WRANGLING THE
DRAGEN IN THE CLOUD

A BCM-HGSC In-House Guide to Using Illumina's DRAGEN Technology with Amazon Web Services (AWS)*

*https://www.illumina.com/products/by-type/informatics-products/dragen-bio-it-platform.html

Author: Noora Siddiqui, nooras@bcm.edu

PART I

A simple introduction to running a single WES or WGS dragen job.
Includes manually starting and utilizing a single on demand instance.

**Note: This tutorial is aimed for IAM users within the BCM-HGSC AWS organization.*
Bucket assets are not public but instructions may be generalized
to be applicable to individuals outside the organization.

We will begin with a canned example

e.g. Taking advantage of a preconfigured EC2 instance to successfully run Dragen. This tutorial assumes basic knowledge of AWS terminology and an existing IAM user account

1. Configure the AWS Command Line Interface (CLI):

[1] → aws configure

AWS Access Key ID: *ask admin
AWS Secret Access Key: *ask admin
Default region name: **us-east-1**
Default output format: (options: **text**, **json**, or **table**)

The default region name should not be changed from **us-east-1**. We currently take advantage of F1 instances (FPGA-enabled) hosted in N. Virginia. Please keep in mind that all of the IAM users work on the same AWS Marketplace account.

Now we learn about our instance(s)

2. The following will display detailed information in .json format regarding instances managed by our group:

[2] → aws ec2 describe-instances

We currently have several instances configured for use. You should see output related to all of them upon using the above command. Note the image IDs and instance types. The f1.4 and f1.2 instances are specific to running Dragen.

Note: You may launch your own f1.2 instance with a personal key pair (e.g. yourname-KeyPair). Ask me for help if you are unsure what you are authorized to use for the rest of the tutorial.

TIP! Similarly worded commands can be used to describe a variety of ec2 services (e.g. **aws ec2 describe-vpcs**, **aws ec2 describe-key-pairs**, **aws ec2 describe-images**)

Let’s manage our instance

Look at the current state of some of the instances described in the previous slide. Are they running, pending, or stopped?

3. To start an instance:

[3] → aws ec2 start-instances --instance-ids <id of 1 or more instances>

4. To stop an instance:

[4] → aws ec2 stop-instances --instance-ids <id of 1 or more instances>

Start the instance created via ami-3.3.12. Try describing the instance again. You should see the previously “*stopped*” state turn to “*pending*” before eventually “*running*.” The instance is *stopped* when not in use to prevent running costs during preliminary testing. However, it is important to understand what “*stopping*” actually means.

What happens when I stop my instance?

AWS supports the ability to stop OR terminate an instance launched using an EBS-based AMI. There are distinct differences between stopping and terminating.

When you **STOP** an instance, the instance is shut-down, the virtual machine that was provisioned will be permanently taken away, and you will not be charged for instance usage.

- Data attached in any EBS volumes will not be deleted.
- Data mounted in the /ephemeral hard drive will be lost.
- When you restart a stopped instance, the EBS volume is simply attached to the newly provisioned virtual machine

When you **TERMINATE** an instance, the instance is shut-down, the virtual machine that was provisioned will be permanently taken away, and you will not be charged for instance usage.

- Any attached EBS volumes will be **detached and deleted**.
- Any data that was stored locally on the instance will be lost.

Time to log in!

5. First obtain the public IP using the following command:

[5] → aws ec2 describe-instances --instance-ids <your instance> **| grep PublicIpAddress**

Note: restarting an instance results in a new IP address, DNS names, etc.

6/7. Now use the public IP in conjunction with a private key to log in. To ssh into the instance, you need access to the specific private key used during launch-time. To successfully log-into the instance, permissions for yourname-KeyPair.pem *must be set correctly.*

[6] → chmod 400 path/to/yourname-KeyPair.pem

[7] → ssh -i path/to/yourname-KeyPair.pem centos@<publicIP>

Let’s look at what is in S3

For our canned example, we will use some fastqs currently stored in an S3 bucket specifically created for Dragen evaluation

The existing S3 bucket, named “drageneval,” contains the Dragen proprietary reference hashtable, many fastqs, and sample-specific objects. Your IAM user can write to the space “*s3://drageneval/user/yourname*"

8. In AWS S3, “buckets" and “objects” are the primary resources, with objects stored within buckets. **Unlike a file system**, S3 has a flat structure. See what’s in your space in the bucket using the following command:

