

Celeste – Overview & Guide

Authors: Noora Siddiqui

BACKGROUND AND PURPOSE

The HGSC-CL has developed a cloud infrastructure, *Celeste*, packaged with a variant calling pipeline to analyze NGS data utilizing the Illumina's DRAGEN technology and Amazon Web Services (AWS). This cloud infrastructure manages the secondary analysis of whole genome sequencing data in a series of Extract Transform Load (ETL) steps and is capable of dynamically scaling to process tens of thousands of samples for population-scale sequencing projects. Resources in AWS were designed as highly available, decoupled, event-driven systems in accordance with microservices architectural principals. Command-line parameters included in the guide are suitable for clinical projects.

This overview and guide provides an extensive view of *Celeste* with a focus on AWS services, describing how to troubleshoot errors, and consolidating external resources for users to explore. It also serves to document specific parameter sets and enumerate existing resources for development purposes.

Celeste

1.1 Table of Contents

| | |
|--|-----------|
| BACKGROUND AND PURPOSE..... | 1 |
| PROCEDURE OVERVIEW | 4 |
| OVERVIEW OF NGS SECONDARY ANALYSIS STEPS AND I/O | 4 |
| <i>Figure 1. Overview of NGS Secondary Analysis Steps and I/O in Celeste</i> | <i>5</i> |
| AWS CLOUDFORMATION..... | 6 |
| <i>Templates/.....</i> | <i>6</i> |
| <i>Exercise: Understanding Nesting Relationships</i> | <i>7</i> |
| <i>ci/.....</i> | <i>8</i> |
| <i>app/.....</i> | <i>9</i> |
| <i>doc/, submodules/, and taskcat dir/files.....</i> | <i>9</i> |
| TASKCAT AND CI/CD PIPELINE FOR INFRASTRUCTURE DEPLOYMENTS | 10 |
| <i>MUST Read the Following:</i> | <i>10</i> |
| <i>CI/CD CloudFormation Templates with AWS CodePipeline.....</i> | <i>10</i> |
| DEPLOY CELESTE | 11 |
| UPDATING THE DRAGEN AMI WITHIN TEMPLATES | 12 |
| AWS SIMPLE NOTIFICATION SERVICE (SNS) | 14 |
| WHY SNS? | 14 |
| OVERVIEW OF THE JOB TRACKING SYSTEM | 16 |
| HOW TO SET-UP AN SNS NOTIFICATION..... | 16 |
| <i>Case-Study: The Process of Creating A JobFail SNS Notifications</i> | <i>16</i> |
| AWS BATCH | 20 |
| COMPONENTS | 20 |
| <i>Job.....</i> | <i>20</i> |
| <i>Job Queue</i> | <i>20</i> |
| <i>Compute Environment.....</i> | <i>20</i> |
| <i>Job Definition</i> | <i>21</i> |
| AWS BATCH AND VPC ARCHITECTURE | 23 |
| <i>A Suggested Multi-Environemtn VPC IPv4 CIDR Scheme</i> | <i>24</i> |
| <i>A Note on Availability Zones.....</i> | <i>24</i> |
| SPOT INSTANCES | 25 |
| <i>Case-Study: Automating Job Retries After Spot Interruption</i> | <i>26</i> |
| GOOD PRACTICES FOR CELESTE BATCH JOB SUBMISSION | 27 |
| TROUBLESHOOTING AWS BATCH JOBS..... | 27 |
| <i>Suggested Reading</i> | <i>27</i> |
| CONTAINERS WITHIN THE CELESTE ARCHITECTURE..... | 28 |
| <i>Development Steps.....</i> | <i>28</i> |
| <i>Example</i> | <i>29</i> |
| AWS LAMBDA | 31 |
| RECOMMENDED READING AND TUTORIALS: | 31 |
| AWS SIMPLE STORAGE SERVICE (S3) | 32 |

The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

| | |
|---|-----------|
| INTRODUCTION TO S3 | 32 |
| <i>Beginner S3 Commands</i> | 32 |
| <i>Checksums</i> | 32 |
| UNDERSTANDING SELECT S3 REFERENCES | 33 |
| <i>Buckets</i> | 33 |
| <i>Prefixes within s3://<bucketname> (referred to as Genomics S3 Bucket)</i> | 35 |
| S3 LIFECYCLE MANAGEMENT | 36 |
| ARCHIVAL | 39 |
| <i>Understanding Automated Tag-based Archival in the Pipeline</i> | 39 |
| <i>Case-Study: Unarchiving data via AWS S3 Batch Operations</i> | 40 |
| <i>Intermediate S3 Commands</i> | 42 |
| S3 EVENT NOTIFICATIONS | 45 |
| <i>Creating an S3 Event Notification via Console</i> | 46 |
| MAKING A NEW S3 BUCKET | 47 |
| <i>Case-Study: A bucket with lifecycle configuration, encryption, and security specifications</i> | 47 |
| AOU PARAMETER SET – HS37D5 | 49 |
| WORKFLOW (TOOL AND VERSION): DRAGEN SPECIFICATIONS | 49 |
| <i>Deployment/Environment</i> | 49 |
| <i>hs37d5 Merge Event: DRAGEN Map & Align, and Variant Call Command Line</i> | 49 |
| <i>hs37d5 Single Sequencing Event: DRAGEN Map & Align Command Line</i> | 50 |
| WORKFLOW (TOOL AND VERSION): ALIGNSTATS SPECIFICATIONS | 51 |
| <i>If BAM:</i> | 51 |
| <i>If CRAM:</i> | 51 |
| WORKFLOW (TOOL AND VERSION): VERIFYBAMID SPECIFICATIONS | 52 |
| <i>If BAM:</i> | 52 |
| <i>If CRAM:</i> | 52 |
| AOU PARAMETER SET – GRCH38 | 53 |
| GRCH38 DRAGEN SPECIFICATIONS | 53 |
| <i>MERGE EVENT: DRAGEN Map & Align, and Variant Call Command Line</i> | 53 |
| GRCH38: ALIGNSTATS SPECIFICATIONS | 54 |
| <i>If BAM:</i> | 54 |
| <i>If CRAM:</i> | 54 |
| GRCH38: INTERSECT SPECIFICATIONS | 55 |
| <i>BAM/CRAM:</i> | 55 |
| GRCH38: PREPROCESSING SPECIFICATIONS | 55 |
| <i>BAM/CRAM:</i> | 55 |
| GRCH38: PREPROCESSING SPECIFICATIONS | 55 |
| <i>BAM/CRAM:</i> | 55 |
| GRCH38: LIFTOVER SPECIFICATIONS | 56 |
| <i>BAM/CRAM:</i> | 56 |
| GRCH38: STARGAZER SPECIFICATIONS | 56 |
| <i>BAM/CRAM:</i> | 56 |

Celeste

PROCEDURE OVERVIEW

Overview of NGS Secondary Analysis Steps and I/O

As outlined in the diagram below (Figure 1), the pipeline begins in the “HGSC Data Center” (blue-outlined box) by converting the primary image data (.bcl files) generated from the Illumina sequencing instruments to fastq files. The pipeline communicates with the LIMS to obtain information that specifies the pipeline management tool (HgV) and analysis protocols that directly feed into the AWS cloud environment (grey-outlined box).

The AWS cloud environment automates and orchestrates the NGS secondary analysis. This includes submitting the needed jobs for quality control reports and managing the storage class and archival logic for all resulting outfiles.

When fastqs for a single Sequencing Event (SE) have completely generated at the HGSC Data Center, they are then transferred to AWS Simple Storage Service (S3). HgV initial analysis protocols directly submit DRAGEN alignment jobs to AWS Batch (fastqs to .bam/.cram). During the mapping step, reads are aligned to the human reference, hs37d5, using the DRAGEN mapper/aligner with marked duplicate reads. DRAGEN coverage reports are generated across autosomal, ACMG, and PGx regions. Other notable outfiles include a DRAGEN contamination metric, ploidy report, time metrics, and WGS metrics. These DRAGEN outfiles land in S3 and publish a message to AWS Simple Notification Service (SNS). AWS Lambda functions subscribed to the SNS topic are responsible for the submission of quality control jobs (such as AlignStats and VerifyBamID). Another Lambda function tags S3 files for archival and/or data delivery.

When a “Merge Ready” Event (ME, yellow “EVENT” in diagram) is detected at the HGSC Data Center, separate HgV protocols submit DRAGEN **merge** jobs to AWS Batch. These **full-pipeline analysis** jobs merge data across multiple paired fastqs to produce a single .bam/.cram alignment file, .hard-filtered.vcf.gz, and .hard-filtered.gvcf.gz containing variant calls. The resulting DRAGEN output will again trigger a fan-out of serverless functions that submit quality control/report jobs and generate tags for file archival and/or delivery, dependent on filename.

Output files of both the SE and ME workflows are periodically pulled down to the local cluster from S3.

The metrics contained in the specified AlignStats, DRAGEN, VerifyBamID, etc. files are parsed by HgV and then pushed to LIMS. Stargazer, Liftover, and Intervar files are saved locally for later report generation.

Celeste

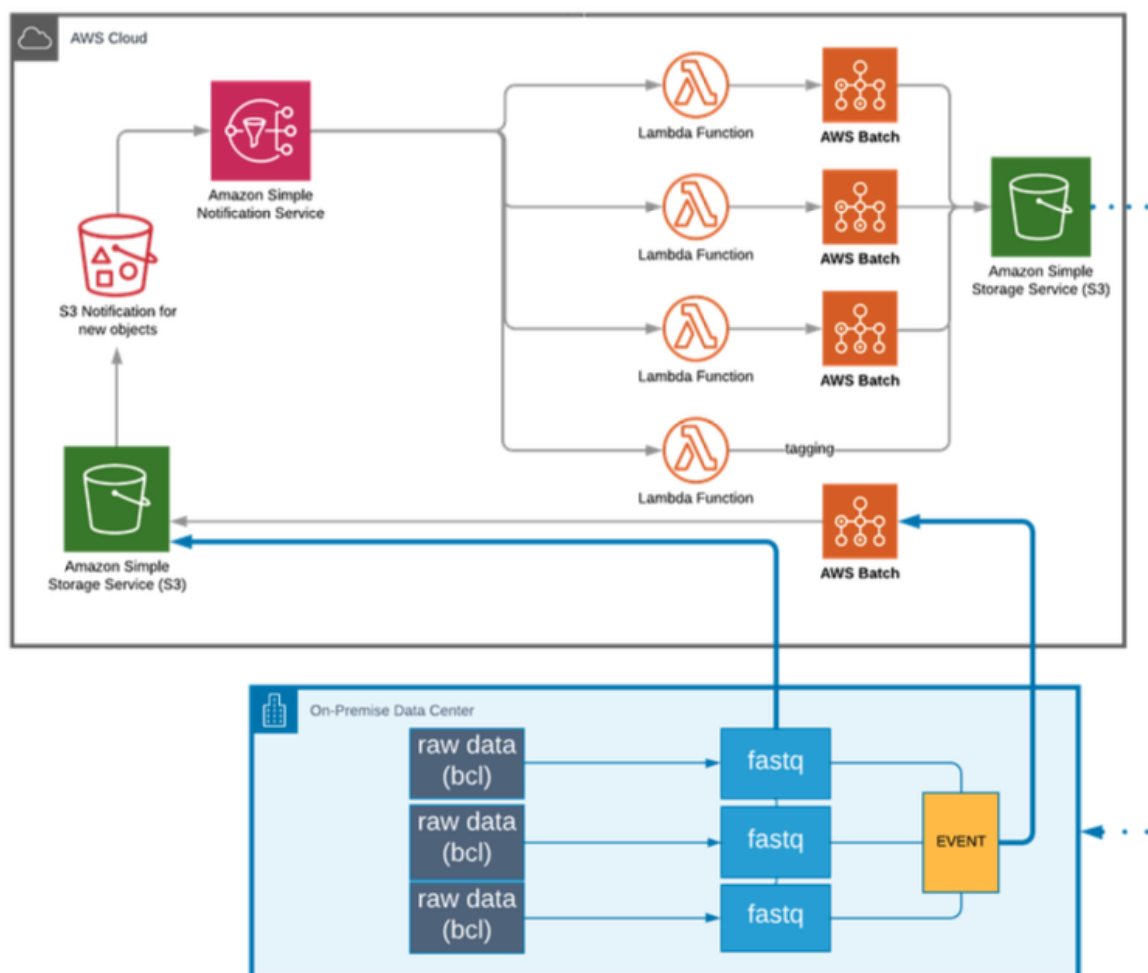


Figure 1. Overview of NGS Secondary Analysis Steps and I/O in Celeste

Celeste

AWS CLOUDFORMATION

AWS CloudFormation allows us to spend less time managing minute details of AWS resources and more time focusing on our applications running in the cloud. A CloudFormation template in .yaml or .json formats can provision all of your necessary cloud infrastructure and automate the deployment of these resources across all regions and accounts in a reproducible way. These simple templates allow us to effectively version our environment. This is useful to both track changes over time, and efficiently delete and re-create resources.

This is referred to as Infrastructure as Code (IaC).

The *Celeste* repository contains all of the CloudFormation templates and associated assets required to automate the deployment of *Celeste*.

Templates/

Take a look within templates. There are many different .yaml files that specify everything from buckets and lambdas to IAM roles and policies. Many of these templates nest within one another (See: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-nested-stacks.html>).

