to end.
- SLURM on GCP presents the most flexibility and least level of effort (after initial setup), but carries additional costs due to mounted storage.

Workflows:

- For migration to the cloud to be successful, workflows need to align with best practices, whether Snakemake, Nextflow, or WDL. However, conversion of all existing workflows to Snakemake would require a high level of effort and ongoing support as workflow languages adapt. For example the recent changes from Snakemake v.7 to v.8 are not backwards compatible with some of the workflows we benchmarked here.
- Not all staff are trained to use workflow languages, and ad hoc and legacy workflows will continue to use BASH or similar languages. NOAA Fisheries therefore requires a flexible environment to also execute these types of analyses.
- Snakemake's adoption of containers represents an important advancement to reduce software version conflicts, but containers are not often practical for bespoke analyses that are used for many diverse applications in NOAA (e.g. evaluation of inbreeding depression, evaluating epigenetic signatures under environmental stressors, running simple tools for QA/QC, and population genetic statistics). In these cases, researchers would need to build custom containers that

may require periodic updates, and many researchers may not have the expertise to do so.

- Not all bioinformatic and genetic analyses executed by NOAA scientists require workflows or heavy compute environments. For interactive tasks such as plotting results, a simple Jupyter environment like that of VWB or Vertex AI is sufficient.

Environment Management: Access to the cloud for NOAA would need to be thoughtfully managed by cloud engineers who understand bioinformatic use cases and requirements. These engineers could support bioinformatics work across offices.

Recommendations

Objective 3: Make Recommendations for NOAA Cloud Computing

This project sought to test the feasibility, cost, and efficiency of conducting bioinformatics in the cloud. Based on our benchmarking across diverse platforms and methods, we make the following recommendations:

Cloud Platform: Our team recommends using a native Google Cloud environment over VWB or other PaaS offerings. Despite the initial environment setup such as provisioning virtual machines or a SLURM environment, NOAA conducts a wide variety of analyses, and the flexibility of the Google Cloud Console likely outweighs the benefits of managed PaaS offerings. Furthermore, the current dependence on Snakemake as a workflow language gives the Google Cloud Console further advantage.

Execution Environment: Our team recommends using SLURM HPCC over stand alone virtual machines or Batch computing. The mounted storage required substantially increases the cost of a production-scale SLURM cluster, but these costs can be managed using a variety of strategies.

Cost Management: Our team recommends storing data in cloud storage using tiered storage. NOAA could conduct a subsequent pilot to explore the feasibility of compression tools such as Petagene, which can achieve up to 90% compression of FASTQ and BAM files, or explore using JuiceFS as an affordable fast file system.

Future Directions: Our team recommends revisiting Google Batch + Google Cloud Storage using the Snakemake v.8 executor plugins. This is a very new method that did not work well yet, but is advantageous for Snakemake workflows because no initial set up or no additional storage are needed compared with HPCC.

Acknowledgements

This work was primarily funded by the NOAA HPCC Incubator project entitled "Assessing Terra Running on Google Cloud as a Potential Platform for 'Omics Data Storage and Analysis" which was funded in FY23.

The project was proposed by a number of principal investigators, including:

- Krista Nichols: *NOAA Fisheries, NWFSC*
- Nancy Majower, Sean Blakeney, Greg Bledsoe, and Ed Rodgers: *NOAA Fisheries, OCIO*

Partners that participated in the conception of the proposal include:

- Abenaa Addei, Eric Pennaz, Ross Thompson: *Google Cloud Platform*
- Clare Bernard: *Broad Institute*
- Eric Archer and Eric Anderson: *NOAA Fisheries, SWFSC*
- Giles Goetz and Marty Kardos: *NOAA Fisheries, NWFSC*
- Luke Thompson & Rachael Storo: *Northern Gulf Institute, Mississippi State University*
- Sean Horgan: *Verily Life Sciences*
- Wes Larson: *NOAA Fisheries, AFSC*

Once the project was underway, the project was managed by:

- Krista Nichols: *NOAA Fisheries, NWFSC*
- Nancy Majower, Sean Blakeney, Ed Rodgers: *NOAA Fisheries, OCIO*

And the the work would not have been possible without the support of:

- John Bates and Tyler Trueg: *Verily Workbench*

Workflow development, on premise, and cloud benchmarking was supported by:

- Eric Anderson, Giles Goetz, Kim Ledger: *NOAA Fisheries*
- Luke Thompson and Rachael Storo: *Mississippi State University, North Gulf Institute*
- Insana Collins, Kyle O'Connell, Ramiya Sivakumar, Monica McEwen: *Starlo Innovation*

Appendices

Appendix A. Corresponding GitHub Repository

FY24 HPCC Incubator Testing for Bioinformatics

https://github.com/noaa-nwfsc/FY24-HPCC-Incubator-testing-for-bioinformatics

This GitHub repository has all needed config files for bioinformatic workflows, the SLURM Terraform scripts and instructions, and rclone instructions for moving data from Google Drive to Google Cloud.

Appendix B. Workflow Diagrams

Verily Workbench (WB) Implementation Flowchart

[Github View]

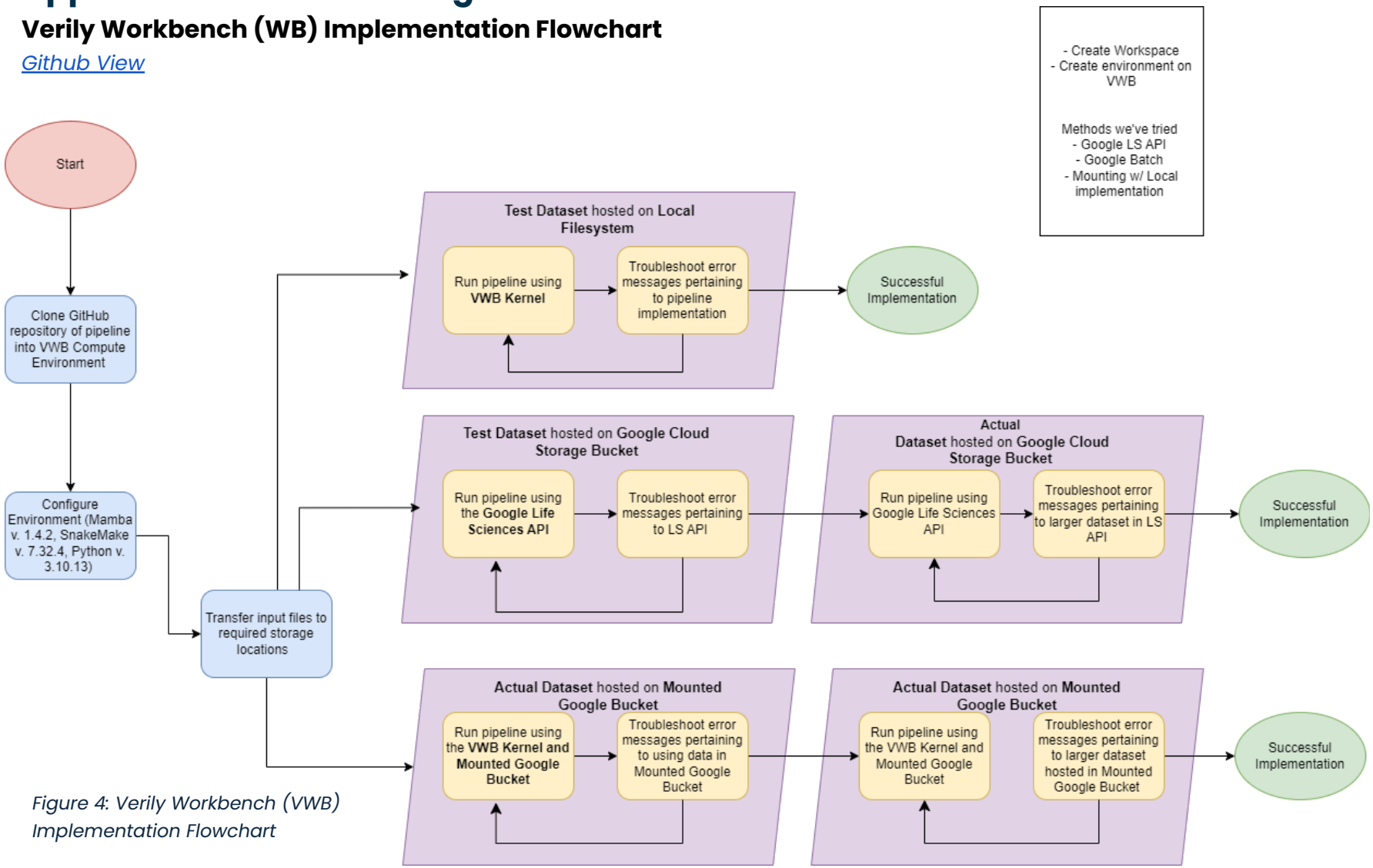

Figure 4: Verily Workbench (VWB) Implementation Flowchart

Metagenomics Workflow Diagram

[Github View]