[8] → aws s3 ls s3://drageneval/user/yourname/

9. The best “S3 equivalent” I could find for the UNIX “tree” command:

[9] → aws s3 ls --summarize --human-readable --recursive s3://drageneval/

Practice navigating S3

Here are some example S3 commands you might be interested in using since they differ a bit from similar Linux/Unix commands

10. 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:

[10] → aws s3api put-object --bucket <bucketname> --key <dirnameBUTNOTADIR/>

11. To copy only fastqs to S3 from a directory of mixed file-types:

[11] → aws s3 cp <your directory path> s3://<your bucket name>/ --exclude "*" --include "*.fastq.gz"

12. 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:

[12] → aws s3 sync <your local path> s3://<your bucket name>/

13. Sync data from S3 storage to your storage space. You can use filters for **--exclude --include** and **—dryrun,** among a bevy of other options.

[13] → aws s3 sync s3://<your bucket name>/ <your local path>

Now, the Dragen

The phrase “wrangling the Dragen in the cloud,” implies
the **beast** is the Dragen.
False.
The beast is the **cloud**. Here’s the fun part:

14. Now that you are in the running instance, we’ll run Dragen. I will give commands that begin *ab ovo*, along with a “Martha Stewart pre-baked cake” that you can pull out of the figurative oven, instead.

[14] → cd /ephemeral

15/16. We will use a data flow and storage configuration as follows:

- The hash table is stored in /ephemeral
- Input fastqs are streamed directly from S3
- No additional EBS volumes are attached
- Output is streamed directly to S3

[15] → mkdir hs37d5

[16] → mkdir tmp

Reference hashtable

17. Copy hs37d5.fa to /ephemeral/hs37d5/ and build the reference hashtable:

    [17] → /opt/edico/bin/dragen --ht-reference /ephemeral/hs37d5/ hs37d5.fa --output-directory /ephemeral/hs37d5/ --build-hash-table true

    OR pre-baked cake:

    [17] → aws s3 sync /ephemeral/hs37d5 s3://drageneval/hs37d5_hashtable

Our Goal: Run Dragen on a WES Sample

From the manual: “DRAGEN performs decompression, mapping, aligning, sorting, and optional duplicate marking and feeds directly into the variant caller to produce a VCF file. In this mode, DRAGEN uses parallel stages throughout the pipeline to drastically reduce the overall run time.”

Full pipeline example statement

18. This is the general usage statement for a plain-vanilla, full pipeline Dragen run — paired fastq to BAM and VCF.

```
[18] → /opt/edico/bin/dragen \
 --output-directory s3://<PATH/TO/OUTPUT> \
 --fastq-file1 <FASTQ_R1> \
 --fastq-file2 <FASTQ_R2> \
 --output-file-prefix <OUT_PREFIX> \
 --ref-dir <HASHTABLE_DIR> \
--output-format BAM \
--enable-variant-caller true --enable-map-align true \
--vc-sample-name <SAMPLE_NAME> \
--enable-map-align-output true \
--enable-duplicate-marking true \
--intermediate-results-dir <TMP> \
--lic-server XXX:YYY@license.edicogenome.com \
  |& tee <LOG_FILE_NAME>   <— capture stdout to a log file (local to the instance), for your
 reference
```

A specific WES example

19. This should take less than 10 minutes to run. Refer to the S3 slides and feel free to make an S3 object for your test run. Alternatively, you could just store the output directly on the instance for your test. This is the general usage statement for a plain-vanilla, full pipeline Dragen run of a NA12878 WES sample already in S3.

```
[19] → /opt/edico/bin/dragen \
 --output-directory s3://drageneval/user/yourname/dragenWEStest/ OR /ephemeral/someplace/you/
specify \
 --fastq-file1 s3://drageneval/testWES/HFHMCDSXX-1-IDUDI0075_S52_L001_R1_001.fastq.gz \
 --fastq-file2 s3://drageneval/testWES/HFHMCDSXX-1-IDUDI0075_S52_L001_R2_001.fastq.gz \
--output-file-prefix HFHMCDSXX-1-IDUDI0075 \
--ref-dir /ephemeral/hs37d5/ \
--output-format BAM \
--enable-variant-caller true --enable-map-align true \
--vc-sample-name HFHMCDSXX-1-IDUDI0075 \
--enable-map-align-output true \
--enable-duplicate-marking true \
--intermediate-results-dir /ephemeral/tmp \
--lic-server XXX:YYY@license.edicogenome.com \ <— ask admin if you need this info
  |& tee dragen_log_HFHMCDSXX-1-IDUDI0075.txt
```

Looking into Output

Congrats on your first Dragen run! Let's see what we have…

You should see the output BAM, VCFs (multiallelic), VCF indexes, md5sums, logs, and metrics. Take a look at the mapping_metrics.csv, time_metrics.csv, and vc_metrics.csv. These are enabled by default. You can specify additional metrics, if desired.