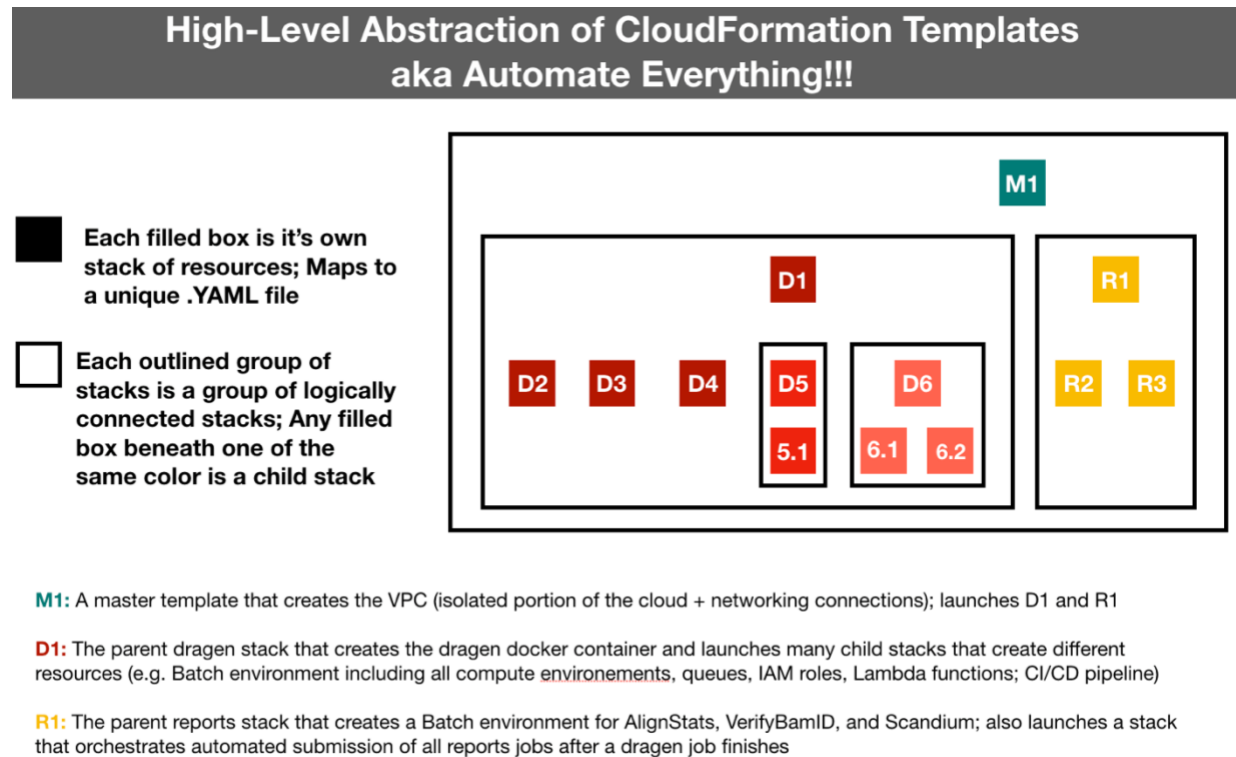
Any templates with “standalone” in the name, have no nesting relationships and can be launched independently.

```
[26] → ls templates/
batch-roles.yaml      container-build.yaml  job-tracker-resub-handler.standalone.yaml
celeste.yaml          copy.yaml             reports-batch-9.19.parent.yaml
clean-bucket-repository.yaml  docker-bucket-repository.yaml  s3-config.standalone.yaml
clean-bucket.yaml      dragen-3.4.12.yaml    sns-lambdas.yaml
clean-repository.yaml  dragen-batch.yaml
```

Celeste

Exercise: Understanding Nesting Relationships

Look through the *Celeste* templates, see if you can use the documentation above and your own exploration to fill template names into the colored boxes in the diagram below.



Celeste

ci/

The Continuous Integration directory contains a parameters file to track the parameters that are fed into the celeste.yaml template during the provisioning of *Celeste* environments.

- The “...Bucket” parameters need to point to existing buckets within your AWS account. The “GenomicsS3Bucket” will be the primary analysis bucket for genomics I/O
- The “KeyPairName” requires a valid, existing keypair from your AWS account
- The “OperationsDataBucket” should house any files referenced by AWS Batch jobs (bed files, reference files) and AWS Batch jobs running within Celeste will only have read access to this data
- The “QSS3BucketName” is the bucket that will contain the Celeste repo with cloudformation templates; this can be the same as the OperationsDataBucket and Cloudformation will have read access to templates in this space
- The “QSS3Prefix” is the prefix where the Celeste deployment/Celeste repo is housed within your account
- All parameters with keys ending in “...ImageRepo” are the recommended names for ECR repositories with docker containers for that software

Celeste

app/

Within the app subdirectory of the Celeste repository lies the docker container assets for DRAGEN, as well as all serverless (Lambda) functions associated with the cloud infrastructure (these are housed under “packages”).

```
[10] → find app -maxdepth 3
app
app/packages
app/packages/cleanRepo
app/packages/cleanRepo/cleanRepo
app/packages/cleanRepo/cleanRepoLambda.zip
app/packages/lambda
app/packages/lambda/VerifyBamID-38-BatchJobLauncher.zip
app/packages/lambda/LiftoverLauncher.zip
app/packages/lambda/LiftoverFunctions
app/packages/lambda/Alignstats-38-BatchJobLauncher.zip
app/packages/lambda/Alignstats-BatchJobLauncher.zip
app/packages/lambda/MpileupCassandraLauncher.zip
app/packages/lambda/PreprocessingLauncher.zip
app/packages/lambda/Cram2GvcfLauncher.zip
app/packages/lambda/trackBatchJobs.zip
app/packages/lambda/otherFunctions
app/packages/lambda/StargazerLauncher.zip
app/packages/lambda/IntervarLauncher.zip
app/packages/lambda/GvcfQcLauncher.zip
app/packages/lambda/CassandraLauncher.zip
app/packages/lambda/Scandium-BatchJobLauncher.zip
app/packages/lambda/Scandium-38-BatchJobLauncher.zip
app/packages/lambda/reportsFunctions
app/packages/lambda/IntersectLauncher.zip
app/packages/lambda/S3TagLambda.zip
app/packages/copyZips
app/packages/copyZips/copyZips
app/packages/copyZips/copyZipsLambda.zip
app/packages/dragen
app/packages/dragen/dragen
app/packages/dragen/dragen.zip
app/packages/updateConfig
app/packages/updateConfig/updateConfigLambda.zip
app/packages/updateConfig/updateConfig
```

doc/, submodules/, and taskcat dir/files

- doc/ contains the original documentation associated with the “Illumina DRAGEN on AWS” QuickStart. (Source code: <https://github.com/aws-quickstart/quickstart-illumina-dragen>)
- submodules/ contains an AWS VPC QuickStart used by the celeste.yaml in *templates/* to configure the environment VPC
- The taskcat dir/files are artifacts of utilizing the TaskCat module to test the deployment of infrastructure. The **.taskcat.yml** within the git repository is a file utilized by TaskCat version 0.9.17 to run a test (ephemeral) deployment

Celeste

TaskCat and CI/CD Pipeline for Infrastructure Deployments

MUST Read the Following:

The following links are all interconnected with one another, and some cases describe the same material. Explore each thoroughly:

- <https://aws.amazon.com/blogs/infrastructure-and-automation/a-deep-dive-into-testing-with-taskcat/>
- <https://docs.aws.amazon.com/quickstart/latest/cicd-taskcat/welcome.html> (read all deployment steps)
- <https://github.com/aws-quickstart/taskcat>
- <https://aws.amazon.com/quickstart/architecture/cicd-taskcat/>
- <https://stelligent.com/2019/11/27/run-aws-cloudformation-tests-from-codepipeline-using-taskcat/>

CI/CD CloudFormation Templates with AWS CodePipeline

Celeste is compatible for use with the AWS QuickStart “CI/CD Pipeline for AWS CloudFormation templates on AWS”, with the addition updates that involve utilizing TaskCat 0.9.17 or higher.

Celeste

Deploy Celeste

```
$ git clone <celeste-repo>
```

```
$ aws s3 sync <local celeste repo> s3://QSS3BucketName/QSS3Prefix/cloudformation/clinical-dragen-prod/
```

Utilize the s3 s3-config.standalone.yaml to provision a GenomicsS3Bucket if you don't have one already

Update ci/params.json (specifically the keys listed in the previous section regarding "ci/")

Create and push docker images (listed within params.json) to ECR

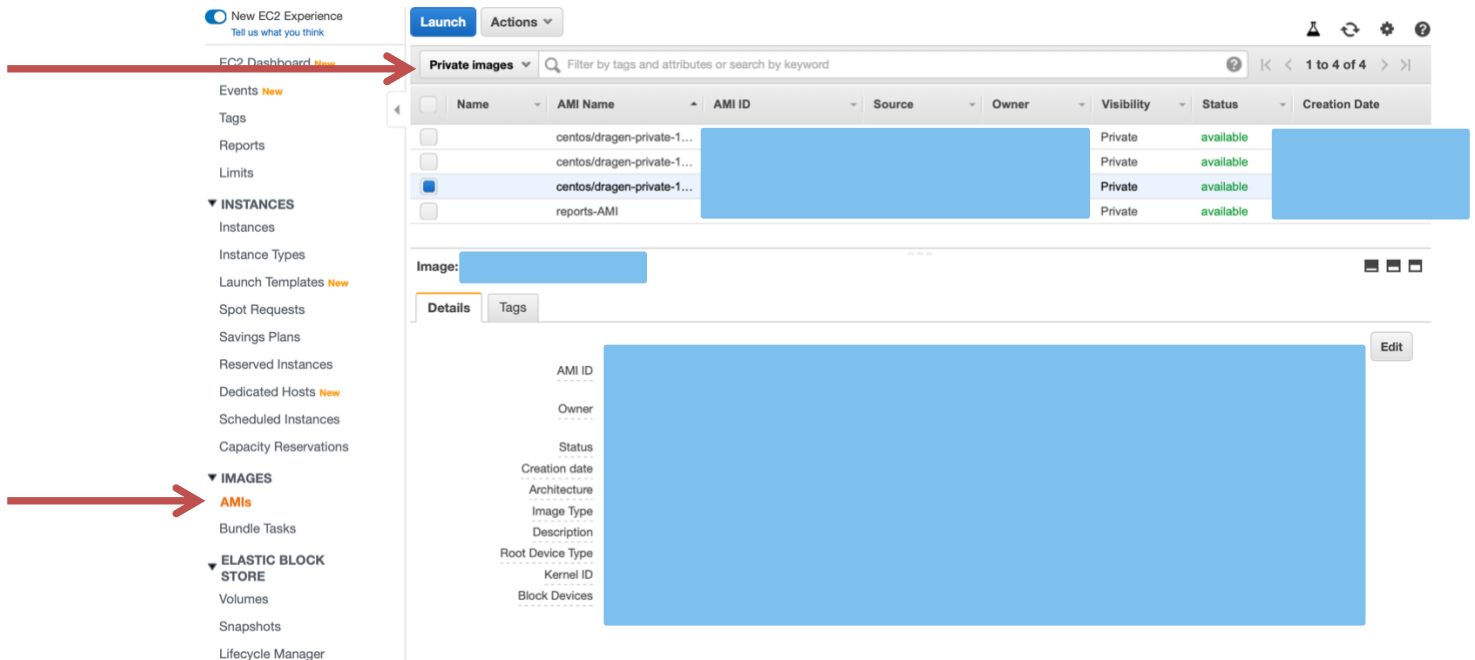
Launch stack:

```
$ aws --region us-east-1 cloudformation create-stack --stack-name clinical-dragen-dev41 --template-url  
https://hgsccl-op-data.s3.amazonaws.com/cloudformation/clinical-dragen-dev/templates/master.yaml --  
parameters file://ci/params.json --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM  
CAPABILITY_AUTO_EXPAND --tags Key=env,Value=prod
```

Celeste

UPDATING THE DRAGEN AMI WITHIN TEMPLATES

1. From the EC2 Management Console, navigate to “AMI” and select “Private AMIs” to view those shared with our account (not owned by our account).



2. Select the latest DRAGEN AMI and note the DRAGEN “AMI Name” which contains the software release version (e.g. a software release is 3.4.12). Also note the “AMI ID”.
3. Navigate to your local copy of *Celeste* repo.
4. We will find the templates in which the DRAGEN AMI are specified via the following commands:

```
$ cd templates
```

```
$ for file in *.yaml; do echo $file; cat $file | grep ami; done
```

```
batch-roles.yaml  
clean-bucket-repository.yaml  
clean-bucket.yaml  
clean-repository.yaml  
container-build.yaml  
copy.yaml  
docker-bucket-repository.yaml  
dragen-3.4.12.yaml
```

Description: Amazon EC2 instance type. DRAGEN requires an FPGA to run, so the instance type must be in the F1 instance family.

The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

```
DRAGEN: ami-1234#####  
DRAGEN: ami-1234#####  
dragen-batch.yaml  
job-tracker-resub-handler.standalone.yaml  
master.yaml  
...
```

- From the above, note that the dragen ami is only specified in the dragen-3.4.12.yaml template (which is conveniently named to indicate this very fact). We will update the AMI id within this template and update the template name, as well.

```
$ vi dragen-3.4.12.yaml  
<edit lines with ami-id to reflect new ami-id then save and close>  
  
$ mv dragen-3.4.12.yaml dragen-<new-version-number>  
  
$ for file in *.yaml; do echo $file; cat $file | grep dragen-3.4.12; done  
batch-roles.yaml  
clean-bucket-repository.yaml  
clean-bucket.yaml  
clean-repository.yaml  
container-build.yaml  
copy.yaml  
docker-bucket-repository.yaml  
dragen-3.4.12.yaml  
dragen-batch.yaml  
job-tracker-resub-handler.standalone.yaml  
master.yaml  
    TemplateURL:                                     !Sub  
    https://${QSS3BucketName}.s3.amazonaws.com/${QSS3KeyPrefix}templates/dragen-3.4.12.yaml  
reports-batch-9.19.parent.yaml  
s3-config.standalone.yaml  
sns-lambdas.yaml
```

- From the above, note that we have to edit the path of the dragen-<version>.yaml file from within the master.yaml template.

```
$ vi master.yaml  
<edit lines with dragen-3.4.12.yaml to reflect new version/filename then save and close>
```

Celeste

AWS SIMPLE NOTIFICATION SERVICE (SNS)

Why SNS?

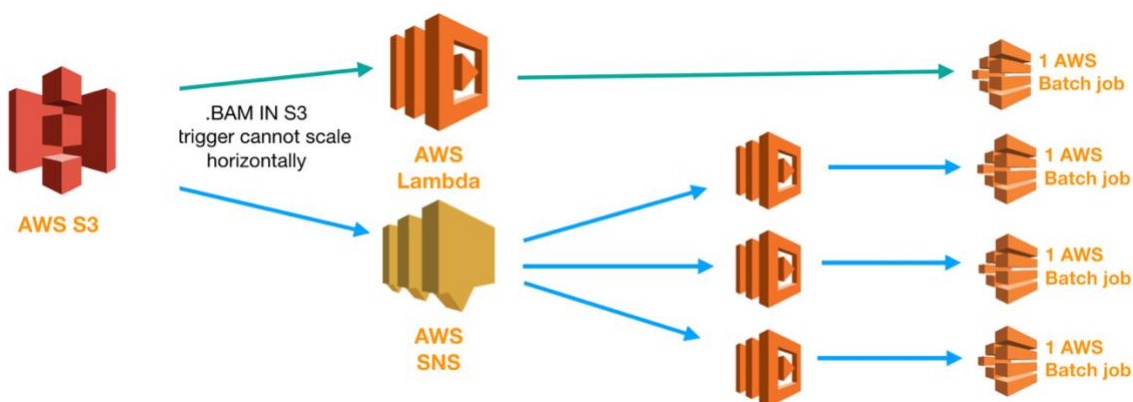
Look back at Figures 1 and 2 at the beginning of this guide. An astute observer might note that S3 events can trigger an AWS Lambda function directly. This begs the question, why then, is AWS SNS a part of the existing *Celeste* infrastructure?