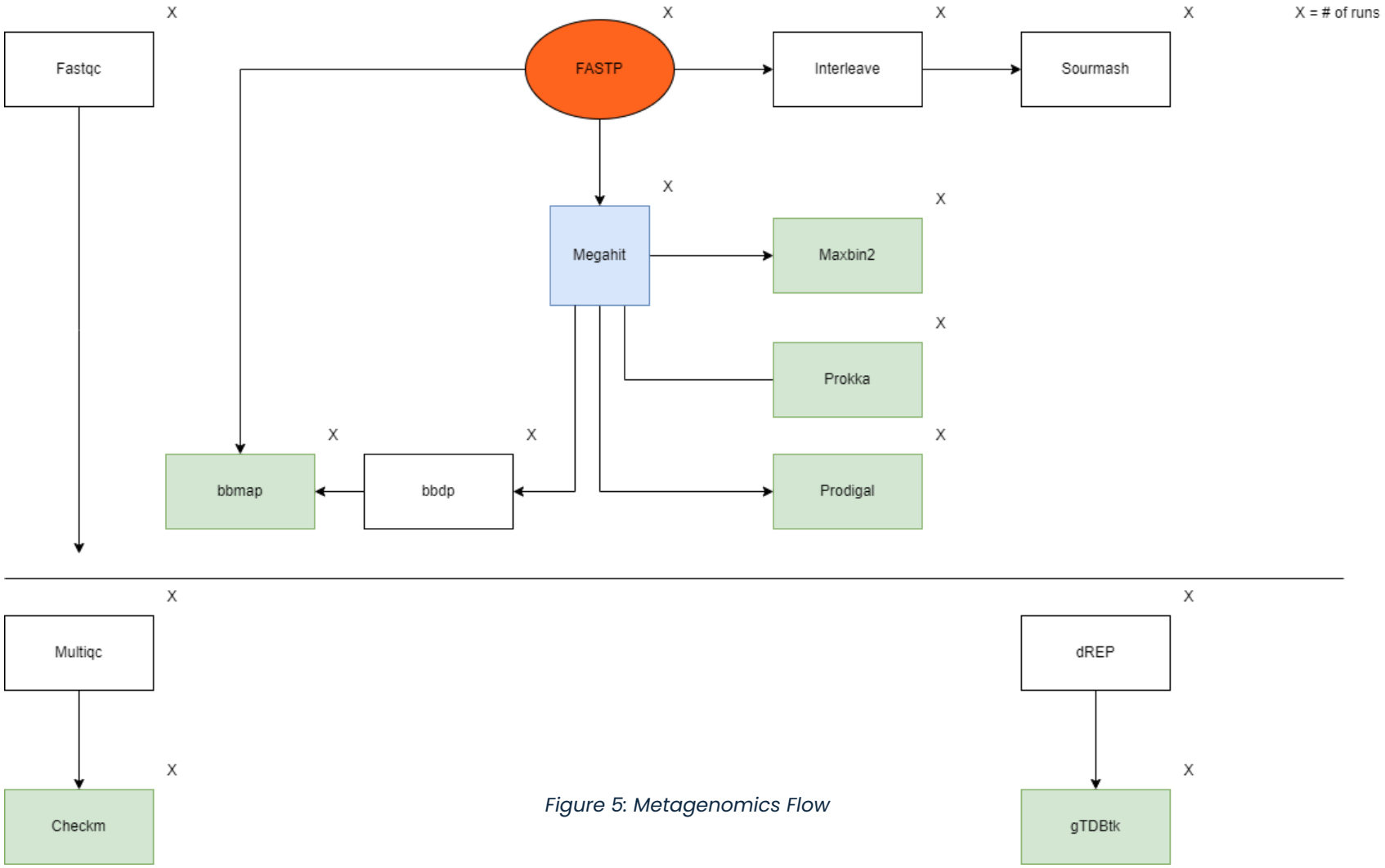

Figure 5: Metagenomics Flow

WGS Variant Calling Workflow