> **See:** *dragen-bio-it-platform-user-guide-v3-3-5.pdf* page 65.

Dragen supports a lot of different variant call filtering methods. The default hard filter thresholds on a QUAL value.

> **See:** *dragen-bio-it-platform-user-guide-v3-3-5.pdf* page 123 for options, as well as page 26.

What if I have multiple fastqs?

In the presence of multiple fastq input files for either single or paired-end data, Dragen can handle a csv containing a list of fastqs. This saves A LOT of time when compared to fastq concatenation. For example, concatenating forward and reverse fastqs might take several hours before a WGS Dragen run — whereas a WGS Dragen run given a list of fastqs might take an hour on an f1.4x instance, plus change.

20. The fastq-list CSV file must have the following columns:

RGID,RGSM,RGLB,Lane,Read1File,Read2File

RGID = Read Group ID
RGSM = Read Group Sample Name
RGLB = Read Group Library
Lane = Flowcell Lane
Read1File & Read2File = Full paths to valid fastqs

The above data can be easily found in BAM files using the following command:

[20] → samtools view -H <BAM_FILE> | grep '@RG'

If this metadata is unavailable, RGID can be extrapolated from Platform Unit (PU), of the form flowcell-barcode.lane or from the in-house method, flowcell-lane-barcode.

RGID for Merge_H577CDSXX-1-IDUDI0001.fastq.gz would be H577C.1

Dragen with fastq CSV list

Add the following tags to run Dragen with the CSV list option:

**--fastq-list <csv_file> **
**--fastq-list-sample-id <Sample ID> **

Note that you can use one csv file to list out all of your fastq paths for multiple samples. Dragen will reference the correct fastqs based on sample ID (RGSM).

Try a WGS example with fastq list

A list of relevant fastqs:

s3://drageneval/testWGS/H577CDSXX-1-IDUDI0001_S1_L001_R1_001.fastq.gz
s3://drageneval/testWGS/H577CDSXX-1-IDUDI0001_S1_L001_R2_001.fastq.gz
s3://drageneval/testWGS/H577CDSXX-2-IDUDI0001_S25_L002_R1_001.fastq.gz
s3://drageneval/testWGS/H577CDSXX-2-IDUDI0001_S25_L002_R2_001.fastq.gz
s3://drageneval/testWGS/H577CDSXX-3-IDUDI0001_S49_L003_R1_001.fastq.gz
s3://drageneval/testWGS/H577CDSXX-3-IDUDI0001_S49_L003_R2_001.fastq.gz
s3://drageneval/testWGS/H577CDSXX-4-IDUDI0001_S73_L004_R1_001.fastq.gz
s3://drageneval/testWGS/H577CDSXX-4-IDUDI0001_S73_L004_R2_001.fastq.gz

A fastq csv list and example output (the pre-baked cake) are all in the S3 drageneval bucket. You will find this material available for sample H577CDSXX-IDUDI0001 if you peruse through the objects. Talk to me if you require access.

The WGS Dragen run should take ~1-1.5 hours total on an f1.4x instance and ~2 hours on an f1.2x instance. You can run on a detached screen session.

As you percolate on what you've learned…

This Quick Guide is meant as a simple, plain-vanilla introduction to AWS and Dragen. For a more comprehensive look into certain aspects of Dragen, I recommend the most up-to-date version of the "Illumina DRAGEN Bio-IT Platform User Guide."

. . .

Lastly, I spent days to understand AWS/Dragen so that others could spend less than a minute. If anything in this guide happens to take you over a minute to figure out, just ask me. :)

Part I, Created May 2019, Last updated Oct 2019
Contact: Noora Siddiqui, nooras@bcm.edu

PART II

Running dragen jobs using AWS Batch
and an amalgam of pre-configured AWS resources.