Introducing a pub/sub service (AWS SNS) between S3 and Lambda is a very powerful hallmark of **decoupled, microservices architecture**. The general idea is to separate (or decouple) message producers (in this case, S3) from message subscribers (in this case, Lambda) so that each can be developed, expanded, scaled, and/or deployed independently of the other.

Adding this separation also makes any two services more resilient to system failure. When the interaction between two entities is tightly coupled, the inability of one to scale directly limits the ability of the second to scale. The failure of one service will bring down the second service.

In the architecture diagrammed in Figure 2 at the beginning of this guide, the yellow and purple arrows illustrate that I can push the **same** message dynamically to a wide range of endpoints -- whether they are Lambdas, emails, or an HTTP/HTTPS endpoint... the possibilities are far-reaching to several millions of scaled interactions.

The below image illustrates how a direct connection between S3 and Lambda leaves little room to scale the S3 notification horizontally in a case where one .bam file should trigger the submission of multiple AWS Batch jobs (quality control jobs such as AlignStats, VerifyBamID, etc.).



Even if one were to begin with only one SNS -> Lambda subscription, this design provides the infrastructure the flexibility to scale and evolve systems based on business demands in the future, without a complete re-engineering. If the system began with one S3 -> Lambda direct

Celeste

connection, a lot of future development would be restricted based on stringent S3 event notification rules (cannot have overlapping suffixes/prefixes in certain cases with similar event types).

SNS is also extremely useful in the case of a heterogeneous system. For example, it is useful in cases where multiple buckets or environments all need to dynamically feed into one workflow.

Celeste

Overview of the Job Tracking System

Jobs that hit FAILED status in us-east-1 are routed via SNS to a number of email subscription endpoints. A separate AWS Batch job change notification specific to FAILED or SUCCEEDED jobs (basically, jobs that have completed) is forwarded via CloudWatch to a serverless function.

1. If a job has failed due to a Spot Instance interruption, this Lambda detects that and automates the resubmission of the job for resilient failure handling without human-involvement.

How to Set-up an SNS Notification

The AWS documentation for SNS includes a tutorial detailing how these notifications were set-up, as well as tutorials for specific cases. (See <https://docs.aws.amazon.com/sns/latest/dg/sns-tutorial-create-topic.html>)

Case-Study: The Process of Creating A JobFail SNS Notifications



Description: The system outlined in the image above ensures that any AWS Batch job failure in the specified AWS region will publish a message to SNS. Email subscribers will receive these published messages from AWS Batch. Let's imagine that the organization again made the decision to migrate to a new AWS account, or expand the existing account into the Orgeon region. How would we set this system up from scratch? Ideally, you would utilize a cloudformation template that specifies all of your resources. We will walk through the creation of the above diagrammed workflow in the console and from this exercise, you can check your understanding by developing your own template that provisions these resources!

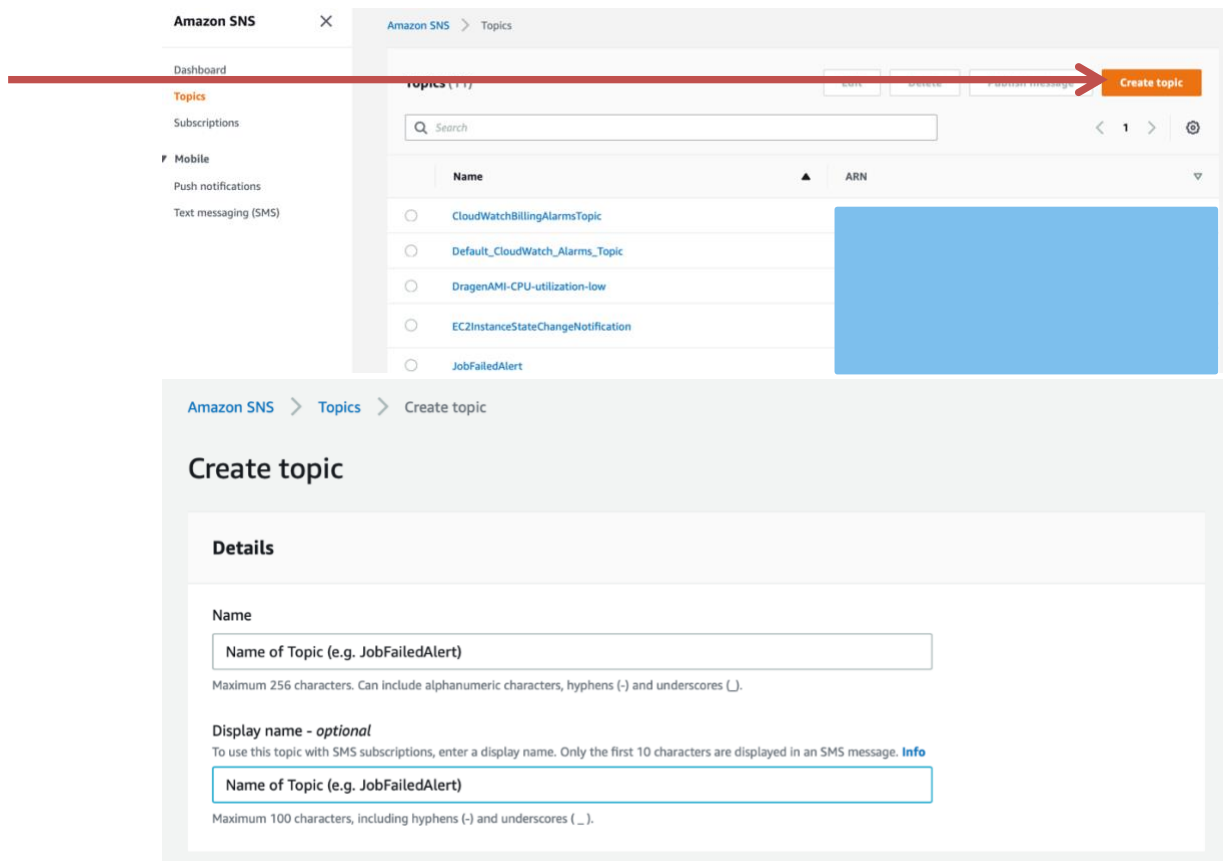
This entire process is perfectly documented in AWS as a tutorial that matches our environment workflow exactly:

(See https://docs.aws.amazon.com/batch/latest/userguide/batch_sns_tutorial.html)

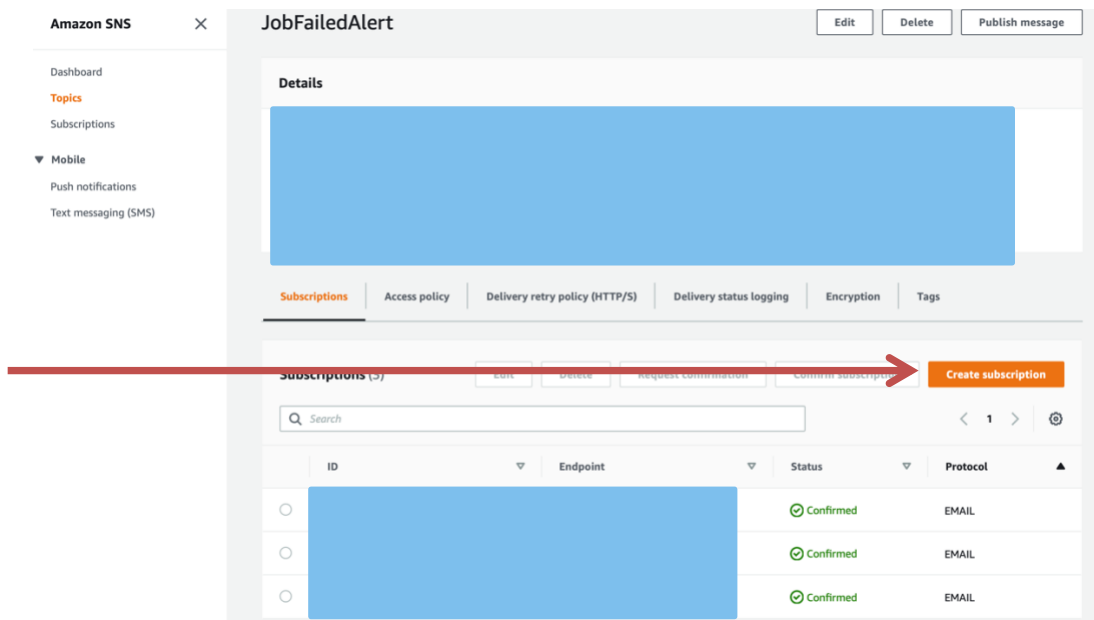
Let's work through the steps together. I will share a few relevant screenshots to build upon the tutorial linked and referenced in the line above.

1. For step one, we would like to create an SNS Topic. Follow "Step 1: Create and Subscribe to an Amazon SNS Topic" in the link above. Below are screenshots of the process to further guide you.

Celeste



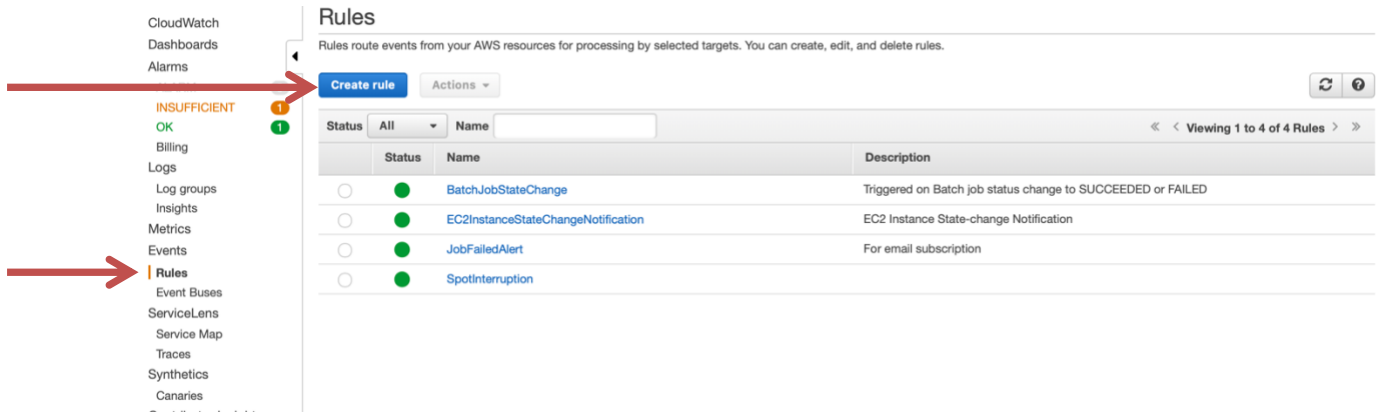
Once the topic is created, you can click on the topic name and select “create” subscriptions. Create an email subscription to the topic as described in the AWS tutorial link above.



The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

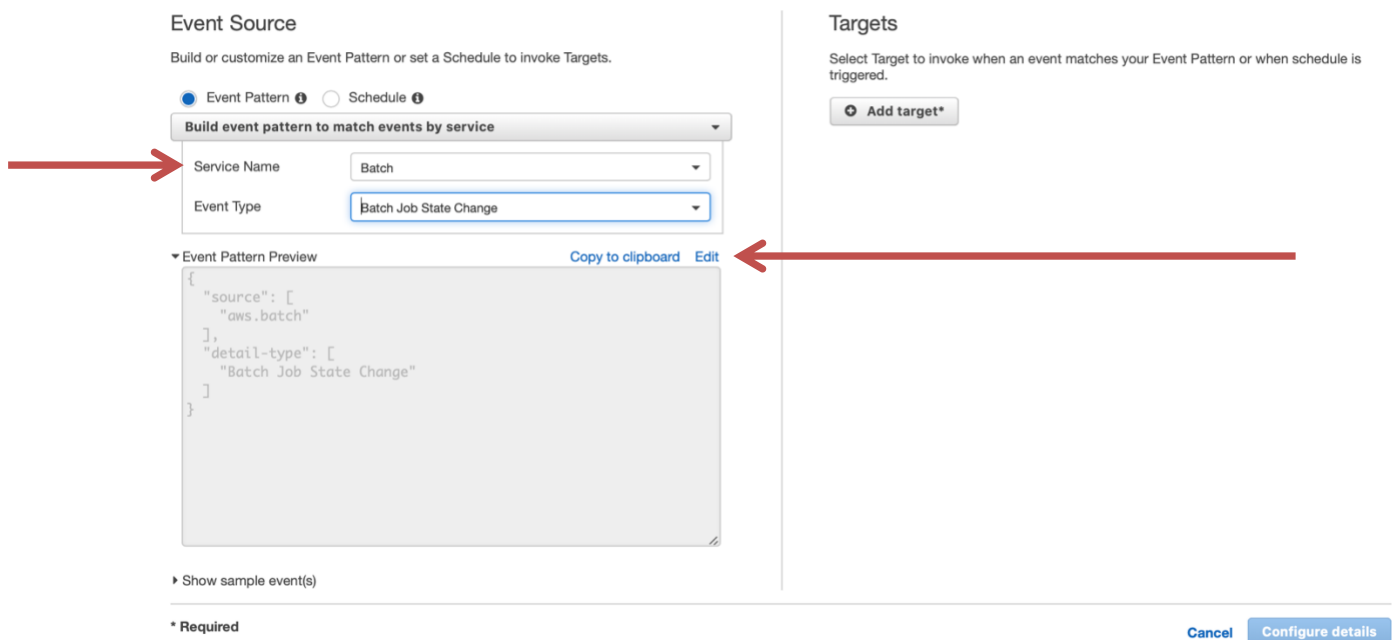
2. AWS Batch can send job data to CloudWatch Events for monitoring. Navigate to the CloudWatch Management Console and Select “Rules” under “Events”.
3. Next hit “Create Rule”



4. Under “Event Source” select the service name (Batch) and Event Type (Batch Job State Change). We will modify the event pattern json to only track job failures.
 - a. Select “Edit” under “Event Pattern Preview”. Match the pattern to (b), below.
 - b. `{ "source": ["aws.batch"], "detail-type": ["Batch Job State Change"], "detail": { "status": ["FAILED"] } }`

Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.