Github View

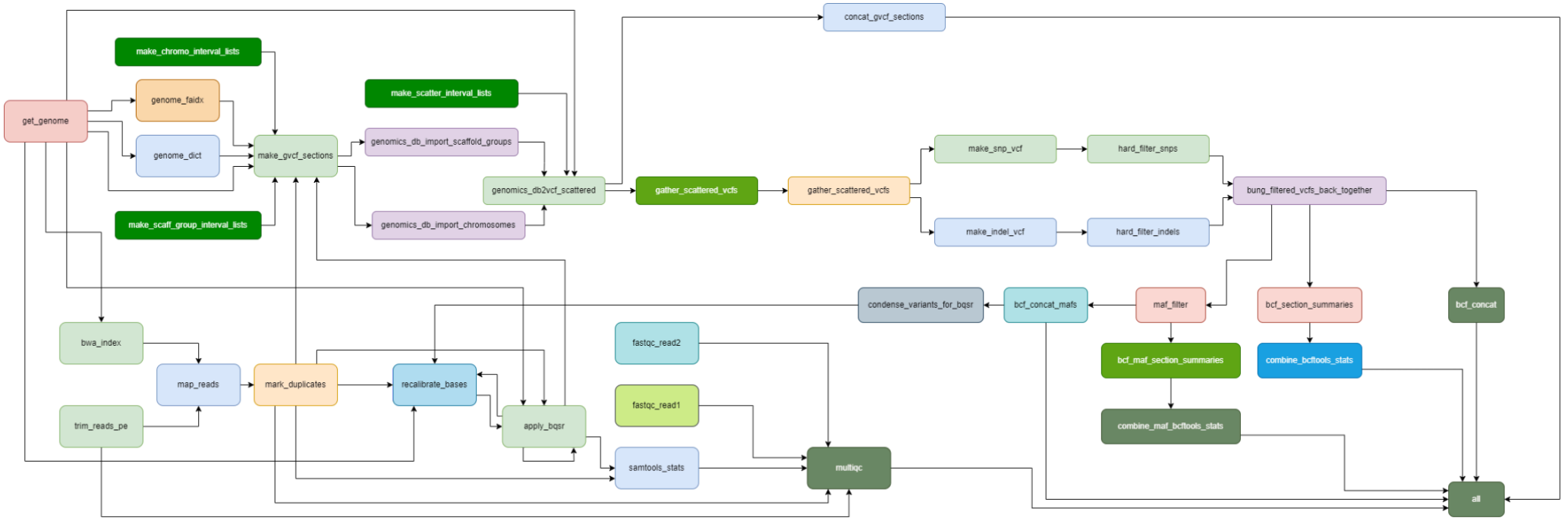

Figure 6: WGS Variant Calling Workflow