Before we begin:

Source virtual environment and configure AWS command line interface (CLI)

[1] → source /hgsccl/next-gen/scratch/nooras/.pythonuserbase/virtualenvs/awsenv/.venv/bin/activate

Check that the venv works as expected. You should see no errors with the following command:

[2] → python3
>> import boto3
>> exit()

[3] → aws configure

AWS Access Key ID:
AWS Secret Access Key:
Default region name: **us-east-1**
Default output format: (options: **text**, **json**, or **table**; **just hit the RETURN key**)

The default region name should not be changed from **us-east-1**. We currently take advantage of F1 instances (FPGA-enabled) hosted in N. Virginia. Please keep in mind that all of the IAM users work on the same AWS Marketplace account.

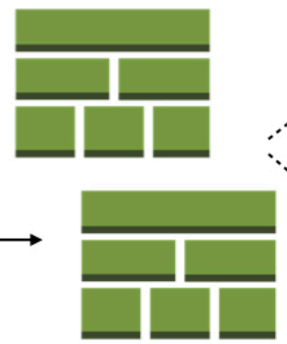
We will utilize an amalgam of AWS resources

Let's run through the one or two of the key pieces in the console real quick

AWS CloudFormation

Easy way to "template" the creation of related AWS resources and predictably provision them.

Can also manually create and provision resources. In fact, it's good to know how to create individual pieces and define IAM roles stringently before finally developing a best-practices template for future automation of the same.



Resource Stack

Includes the Virtual Private Cloud (VPC), its associated subnets, and the compute environments specifying instance types and billing constructs

AWS Batch

"plans, schedules, and executes your **batch** computing workloads"

Flavors of Job submission:

1. From CLI using job.json
2. Via boto3, an AWS SDK for Python <— Also useful for workflow orchestration independent of AWS services

BATCH LAYER*

Where data processing occurs

AWS ECR

"a managed AWS Docker registry service"

AWS S3

Storage service; simply, the easiest piece to understand here

JOB LAYER*

A reservoir for 1. definitions supporting each analytical application and 2. the input and transformed data

AWS StepFunctions

AWS Lambda

AWS CloudWatch

WORKFLOW LAYER*

State-monitoring machinery & orchestration to manage job dependencies & trigger job submission.

*Layer concept derived from Aaron Friedman and Andy Katz's "Building High-Throughput Genomics Batch Workflows on AWS: Introduction"

NS

AWS/Illumina Quickstart

Our tutorial utilizes pieces from:

s3://aws-quickstart/quickstart-illumina-dragen

From AWS: “Quick Starts are built by AWS solutions architects and partners to help you deploy popular technologies on AWS, based on AWS best practices for security and high availability. These accelerators reduce hundreds of manual procedures into just a few steps, so you can build your production environment quickly and start using it immediately.”