The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

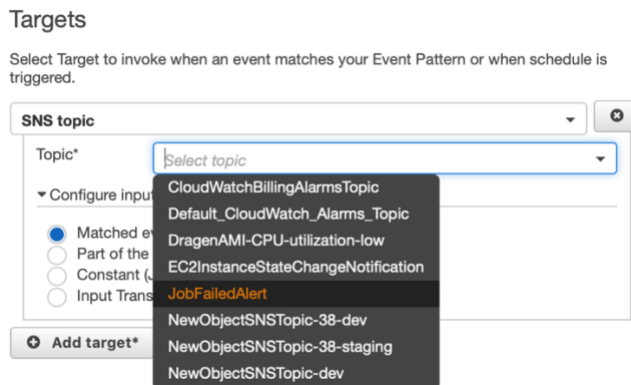
Celeste

- c. Let's say I wanted to track all state changes in the future, or only Batch jobs that have reached a "SUCCEEDED" status. I could list all valid job status types in the json document in orange, above. Here is a description of job states for further reading.

(See:

https://docs.aws.amazon.com/batch/latest/userguide/job_states.html)

5. Select the name of the SNS Topic you made in Step 1 for "Targets". Once this is complete, you can hit "Configure Details" and enter a rulename and description. Then select "Create Rule".



6. The process is complete! You can test the system by submitting a batch job that you know will fail. This is described in Step 3 in the AWS tutorial linked above. I suggest submitting a dra-gen job that you know does not have a valid command so you can review the notification details and correlate these with known attributes of the dra-gen job you tried.

Celeste

AWS BATCH

At its core, AWS Batch is a service that allows users to run and scale containerized workloads in the cloud. Very simply, Batch allows you to link a certain command with a certain docker image for execution of that command. Batch manages the horizontal scaling of resources and efficient task distribution of many simultaneous running jobs while utilizing the AWS Elastic Container Service (a container orchestration platform) under its hood.

(See: <https://docs.aws.amazon.com/batch/latest/userguide/what-is-batch.html> and This introductory video: <https://www.youtube.com/watch?v=T4aAWrGHmxQ>)

Components

Job

Same idea as an HPC job; a basic unit of work.

Job Queue

Synonymous with the construct for queues on a cluster. However, these queues are relatively simple to make and edit. You can assign priority values to them easily and connect or detach them from compute environments, as needed.

Utilize the following command:

```
aws batch describe-job-queues --output table
```

Note that we have several queues depending on job type and environment.

CELESTE QUEUES

dragen-queue-prod → for dragen jobs

reports-queue-prod → for reports jobs like AlignStats and VerifyBamID

reports-plus-queue-prod → for reports jobs like that require more resources

Each of the queues above has 1-3 associated “compute environments.”

Compute Environment

The compute resources (instance type, billing construct, IAM roles) associated with a queue. We utilize managed environments so that AWS Batch handles scaling.

Utilize the following command:

```
aws batch describe-compute-environments --output table
```

Celeste

Note: dragen-spot-dev, dragen-on-demand-dev

The above are two of the compute environments currently associated with *dragen-queue-dev*. They both belong to the development environment (as opposed to “staging” or “prod”) and differ only by billing construct (spot versus on demand) and bid price (e.g. 40% or 50% bid price for the spot environments).

For reference: The dragen compute environments specify the use of f1.4xlarge instances. These belong to the “F family” of instances which utilize FPGAs for accelerated computing. (See the specs: <https://aws.amazon.com/ec2/instance-types/f1/>)

Dragen currently uses only 1 FPGA under-the-hood, the additional speed-up in terms of runtimes on f1.4xlarge or f1.16xlarge instances over f1.2xlarge is due to the additional vCPUs.

Job Definition

The definition specifies parameters that will be supplied to the job (e.g. number of vCPUs and memory), as well as the **docker image**, environmental variables, and the basic command that will be supplied at run time. Certain pieces of the job definition can be over-written at runtime.

Utilize the following command:

```
aws batch describe-job-definitions --job-definition-name dragen --output json
```

Note that there are (currently) x number of revisions for the “dragen” job. The different versions are an artifact of testing combined with the fact that job definitions can have *active* or *inactive* states. Look for the “Active” revision(s). You can specify any active revision at job submission time (e.g. using the form “dragen:12” for the 12th revision of the “dragen” job definition).

To begin creating a job definition:

(See: <https://docs.aws.amazon.com/batch/latest/userguide/create-job-definition.html>)

```
aws batch register-job-definition --generate-cli-skeleton  
aws batch register-job-definition help
```

Celeste

Look at the below job definition for AlignStats. Note that the command is empty. In our case the command is supplied when submitting the batch job itself. To run an AlignStats job in the development environment, I would specify the below job definition and job command (e.g. `-i <input> -o <output> ...`).

Also note the job **retry** number. This is the amount of times Batch will automatically re-submit a job that encounters a FAILED status, for any reason.

```
$ aws batch describe-job-definitions --job-definition-name reports-alignstats-dev --status ACTIVE
```

```
{
  "jobDefinitions": [
    {
      "jobDefinitionName": "reports-alignstats-dev",
      "jobDefinitionArn": "arn:aws:batch:us-east-1:<account>:job-definition/reports-alignstats-dev:7",
      "revision": 7,
      "status": "ACTIVE",
      "type": "container",
      "parameters": {},
      "retryStrategy": {
        "attempts": 5
      },
      "containerProperties": {
        "image": "<account>.dkr.ecr.us-east-1.amazonaws.com/alignstats0.9.1",
        "vcpus": 1,
        "memory": 7000,
        "command": [],
        "jobRoleArn": "arn:aws:iam::<account>:role/clinical-dragen-dev2-ReportsStack-58H9QASR-JobRole-OJL1DYZMU4V9",
        "volumes": [],
        "environment": [],
        "mountPoints": [],
        "ulimits": [],
        "resourceRequirements": []
      }
    }
  ]
}
```

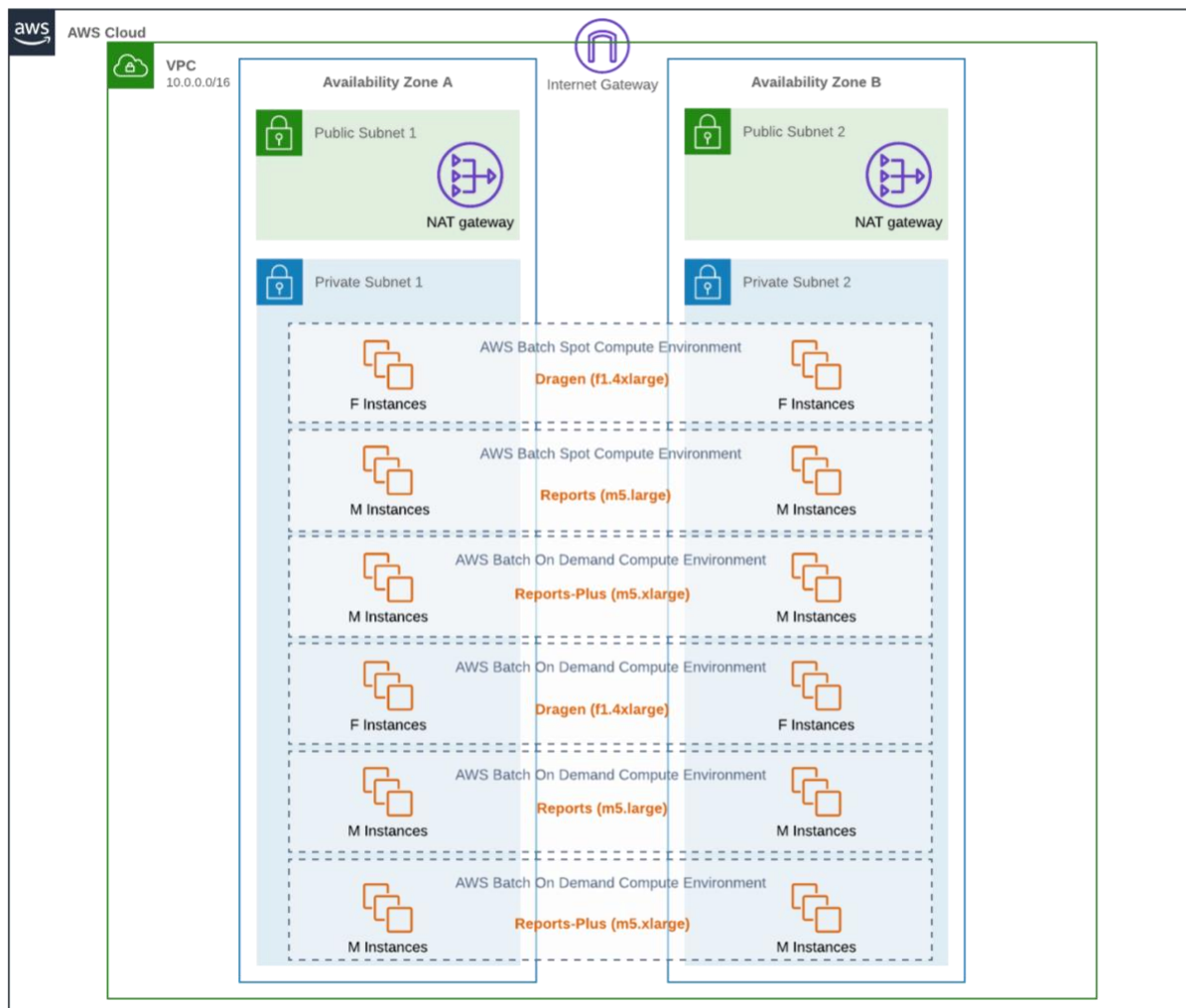


Celeste

AWS Batch and VPC Architecture

Taking an example VPC (abstracting the Availability Zones as “A” and “B” and assuming a 2 AZ deployment), let’s look at the architecture diagram below. The example VPC (green outlined box) deploys highly available architecture across 2 Availability Zones (AZs), each configured with a private and public subnet in accordance with AWS best practices.

The AWS Batch compute environments for DRAGEN and quality control reports exist across the two private subnets with the specified instance types and billing constructs. Each queue is connected to both an on-demand compute environment and a spot compute environment with a bid price of 100% to sustainably utilize more cost-effective compute resources without being kicked off during fluctuations in the market.



The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

The public subnets contain a NAT Gateway each to allow outbound internet access to jobs in the private subnets.

This closely imitates and builds upon the architecture deployed via the “Illumina Dragen on AWS” QuickStart. (See: <https://aws.amazon.com/quickstart/architecture/illumina-dragen/>)

A Suggested Multi-Environemtn VPC IPv4 CIDR Scheme

| VPC | CIDR |
|------------|---------------|
| Test | 10.60.0.0/16 |
| Dev | 10.0.0.0/16 |
| Staging | 10.40.0.0/16 |
| Prod | 10.10.0.0/16 |
| Management | 10.100.0.0/16 |

(See: <https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html> and <https://docs.aws.amazon.com/vpc/latest/userguide/working-with-vpcs.html>)

A Note on Availability Zones

The availability zone names are not consistent across accounts. In order to compare AZs between two accounts, you’d have to look up the AZ ID mapping for each account in the Resource Access Manager console. Instructions are here: <https://docs.aws.amazon.com/ram/latest/userguide/working-with-az-ids.html>. That’s why it may look like one account can get capacity in an AZ while another cannot.

That being said, the best practice is to use the widest range of AZs possible across accounts and avoid targeting specific AZs.

Celeste

Spot Instances

Spot instances allow AWS users to run hyperscale workloads at up to 90% off of the on-demand price for any given server-type. *Celeste* utilizes several spot compute environments to significantly reduce cloud costs.

(See: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html>)

At any given time, these savings may be visualized by navigating to the EC2 Management Console and selecting “Spot Requests” followed by “Savings Summary”.

The screenshot shows the AWS EC2 Management Console interface. On the left is a navigation menu with categories like INSTANCES, IMAGES, ELASTIC BLOCK STORE, NETWORK & SECURITY, and LOAD BALANCING. A red arrow points to 'Spot Requests' under the INSTANCES category. The main content area shows the 'Request Spot Instances' page with tabs for 'Request Spot Instances', 'Actions', 'Pricing History', and 'Savings Summary'. The 'Savings Summary' tab is active, displaying a summary of savings across running and recently terminated Spot Instances. The summary includes a table with columns for Request Id, Request type, Instance type, State, Capacity, Status, Persistence, and Created. Below this, a 'Savings Summary' box shows a high-level summary of savings, including a table with columns for Spot Instances, vCPU-hours, Mem(GiB)-hours, On-Demand total, and Spot total. The table shows 49 Spot Instances, 1552 vCPU-hours, 23668 Mem(GiB)-hours, \$320.10 On-Demand total, and \$96.03 Spot total, resulting in 70% savings. A 'Details' section below shows a table with columns for Instance type, vCPU-hours, Mem(GiB)-hours, Total cost, and Savings. The table shows f1.4xlarge (49) instances, 1552 vCPU-hours, 23668 mem(GiB)-hours, \$96.03 total, and 70% savings. A red arrow points to the 'Savings Summary' tab in the main content area.

| Request Id | Request type | Instance type | State | Capacity | Status | Persistence | Created |
|------------|--------------|---------------|-------|----------|--------|-------------|---------|
| ... | ... | ... | ... | ... | ... | ... | ... |

Savings Summary

A high-level summary of your savings across all of your running and recently terminated Spot Instances. For detailed reporting on your account-level Spot usage, visit [Cost Explorer](#).

Spot usage and savings

| | | | | |
|----------------------|--------------------|-------------------------|-----------------------------|-----------------------|
| 49 Spot Instances | 1552 vCPU-hours | 23668 Mem(GiB)-hours | \$320.10 On-Demand total | \$96.03 Spot total |
| 70% Savings | | | | |

Average cost per vCPU-hour: \$0.0619
Average cost per mem(GiB)-hour: \$0.0041

Details

| | | | | |
|-----------------|-----------------|----------------------|---------------|-------------|
| f1.4xlarge (49) | 1552 vCPU-hours | 23668 mem(GiB)-hours | \$96.03 total | 70% savings |
|-----------------|-----------------|----------------------|---------------|-------------|

* Spot savings are estimated savings and may differ from actual savings. This is because the savings shown on this page do not include the billing adjustments for your usage.

Close

Spot instances are finicky. As market prices or supply and demand fluctuate, AWS may reclaim the spot EC2 capacity and terminate your running job. If a spot instance terminates, the job will retry on another instance with a new job retry number. Batch will retry the job on another instance with a new ECS task ID. If a job is continuously kicked off of spot instances to the point where it runs out of retries, the job hits “FAILED” status with the status reason “Host EC2 (instance <ID>) terminated”.

Let’s say a DRAGEN job with a retry number of 3 is running on an f1.4xlarge spot instance when the spot instance is suddenly reclaimed by AWS. The job will not hit FAILED status yet and the first of two job retries will begin. If the spot instance is again reclaimed by AWS, the job will hit the second of its two retries. If the job successfully completes and encounters no more host

Celeste

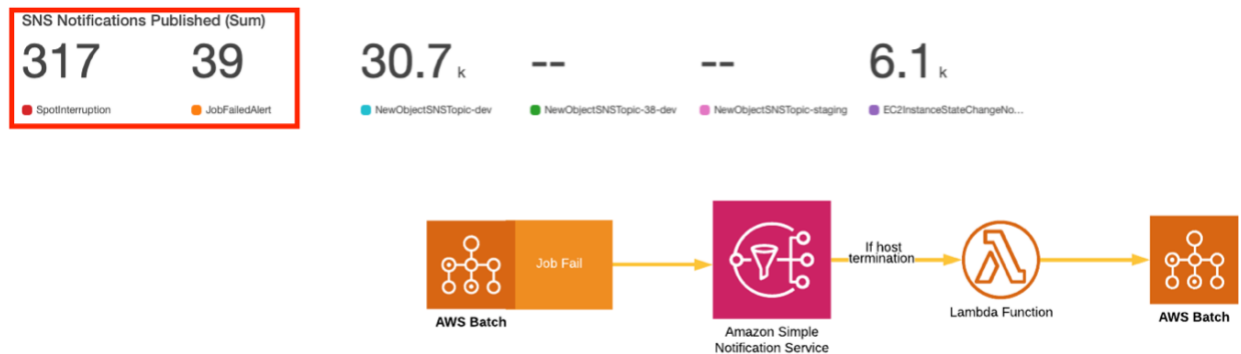
instance termination, then the job will show as SUCCEEDED. If however, the host EC2 instance is again terminated due to bid price or demand, the job will fail.

Due to the high volume of samples running through *Celeste*, at any given time, a portion of the workload will run on spot instances. Market price or supply and demand may fluctuate enough to lead to a large volume of spot interruptions. The pipeline system is engineered to automate the handling of such failures.

Case-Study: Automating Job Retries After Spot Interruption

Description: Let's look at a real case. In a span of a few days, thousands of dragen pipeline jobs were submitted. During this period, 317 spot instances with pipeline tasks were interrupted. Due to the high default retry number on all of the running jobs, not all jobs failed. However, 39 jobs maxed out their retry number and hit "FAILED" status.

To avoid a case in which the spot interruptions and job failures spiral out of control, *Celeste* routes AWS Batch job information to AWS SNS. A Lambda subscribed to this SNS topic checks if the job has failed. If the job has failed and is noted to have an exit status related to host EC2 termination, this serverless function will automate the resubmission of the job to AWS Batch.



Celeste

Good Practices for *Celeste* Batch Job Submission

Ensure jobs are properly tracked by including PARAMETERS in job submission.

```
[13] → cat job-input-json-shell.json
{
  "jobName": "some-name",
  "jobQueue": "some-queue-dev",
  "jobDefinition": "some-defn-dev",
  "containerOverrides": {
    "command": [
      "--input",
      "inputfile",
      "--output",
      "out"
    ]
  },
  "retryStrategy": {"attempts": 2},
  "parameters": {"project": "AoU-or-some-project", "user": "some-user-or-team", "hgsccl:env": "dev"}
}
```

Parameters can include the minimum key:value pairs for user:<user-name/team> or even for project. See where these occur in the example batch job submission cli-input-json shell above.

Troubleshooting AWS Batch Jobs

Suggested Reading

AWS Documentation: “Troubleshooting AWS Batch”

- <https://docs.aws.amazon.com/batch/latest/userguide/troubleshooting.html>

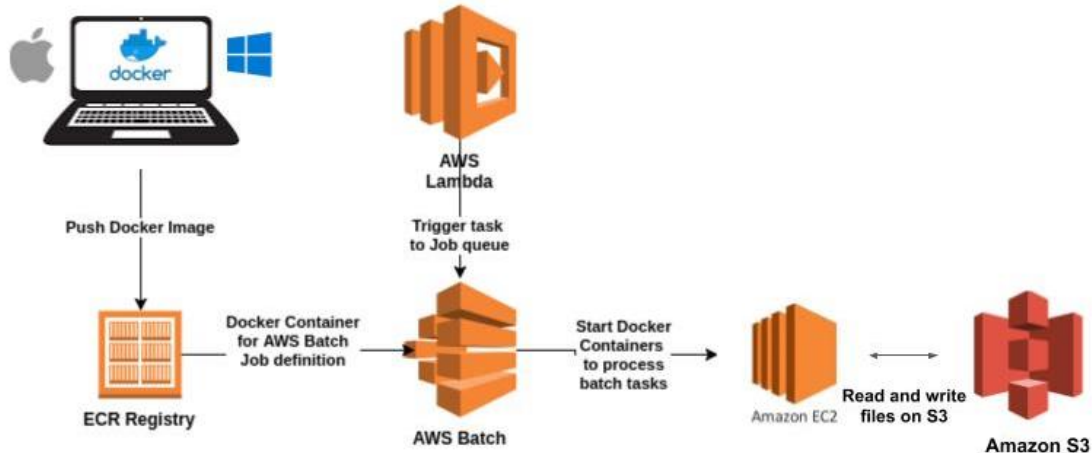
AWS Support: “Why is my AWS Batch job stuck in RUNNABLE status?”

- <https://aws.amazon.com/premiumsupport/knowledge-center/batch-job-stuck-runnable-status/>

Celeste

Containers Within the *Celeste* Architecture

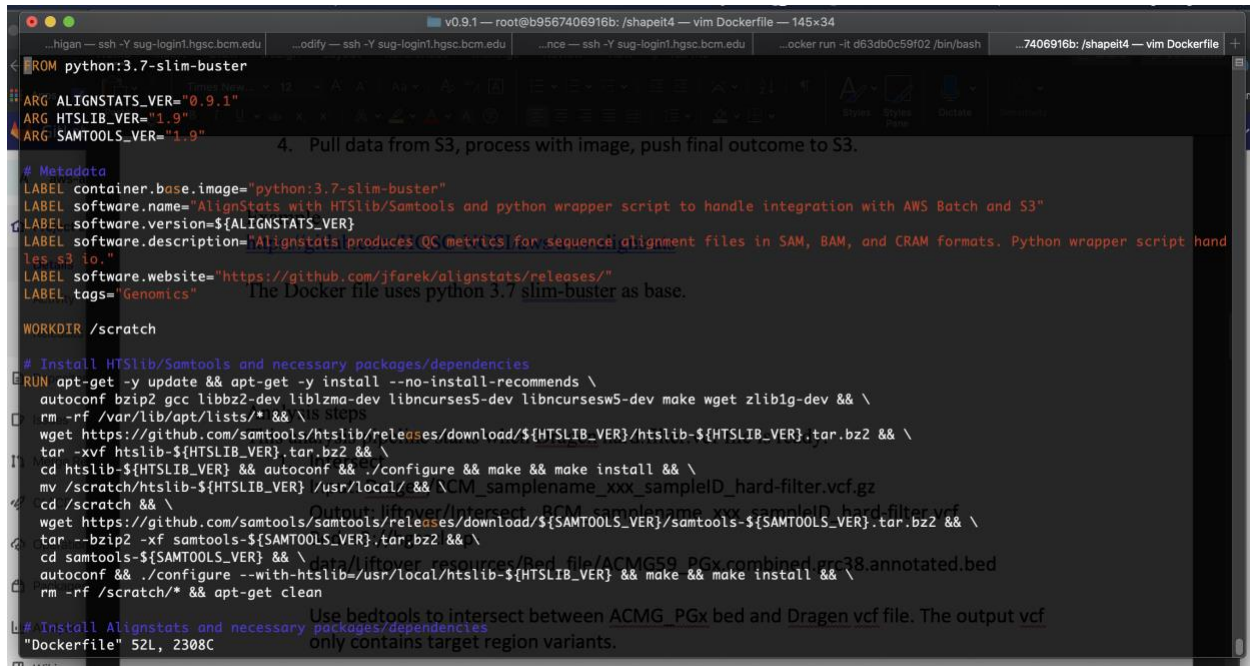
Development Steps



1. Build image and test docker image on local machine, include a wrapper script that can handle conditional I/O to and from S3.
2. Push the completed image to AWS ECR.
3. Connect a new AWS Batch job definition with the ECR image arn.
4. Add a Lambda that can trigger AWS Batch job submission (and ensure this function has the appropriate lambda to batch invoke permissions)
5. Pull data from S3, process with image, push final outcome to S3.

Celeste

Example



```
FROM python:3.7-slim-buster
ARG ALIGNSTATS_VER="0.9.1"
ARG HTSLIB_VER="1.9"
ARG SAMTOOLS_VER="1.9"

# Metadata
LABEL container.base.image="python:3.7-slim-buster"
LABEL software.name="AlignStats with HTSLib/Samtools and python wrapper script to handle integration with AWS Batch and S3"
LABEL software.version=${ALIGNSTATS_VER}
LABEL software.description="Alignstats produces QC metrics for sequenced alignment files in SAM, BAM, and CRAM formats. Python wrapper script handles S3 IO."
LABEL software.website="https://github.com/jfarek/alignstats/releases/"
LABEL tags="Genomics"

WORKDIR /scratch

# Install HTSLib/Samtools and necessary packages/dependencies
RUN apt-get -y update && apt-get -y install --no-install-recommends \
    autoconf bzip2 gcc libbz2-dev liblzma-dev libncurses5-dev libncursesw5-dev make wget zlib1g-dev && \
    rm -rf /var/lib/apt/lists/* && \
    wget https://github.com/samtools/htslib/releases/download/${HTSLIB_VER}/htslib-${HTSLIB_VER}.tar.bz2 && \
    tar -xvf htslib-${HTSLIB_VER}.tar.bz2 && \
    cd htslib-${HTSLIB_VER} && autoconf && ./configure && make && make install && \
    mv /scratch/htslib-${HTSLIB_VER} /usr/local/ && \
    cd /scratch && \
    wget https://github.com/samtools/samtools/releases/download/${SAMTOOLS_VER}/samtools-${SAMTOOLS_VER}.tar.bz2 && \
    tar --bzip2 -xf samtools-${SAMTOOLS_VER}.tar.bz2 && \
    cd samtools-${SAMTOOLS_VER} && \
    autoconf && ./configure --with-htslib=/usr/local/htslib-${HTSLIB_VER} && make && make install && \
    rm -rf /scratch/* && apt-get clean

# Install Alignstats and necessary packages/dependencies
"Dockfile" 52L, 2308C
```

Most of our docker containers uses python 3.7 slim-buster as base. We then add **ARG** and **LABEL** lines to describe and define its function.

WORKDIR define where to run the job in docker image.

RUN can install applications, packages, and wrapper scripts.

COPY specified what files need to be copy from Docker folder to image and do the job.

ENV defines environment path.

ENTRYPOINT and **CMD** are for running the job.

Celeste

Scripts in this Example

```
...higan — ssh -Y sug-login1.hgsc.bcm.edu | ...odify — ssh -Y sug-login1.hgsc.bcm.edu | ...nce — ssh -Y sug-login1.hgsc.bcm.edu |
tsungjuw@TJ2-MacBook-Pro src % cd ..
tsungjuw@TJ2-MacBook-Pro v0.9.1 % ls
Dockerfile      src
tsungjuw@TJ2-MacBook-Pro v0.9.1 % cd src
tsungjuw@TJ2-MacBook-Pro src % ls
alignstats_aws.py  aws_tools  report-generator.py
tsungjuw@TJ2-MacBook-Pro src %
```

`alignstats_aws.py` is the primary running script.

This script serves several purposes:

1. Taking arguments as input
2. Handles the input command and download files from S3.
3. Compose running command
4. When the run complete, generate report and push files to S3.

```
tsungjuw@TJ2-MacBook-Pro src % cd aws_tools
tsungjuw@TJ2-MacBook-Pro aws_tools % ls
__init__.py  __pycache__  s3utils.py
tsungjuw@TJ2-MacBook-Pro aws_tools % vim s3utils.py
tsungjuw@TJ2-MacBook-Pro aws_tools %
```

Another important script is `s3utils.py`.

This set of functions, or utilities does the following:

1. Handle s3 input. Parse and download the file.
2. Remove file full path and keep only filename for later analysis use
3. Take local output files, assemble the S3 output bucket/folder path
4. Push the file to S3

Celeste

AWS LAMBDA

Recommended Reading and Tutorials:

- <https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html>
- <https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-awscli.html>
- <https://docs.aws.amazon.com/lambda/latest/dg/with-sns.html> and the tutorial that follows
- <https://docs.aws.amazon.com/lambda/latest/dg/python-package.html>
- **Testing:** <https://aws.amazon.com/blogs/compute/improved-testing-on-the-aws-lambda-console/>
- <https://docs.aws.amazon.com/lambda/latest/dg/configuration-versions.html>

Celeste

AWS SIMPLE STORAGE SERVICE (S3)

Introduction to S3

In AWS S3, “buckets” and “objects” are the primary resources, with objects stored within buckets. **Unlike a file system**, S3 has a flat structure.

Beginner S3 Commands

1. See what is in a space in the bucket using the following command:
`aws s3 ls s3://bucket/prefix/`
2. An S3 equivalent to the UNIX “tree” command:
`aws s3 ls --summarize --human-readable --recursive s3://bucket/`
3. To some extent, S3 creates an illusion of directories when objects have a trailing “/”. You cannot upload an object key name with a trailing “/” using the S3 console, but you can do this using S3 API. To create a new object in your bucket:
`aws s3api put-object --bucket <bucketname> --key <prefixBUTNOTADIR/>`
4. To copy only fastqs to S3 from a directory of mixed file-types:
`aws s3 cp <your directory path> s3://<your bucket name>/ --exclude "*" --include "*.fastq.gz"`
5. Sync directories and S3 prefixes to recursively copy new and updated files from the source directory to the destination. This only creates “folders” in an S3 destination if they contain 1 or more files:
`aws s3 sync <your local path> s3://<your bucket name>/`
6. Sync data from S3 storage to your storage space. You can use filters for --exclude --include and —dryrun, among a bevy of other options.
`aws s3 sync s3://<your bucket name>/ <your local path>`
7. It’s complicated to search within an S3 bucket without knowing the full PREFIX of an object. However, you can use this dryrun sync trick to get “search results” if you only have a target infix or suffix:
`aws s3 sync --dryrun --exclude "*" --include "*<infix>*<suffix>" s3://<your bucket name>/ <your local path>`

Checksums

(See: <https://docs.aws.amazon.com/cli/latest/topic/s3-faq.html>)

Celeste

Understanding Select S3 References

Buckets

- a. **BUCKETNAMES “cicd-pipeline...”, “clinical-dragen-..”:** These artifact buckets are auto-generated by CloudFormation when provisioning a new environment. The logic that generates some of the “clinical-dragen-..” artifact buckets is contained within the original “Illumina DRAGEN on AWS” QuickStart that HGSC-CL infrastructure is built upon.

(See <https://aws.amazon.com/quickstart/architecture/illumina-dragen/>).

A CI/CD pipeline within the Illumina QuickStart uses the instructions and code contained within /app/source/dragen (in the Celeste GitHub repository) to build the DRAGEN container. This process utilizes AWS CodePipeline and will create an artifact S3 bucket.

The buckets “cicd-pipeline...” are part of the continuous integration and delivery pipeline for testing infrastructure deployments. Our environment deployment tests are automated via AWS CodePipeline. To learn more about input and output to an artifact S3 bucket, reference:

<https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome-introducing-artifacts.html>.

Do not delete any artifact bucket that is part of an active CodePipeline. This may result in the inability of the pipeline to run.

To enable reliable global deployment of a CloudFormation template to any AWS Region supported by Lambda, a copyZips lambda function contained within the CloudFormation assets also copies lambdas within /app/packages/ to an artifact bucket in the same AWS region as the stack. To understand this, read:

<https://aws.amazon.com/blogs/infrastructure-and-automation/deploying-aws-lambda-functions-using-aws-cloudformation-the-portable-way/>.

This is in accordance with AWS best practices. Emptying and deleting these buckets for an environment that is no longer active/does not exist may be considered, however, please double check before deleting and ensure **no active CodePipeline buckets are deleted**.

For more details on the DRAGEN QuickStart, contact the DRAGEN team directly.

- b. **BUCKETNAMES “tcat...”:** Again, these buckets are auto-generated by CloudFormation when provisioning a new environment using the testing tool known as “TaskCat”. (See <https://aws.amazon.com/blogs/infrastructure-and-automation/a-deep-dive-into->

Celeste

[testing-with-taskcat/](#)). These can be emptied and deleted if they are no longer being utilized after a successful taskcat test. **These are unneeded because the “test” env is ephemeral and does not “exist” as a real environment utilized by the organization.**

Celeste

Prefixes within s3://<bucketname> (referred to as Genomics S3 Bucket)

Currently, the main object prefixes to note in the HGSC-CL development bucket are as follows:

- PREFIX clinical-dragen-pipeline:** Where fastqs are uploaded, hs37d5 initial alignments and hs37d5 merge event data are stored
- PREFIX 38-clinical-dragen-pipeline:** Where GRCh38 merge event data is stored

hgscccl-dev

Overview Properties Permissions Management Access points

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions Versions Hide Show

US East (N. Virginia)

Viewing 1 to 7

| Name | Last modified | Size | Storage class |
|-----------------------------|----------------------------------|--------|---------------|
| 38-clinical-dragen-pipeline | -- | -- | -- |
| NGIRD | -- | -- | -- |
| clinical-dragen-pipeline | -- | -- | -- |
| cloudformation | -- | -- | -- |
| misc | -- | -- | -- |
| scratch | -- | -- | -- |
| test-archival-object.txt | Dec 17, 2019 2:27:00 PM GMT-0600 | 13.0 B | Glacier |

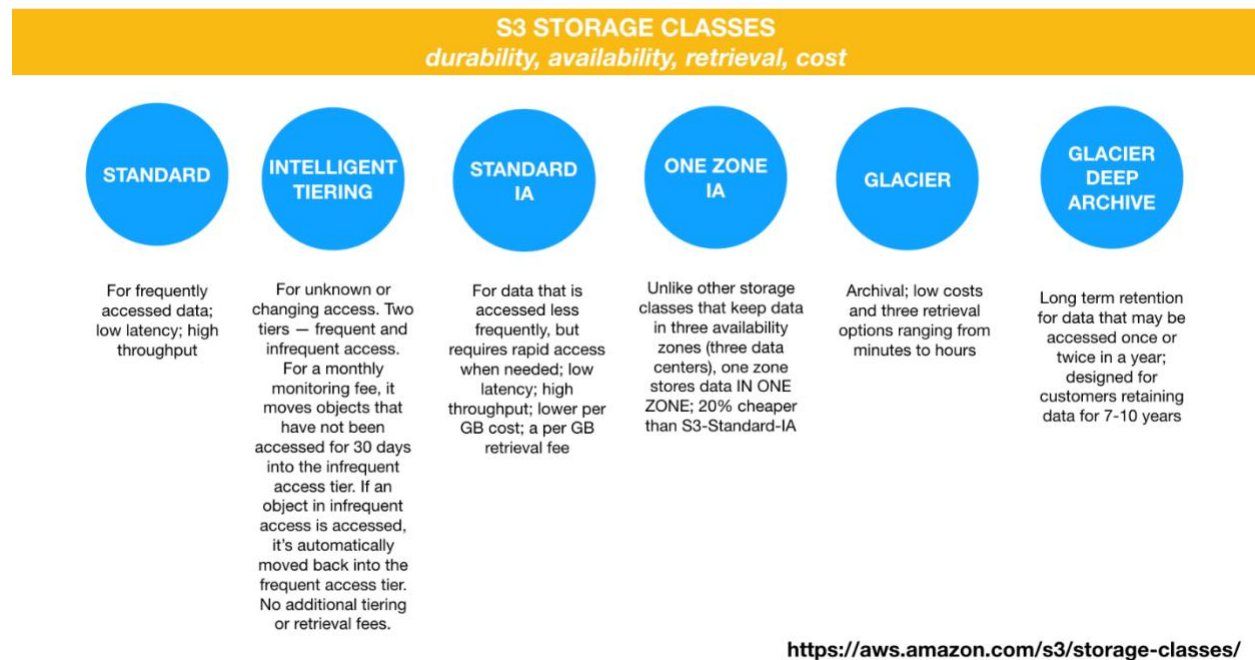
Viewing 1 to 7

Celeste

S3 Lifecycle Management

The S3 buckets in use for genomics analysis utilize S3 Lifecycle configuration rules to manage the transition of objects through different storage classes. The below image describes the different storage classes available in amazon object-store.

(See <https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html>)



To view existing lifecycle management rules in the console, navigate to the “Management” tab under the S3 Management Console. New lifecycle rules for specific prefixes or specific tags can be generated here by clicking “add lifecycle rule” and specifying the desired attributes.

Overview

Properties

Permissions

Management

Access points

Lifecycle

Replication

Analytics

Metrics

Inventory

+ Add lifecycle rule

Edit

Delete

Actions

| Lifecycle rule | Applied to | Actions for current version | Actions for previous version(s) |
|--|--------------------------------------|-----------------------------|---------------------------------|
| <div></div> rule-for-archive-no | Whole bucket , Tags : [Key: hgsc... | Standard-IA | - |
| <div></div> rule-for-archives | Whole bucket , Tags : [Key: hgsc... | Standard-IA / Glacier | - |
| <div></div> rule-for-scratch-nooras | prefix : scratch/nooras | Standard-IA | Permanently Delete |
| <div></div> rule-for-previous-versions | Whole bucket | - | Permanently Delete |

The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

For another fine-grained look at existing lifecycle management rules in a given bucket:

```
$ aws s3api get-bucket-lifecycle-configuration --bucket dev-example-bucket
```

```
{
  "Rules": [
    {
      "ID": "rule-for-archive-no",
      "Filter": {
        "Tag": {
          "Key": "archive",
          "Value": "no"
        }
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        }
      ],
      "NoncurrentVersionExpiration": {
        "NoncurrentDays": 30
      }
    },
    {
      "ID": "rule-for-archive-yes",
      "Filter": {
        "Tag": {
          "Key": "archive",
          "Value": "yes"
        }
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 60,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "NoncurrentVersionExpiration": {
        "NoncurrentDays": 30
      }
    }
  ],
  ...
}
<truncated>
```

Celeste

The above json document, or a similar one, can be used to quickly provision a new lifecycle configuration rule via the command line interface (CLI). To do this, first create a valid json document with lifecycle specifications. Then utilize the command:

```
aws s3api put-bucket-lifecycle-configuration --cli-input-json file://<jsondocument>  
--bucket <bucketname>
```

Celeste

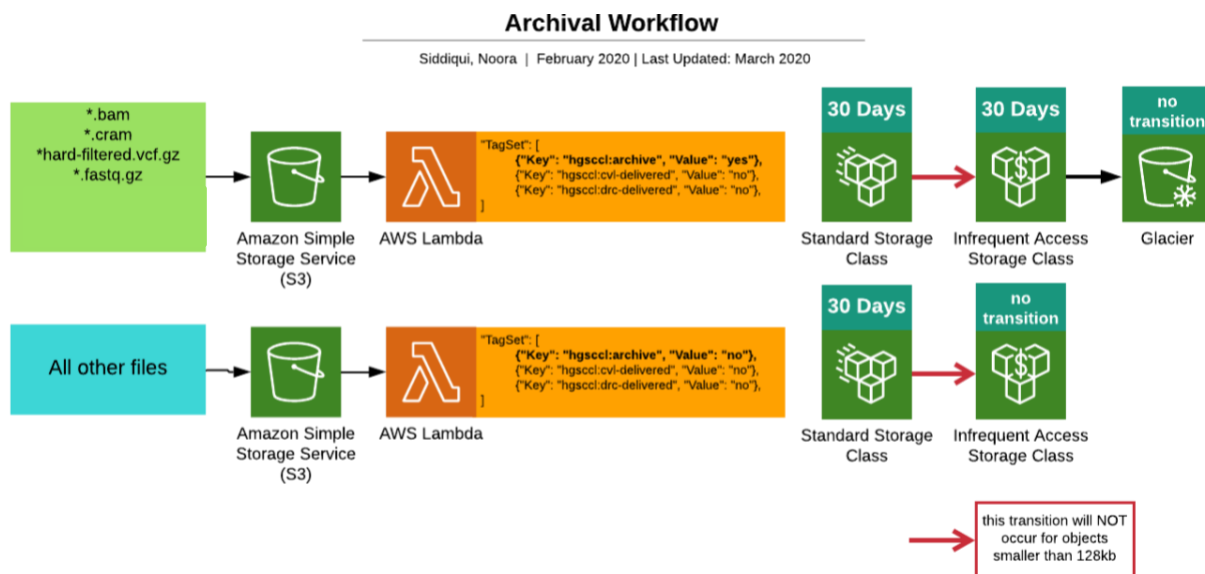
Archival

Understanding Automated Tag-based Archival in the Pipeline

Notice that there are two methods of specifying a lifecycle configuration – one driven by object prefix, and another driven by object tag. *Celeste* utilizes object-level tags to determine traversal through storage classes.

These tags are automatically generated for objects that fall under the S3 prefixes for **clinical-dragen-pipeline** and **38-clinical-dragen-pipeline**. An S3 notification for these two prefixes is sent to AWS Simple Notification Service (SNS), which triggers the **S3TagLambda**.

S3TagLambda assigns object-level tags based on filename in a pattern similar to the following diagram.



The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

Case-Study: Unarchiving data via AWS S3 Batch Operations

Description: When you go to run validation samples for a new clinical project, the DRAGEN jobs fail because they were unable to access the input fastqs. You realize that all of the input fastqs have gone into the Glacier storage class.

It is important to first read through the AWS Documentation governing retrievals: (See <https://docs.aws.amazon.com/amazonglacier/latest/dev/downloading-an-archive-two-steps.html>)

This is a perfect scenario in which to use AWS S3 Batch Operations!

(See <https://docs.aws.amazon.com/AmazonS3/latest/dev/batch-ops-basics.html>)

Here is a relevant and very excellent AWS tutorial on S3 Batch Operations:

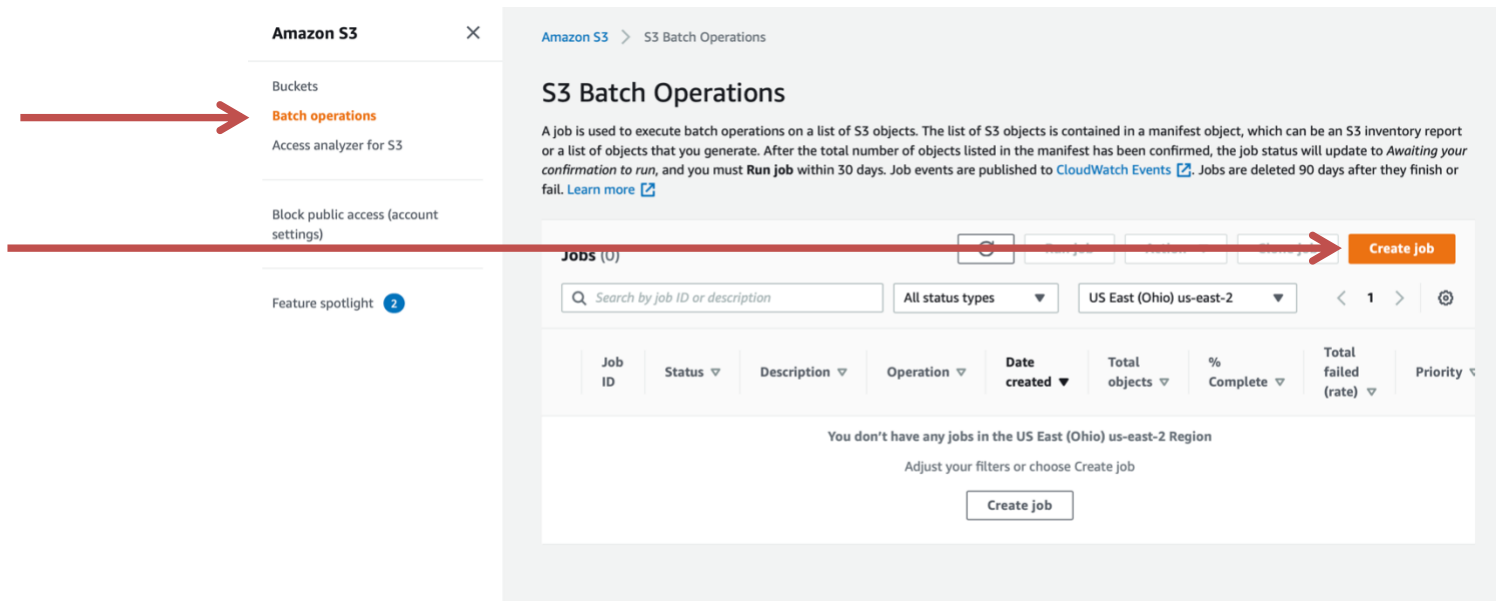
(See <https://aws.amazon.com/blogs/aws/new-amazon-s3-batch-operations/>)

After diligently reading the above material, we understand that we need a “Manifest”, such as a .csv file containing a list of all of the S3 objects to retrieve. Our “job” will be an archive retrieval request which we can submit via awscli or via the management console. We can expect the retrieval to occur within the timeframe specified by the AWS documentation above.

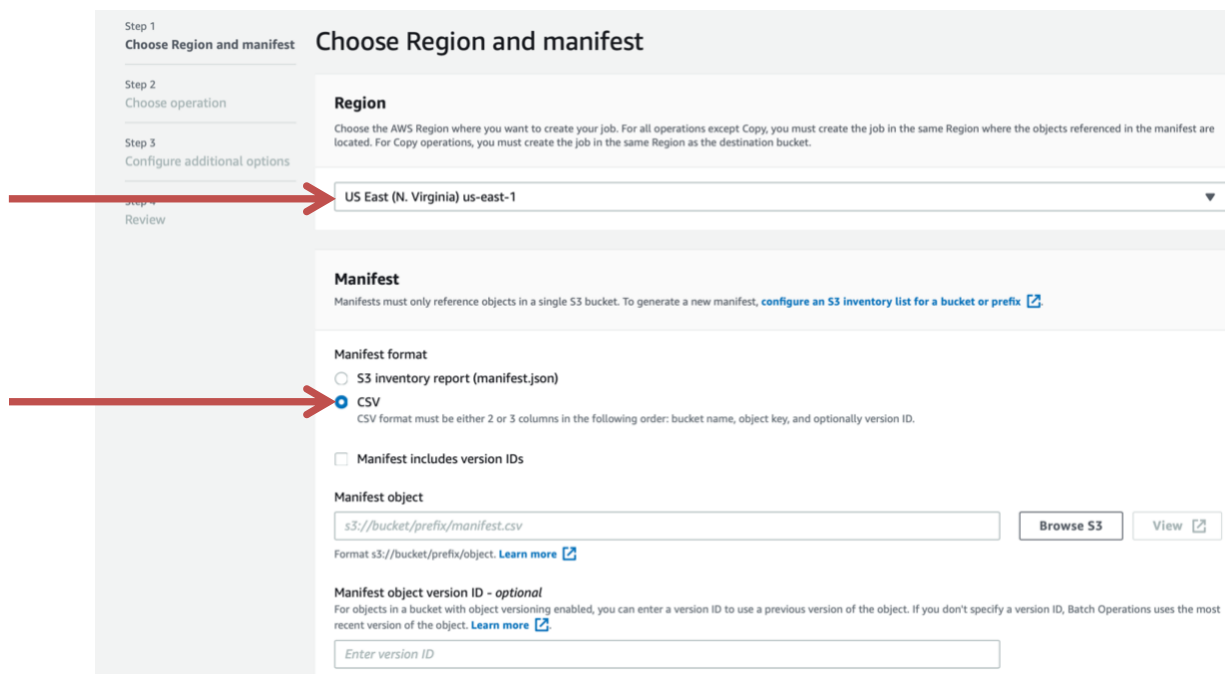
Celeste

Let's run through a few steps in the process together:

1. Navigate to the S3 Management Console and select *"Batch Operations"*. The Select *"Create Job"*



2. Select the region. As of the current date, we work solely in us-east-1, or Virginia. Select the option for *".csv"*



The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

3. Lets discuss the .csv file for a moment. We need it to contain at least two columns. The first should delineate the bucket name, and the second column should contain the full object key. If you need a refresher on object keys:

(See <https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html>)

Here is a snippet of an example manifest.csv:

```
bucket-name,object-key
bucket-name,object2-key
bucket-name,object3-key
dev-example-bucket,clinical-dragen-pipeline/fastqs/some/file/path/to.fastq.gz
```

4. You might be concerned. How do I find all the needed fastqs? Here is a quick look at awscli commands and filters that can guide you to determine your exact object keys.

Intermediate S3 Commands

- a) See keys for items in an S3 bucket are in the GLACIER storage class:

```
aws s3api list-objects --bucket <bucketname> --query 'Contents[?
StorageClass==`GLACIER`][Key]' --output text
```

- b) List specific fastq keys and filter output via bash:

```
aws s3api list-objects --bucket dev-example-bucket --prefix scratch/ --output text | grep
"fastq.gz" | cut -f3
```

- c) Query objects by content size and sum to determine the size of the bucket. Size returned in bytes:

```
aws s3api list-objects --bucket dev-example-bucket --output json --query
"[sum(Contents[].Size), length(Contents[])]"
```

5. Specify the manifest.csv via S3 URI, in the form s3://bucketname/path/to/manifest.csv. The other items do not need specification, but you may read through the AWS Documentation regarding S3 Batch Operations, above, to determine cases in which you might want to use other attributes.
6. Hit “Next” and specify the operation type “Glacier Restore”. The number of days you specify within settings is the number of days a temporary copy of the object will be available for things like GET requests.

Celeste

Step 2
Choose operation

Step 3
Configure additional options

Step 4
Review

Operation

Operation type
Choose the operation that you want to perform on all objects listed in the manifest. [Learn more](#)

- ☐ Copy
Copies each object listed in the manifest to the specified destination.
- ☐ Invoke AWS Lambda function
AWS Lambda is a compute service that lets you run code without provisioning or managing servers.
- ☐ Replace all object tags
Replaces the Amazon S3 object tags of each object listed in the manifest.
- ☐ Replace access control list (ACL)
Replaces the Amazon S3 access control lists (ACLs) for each object that is listed in the manifest.
- ☒ **Glacier restore**
Sends a restore request to Amazon S3 Glacier for each object that is specified in the manifest.

Glacier restore settings

Retrieval fees apply. See [S3 pricing](#)

Number of days that the restored copy is available
The restored copy is automatically deleted after a specified number of days.

14

Operation type

- ☒ **Bulk retrieval**
Typically within 5-12 hours for Glacier and within 48 hours for Glacier Deep Archive.
- ☐ Standard retrieval
Typically within 3-5 hours for Glacier and within 12 hours for Glacier Deep Archive.

Cancel Previous Next

7. Specify the job properties and priority. Also specify a path in S3 for the batch operation report.

Step 1
Choose Region and manifest

Step 2
Choose operation

Step 3
Configure additional options

Step 4
Review

Configure additional options

Additional options

Description
2020-03-26 - Restore
Up to 256 characters

Priority
Higher numbers indicate higher priority. You can modify a job's priority after it's created. [Learn more](#)

10
Positive numbers only.

Completion report

Generate a CSV completion report that lists your target objects, task success or error codes, outputs, and descriptions. Completion reports are encrypted using SSE-S3. [Learn more](#)

☒ Generate completion report

Completion report scope

- ☐ Failed tasks only
- ☒ **All tasks**

Path to completion report destination [Learn more](#)

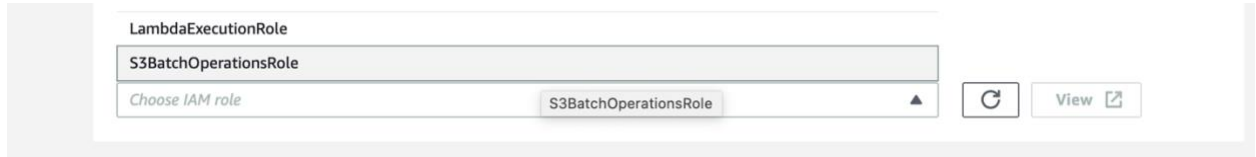
s3://bucket/prefix
Format: s3://mybucket/myprefix. S3 will append the path with a "/". If you add a "/" to the prefix, it will appear as an extra folder in the S3 console.

Browse S3 View

The master copy of this document is maintained by HGSC-CL. Users are responsible for ensuring the latest version of this document is used. Reproduction and/or distribution of this document without proper approval is strictly prohibited.

Celeste

- An IAM role named S3BatchOperationsRoles already exists for your usage with S3 Batch Operations.

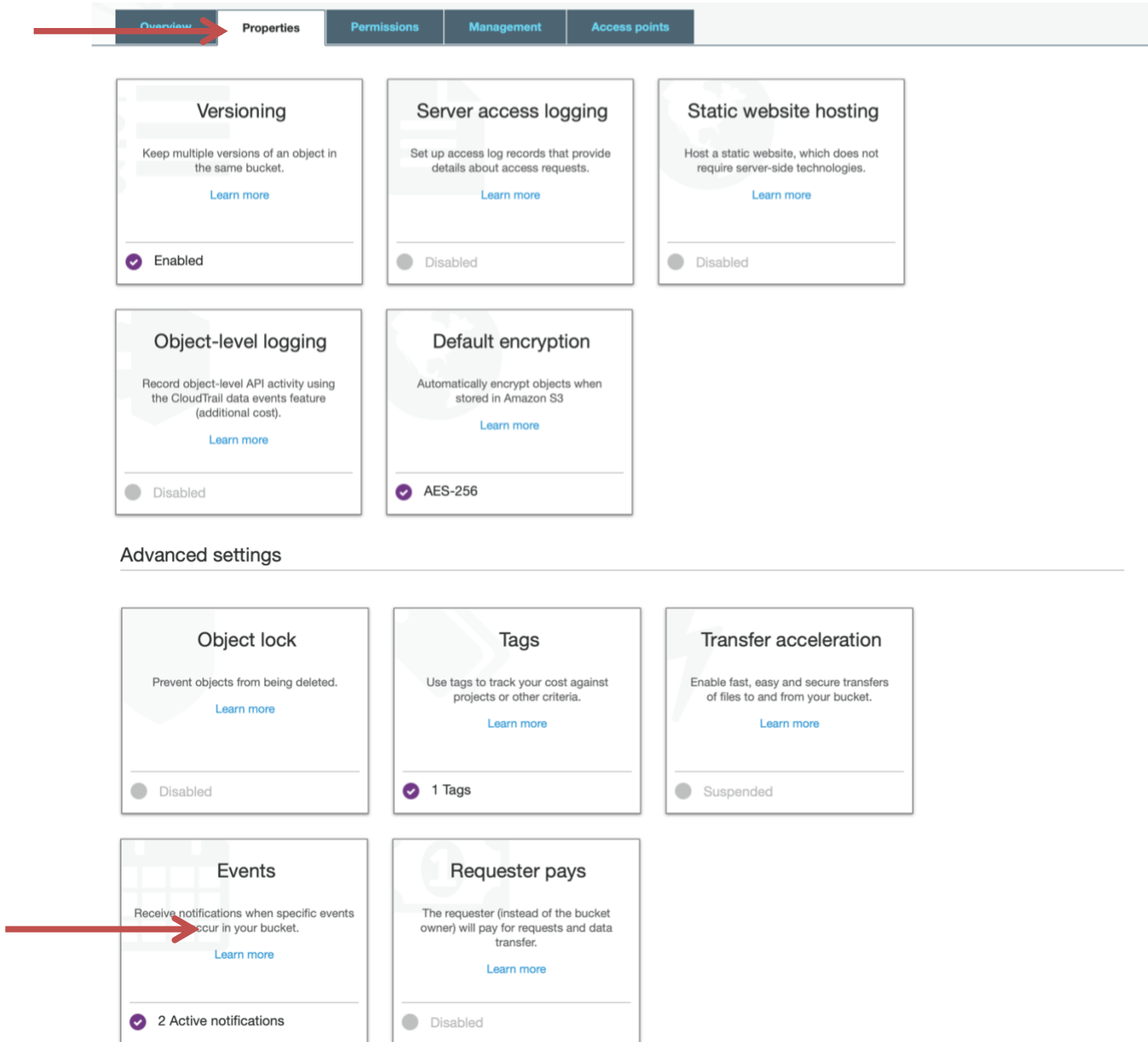


- Hit “Create Job”. The job will enter a preparing state. When it reaches the “Ready” state, you may start the job. It will run and show you completion status as it progresses.

Celeste

S3 Event Notifications

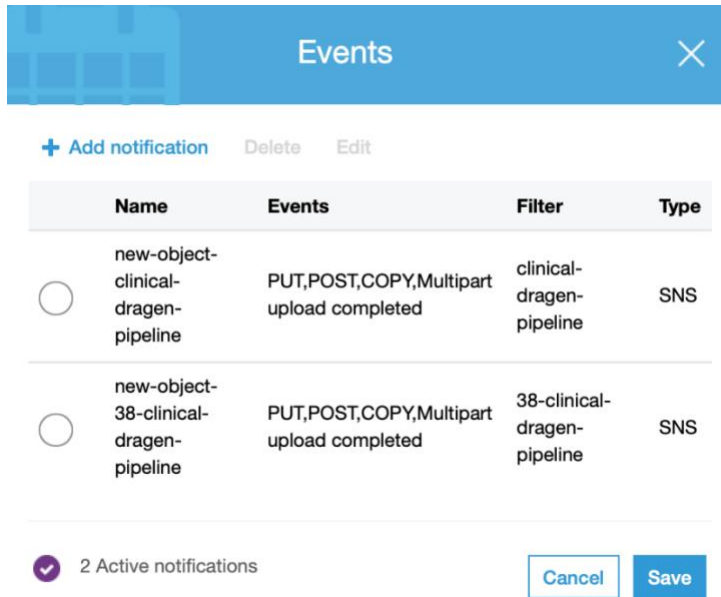
S3 Event Notifications can alert systems when certain actions occur within a bucket. To view existing S3 Event Notifications, navigate to the *Properties* tab from a specific bucket page.



The above image illustrates that the current bucket (s3://dev-example-bucket) has two enabled notifications. The existing notifications will be triggered for any new object that lands under the prefixes for **clinical-dragen-pipeline** and/or **38-clinical-dragen-pipeline**.

Celeste

Referring to Figure 1 in the beginning of the guide, you can verify that these notification events feed into AWS SNS, resulting in a messaging fanout to a collection of serverless functions.



Creating an S3 Event Notification via Console

To create a new notification event for **clinical-dragen-pipeline** in the bucket:

- Navigate to the S3 service page from the AWS Management Console
- Click the bucketname and select *Properties*
- Under *Events* click *Add new notification* and ensure the following are specified

| PARAMETER | VALUE |
|-----------|--|
| Name | new-object-clinical-dragen-pipeline |
| Event | PUT,POST,COPY,Multipart upload completed |
| Prefix | clinical-dragen-pipeline |
| Send To | SNS Topic |
| SNS Topic | NewObjectSNSTopic-<env> |

Celeste

Making a New S3 Bucket

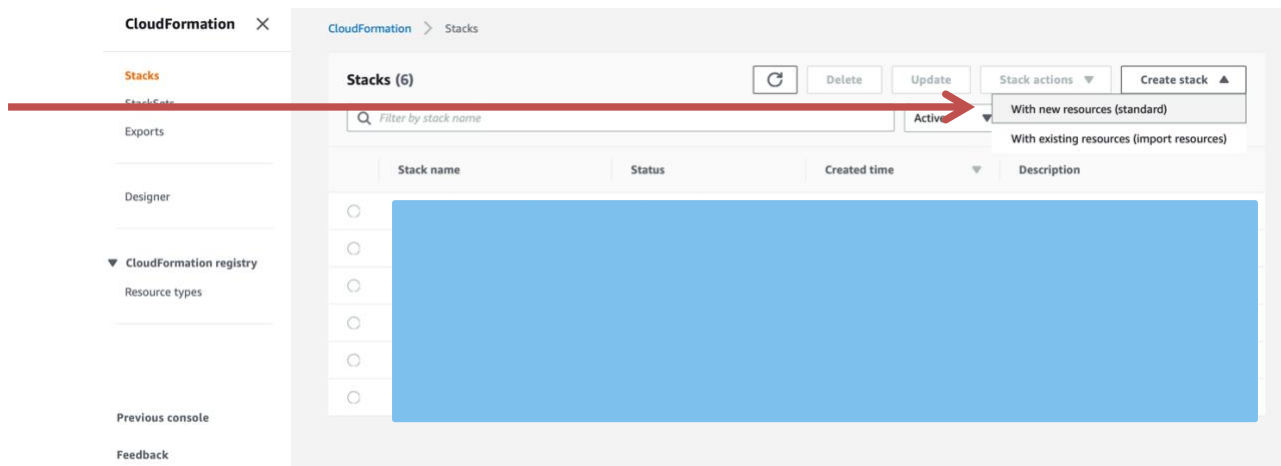
Case-Study: A bucket with lifecycle configuration, encryption, and security specifications

Description: You realize you would like to provision a specific environment, but there is no genomics data bucket set up for it.

This bucket needs to be AES256 encrypted, private (no public access), and have versioning enabled to meet compliance protocols. We will use an existing CloudFormation template to provision the necessary buckets and automate the deployment of these resources in a reproducible manner.

Let's work through the following steps:

2. Navigate to the CloudFormation Management Console and select *Create Stack*
 - a) Select "*with new resources*"



- b) Access the template *templates/s3-config.standalone-template.yaml* from the Celeste git repository.
 - c) Upload this template to a space in S3.
 - d) Enter the https URL for your newly uploaded template (NOT s3uri) in the form: **Error! Hyperlink reference not valid.**
3. Enter *<bucketname>* for *Stack Name*
4. Enter *<bucketname>* for the *S3 Bucket* parameter

Celeste

CloudFormation > Stacks > Create stack

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Specify stack details

Stack name

Stack name

test-bucket-with-a-UNIQUE-NAME-STACK

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Configuration

S3 Bucket

Name of S3 bucket to create.

test-bucket-with-a-UNIQUE-NAME

Cancel Previous Next

5. Hit Next and specify tags under *Configure stack options*.
 - a) For example, add an environment Key:Value tag for the bucket stack, as follows: *Key=env, Value= prod*
6. Hit "Next" and scroll down until you see "Create Stack."
 - a) Hit "Create Stack"!
7. Navigate to the S3 service page from the AWS Management Console to check that the bucket was created. You will see the resulting bucket has versioning enabled, AES256 encryption enabled, and lifecycle configuration rules for archives by clicking through the *Properties* and *Management* tabs.

Celeste

AoU PARAMETER SET – hs37d5

- 1.2 The parameter set described here incorporates AoU Cross Center Equivalency agreements for the analysis of WGS samples with reference build hs37d5.
- 1.3 All resource and reference files can be found in s3://op-data-bucketname/AoU Resources

Workflow (Tool and Version): DRAGEN Specifications

| Site | Sequencer RTA | Demultiplex | Map & Align | Variant Call |
|-------------|---------------|----------------|----------------|----------------|
| BCM HGSC-CL | v3.4.4 | Picard (2.6.0) | DRAGEN v3.4.12 | DRAGEN v3.4.12 |

Deployment/Environment

| Site | Deployment | Hardware Base Version | Software Version |
|-------------|------------|-----------------------|-------------------|
| BCM HGSC-CL | AWS | 0x071417D3 | 05.021.408.3.4.12 |

hs37d5 Merge Event: DRAGEN Map & Align, and Variant Call Command Line

Celeste

```
<dragen>
--force \
--board-count=1
--ref-dir s3://op-data-bucketname/bwa_references/h/hs37d5/hs37d5_hashtable/ \
--fastq-list <fastqlist> \
--fastq-list-sample-id <RGSM> \
--output-file-prefix <[CenterID]_[Biobankid_Sampleid]_[LocalID:optional]_[Rev#]> \
--output-directory <OUTDIR>
--enable-variant-caller true \
--vc-enable-vcf-output true \
--vc-emit-ref-confidence GVCF \
--enable-duplicate-marking true \
--enable-map-align true \
--enable-map-align-output true \
--output-format=BAM \
--vc-hard-filter 'DRAGENHardQUAL:all:QUAL<5.0;LowDepth:all:DP<=1' \
--vc-frd-max-effective-depth=40 \
--qc-cross-cont-vcf s3://op-data-bucketname/AoU-resources/hs37d5-resources/SNP_NCBI_hs37d5.vcf \
--qc-coverage-region-1 s3://op-data-bucketname/AoU-resources/hs37d5-
resources/wgs_coverage_regions.hs37d5_minus_N.interval_list.bed \
--qc-coverage-reports-1 cov_report \
--qc-coverage-region-2 s3://op-data-bucketname/AoU-resources/hs37d5-
resources/acmg59_allofus_19dec2019.bed \
--qc-coverage-reports-2 cov_report \
--qc-coverage-region-3 s3://op-data-bucketname/AoU-resources/hs37d5-
resources/PGx_singleSite_GRCh37_ActualTableToSendToFDA_21jan2020.annotated.bed \
--qc-coverage-reports-3 cov_report
--lic-server XXXYYY
```

hs37d5 Single Sequencing Event: DRAGEN Map & Align Command Line

Celeste

```
<dragen>
--force \
--board-count=1
--ref-dir s3://op-data-bucketname/bwa_references/h/hs37d5/hs37d5_hashtable/ \
--fastq-list <fastqlist> \
--fastq-list-sample-id <RGSM> \
--output-file-prefix <[CenterID]_[Biobankid_Sampleid]_[LocalID:optional]_[Rev#]> \
--output-directory <OUTDIR>
--enable-duplicate-marking true \
--enable-map-align true \
--enable-map-align-output true \
--output-format=BAM \
--qc-cross-cont-vcf s3://op-data-bucketname/AoU-resources/hs37d5-resources/SNP_NCBI_hs37d5.vcf \
--qc-coverage-region-1 s3://op-data-bucketname/AoU-resources/hs37d5-
resources/wgs_coverage_regions.hs37d5.interval_list.bed \
--qc-coverage-reports-1 cov_report \
--qc-coverage-region-2 s3://op-data-bucketname/AoU-resources/hs37d5-
resources/acmg59_allofus_19dec2019.bed \
--qc-coverage-reports-2 cov_report \
--qc-coverage-region-3 s3://op-data-bucketname/AoU-resources/hs37d5-
resources/PGx_singleSite_GRCh37_ActualTableToSendToFDA_21jan2020.annotated.bed \
--qc-coverage-reports-3 cov_report
--lic-server XXXYYY
```

Workflow (Tool and Version): AlignStats Specifications

| Site | Reporting Tool and Version | Other Tool Versions |
|-------------|----------------------------|----------------------------|
| BCM HGSC-CL | AlignStats 0.9.1 | HTSlib 1.9 Samtools 1.9 |

If BAM:

```
<alignstats0.9.1>
-v -i inputBAM -m s3://op-data-bucketname/alignstats/masks/hs37d5_N_regions_autosome.bed \
-o out -C -r s3://op-data-bucketname/alignstats/regions/hs37d5_autosome.bed -P 1 -F -2048 -b 3 -O
```

If CRAM:

```
ADD:
-T s3://op-data-bucketname/bwa_references/h/hs37d5/hs37d5.fa
```

Celeste

Workflow (Tool and Version): VerifyBamID Specifications

| Site | Reporting Tool and Version | Other Tool Versions |
|-------------|----------------------------|----------------------------|
| BCM HGSC-CL | VerifyBamID 1.1.3 | HTSlib 1.9 Samtools 1.9 |

If BAM:

```
<verifyBamID1.1.3>  
--bam inputBAM --vcf s3://op-data-bucketname/verifyBamID/hapmap_3.3.b37.sites.vcf.gz \  
--out out --verbose --ignoreRG --noPhoneHome
```

If CRAM:

No functionality with VerifyBamID 1.1.3

Celeste

AoU PARAMETER SET – GRCh38

The parameter set described here incorporates AoU Cross Center Equivalency agreements for the analysis of WGS samples with reference build GRCh38.

All resource and reference files can be found in `s3://op-data-bucketname/AoU-Resources/`

GRCh38 DRAGEN Specifications

MERGE EVENT: DRAGEN Map & Align, and Variant Call Command Line

```
<dragen>
--force
--board-count=1
--ref-dir s3://op-data-bucketname/bwa_references/h/grch38/grch38_hashtable/ \
--fastq-list <path-to>/fastq_list.csv \
--fastq-list-sample-id <RGSM> \
--output-directory <output-dir> \
--output-file-prefix 38-clinical-dragen-pipeline/[CenterID]_[Biobankid_Sampleid]_[LocalID:optional]_[Rev#] \
--enable-variant-caller true \
--vc-enable-vcf-output true \
--vc-emit-ref-confidence GVCF \
--enable-duplicate-marking true \
--enable-map-align true \
--enable-map-align-output true \
--output-format=CRAM \
--vc-hard-filter 'DRAGENHardQUAL:all:QUAL<5.0;LowDepth:all:DP<=1' \
--vc-frd-max-effective-depth=40 \
--qc-cross-cont-vcf s3://op-data-bucketname/AoU-resources/hg38-resources/SNP_NCBI_GRCh38.vcf \
--qc-coverage-region-1 s3://op-data-bucketname/AoU-resources/hg38-
resources/wgs_coverage_regions.hg38_minus_N.interval_list.bed \
--qc-coverage-reports-1 cov_report \
--qc-coverage-region-2 s3://op-data-bucketname/AoU-resources/hg38-
resources/acmg59_allofus_19dec2019.GRC38.wGenes.bed \
--qc-coverage-reports-2 cov_report
--qc-coverage-region-3 s3://op-data-bucketname/AoU-resources/hg38-
resources/PGx_singleSite_GRCh38_ActualTableToSendToFDA_21jan2020.bed \
--qc-coverage-reports-3 cov_report
--lic-server XXXYYY
```

Celeste

GRCh38: AlignStats Specifications

If BAM:

```
<alignstats0.9.1>
-v -i inputBAM -m s3://op-data-bucketname/alignstats/masks/GRCh38_1000Genomes_N_regions.bed \
-o out -C -r s3://op-data-bucketname/alignstats/regions/GRCh38_full_analysis_set_plus_decoy_hla.bed -P 1
\F -2048 -b 3 -O
```

If CRAM:

```
ADD:
-T s3://op-data-bucketname/bwa_references/h/grch38/GRCh38_full_analysis_set_plus_decoy_hla.fa
```

Celeste

GRCh38: Intersect Specifications

| Site | Reporting Tool and Version | Other Tool Versions |
|-------------|----------------------------|---------------------|
| BCM HGSC-CL | BedTools 2.29.1 | Python3.7 |

BAM/CRAM:

```
<bedtools2.29.1>  
-b s3://op-data-bucketname/Liftover_resources/Bed_file/ACMG59_PGx.combined.grc38.annotated.bed \  
-a DragenOutput.hard-filtered.vcf.gz -o output
```

GRCh38: Preprocessing Specifications

| Site | Reporting Tool and Version | Other Tool Versions |
|-------------|----------------------------|---------------------|
| BCM HGSC-CL | Liftover_helper 1.0 | Python3.7 |

BAM/CRAM:

```
<liftover_helper2.29.1> Intersected_hard-filtered.vcf Output_folder
```

GRCh38: Preprocessing Specifications

| Site | Reporting Tool and Version | Other Tool Versions |
|-------------|----------------------------|---------------------|
| BCM HGSC-CL | Liftover_helper 1.0 | Python3.7 |

BAM/CRAM:

```
<liftover_helper2.29.1> Intersected_hard-filtered.vcf Output_folder
```

Celeste

GRCh38: Liftover Specifications

| Site | Reporting Tool and Version | Other Tool Versions |
|-------------|----------------------------|---------------------|
| BCM HGSC-CL | Picard 2.23.3 | Python3.7 |

BAM/CRAM:

```
<Picard2.23.3> -i Preprocessed_Intersect_hard.filtered.vcf -o Output_folder -c s3://op-data-bucketname/Liftover_resources/Main/hg38ToHg19.over.translate.chain -r s3://op-data-bucketname/bwa_references/h/hs37d5/hs37d5.fa
```

GRCh38: Stargazer Specifications

| Site | Reporting Tool and Version | Other Tool Versions |
|-------------|----------------------------|---------------------|
| BCM HGSC-CL | Stargazer 1.0.9 | Python3.7 |

BAM/CRAM:

```
<Stargazer1.0.9> --vcf Liftover_Preprocessed_Intersect_hard.filtered.vcf -t all -c --data wgs \  
--output_dir Output_folder -o Stargazer_Liftover_Preprocessed_Intersect_hard.filtered.vcf
```