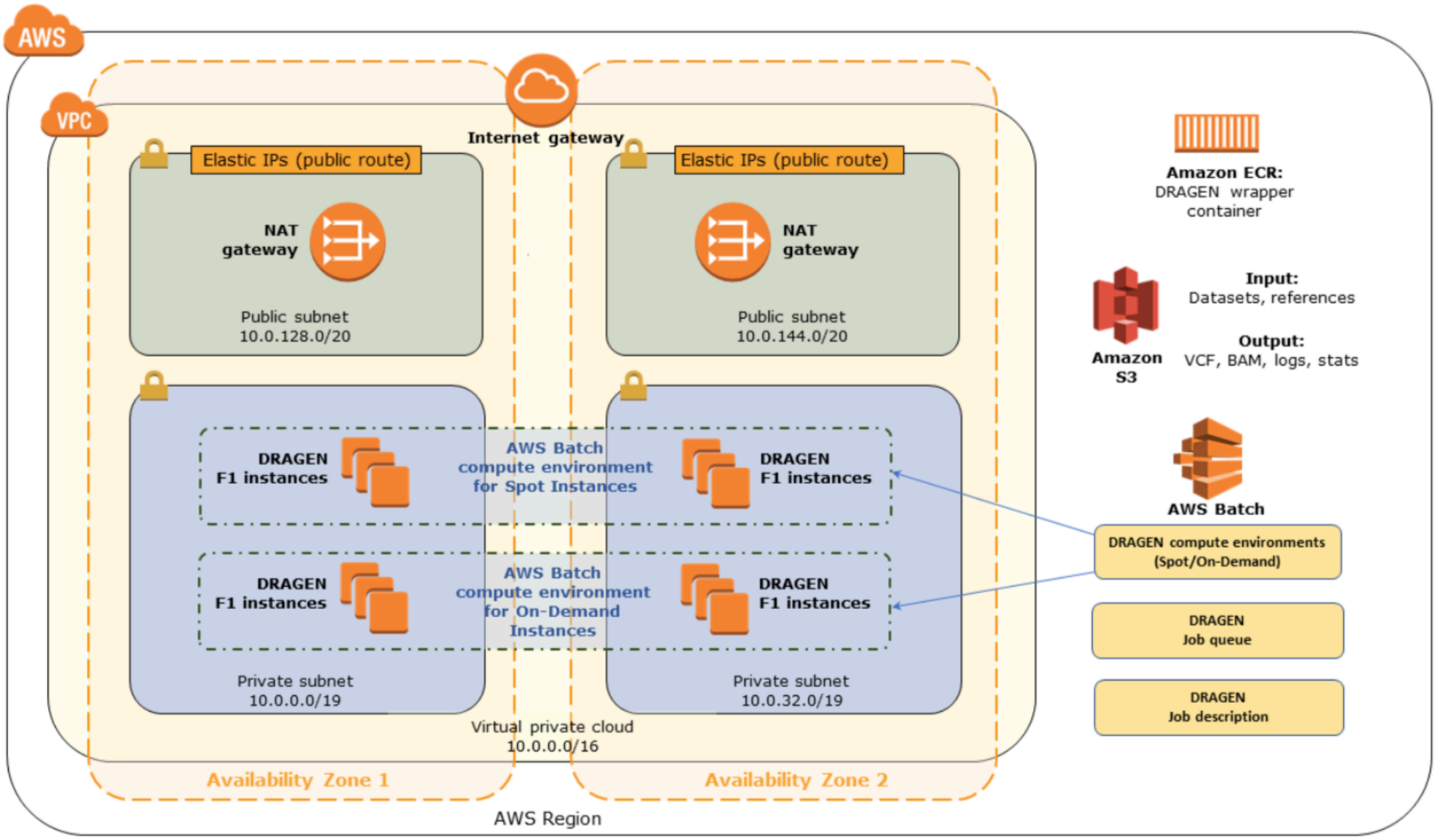


Figure 1: Quick Start architecture for DRAGEN on AWS

https://aws-quickstart.s3.amazonaws.com/quickstart-illumina-dragen/doc/dragen-on-the-aws-cloud.pdf

Let’s look at what is in S3

For our canned example, we will use some fastqs currently stored in an S3 bucket specifically created for Dragen evaluation

The existing S3 bucket, named “drageneval,” contains the Dragen proprietary reference hashtable, many fastqs, and sample-specific objects. Your IAM user has access to the space “*s3://drageneval/user/yourname*"

8. In AWS S3, “buckets" and “objects” are the primary resources, with objects stored within buckets. **Unlike a file system**, S3 has a flat structure. See what’s in your space in the bucket using the following command:

[8] → aws s3 ls s3://drageneval/user/yourname/

9. The best “S3 equivalent” I could find for the UNIX “tree” command:

[9] → aws s3 ls --summarize --human-readable --recursive s3://drageneval/

Practice navigating S3

Here are some example S3 commands you might be interested in using since they differ a bit from similar Linux/Unix commands

10. 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:

[10] → aws s3api put-object --bucket <bucketname> --key <dirnameBUTNOTADIR/>

11. To copy only fastqs to S3 from a directory of mixed file-types:

[11] → aws s3 cp <your directory path> s3://<your bucket name>/ --exclude "*" --include "*.fastq.gz"

12. 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:

[12] → aws s3 sync <your local path> s3://<your bucket name>/

13. Sync data from S3 storage to your storage space. You can use filters for **--exclude --include** and **—dryrun,** among a bevy of other options.

[13] → aws s3 sync s3://<your bucket name>/ <your local path>

Components of AWS Batch

Job

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

Job Queue

Synonymous with the idea of 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.

Compute Environment

The compute resources (instance type, billing construct, IAM roles) associated with a queue. I'm using managed environments so AWS Batch handles scaling.

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.

DRAGEN Job Definition

Use the following command:

[9] → 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 just an artifact of testing combined with the (terrible) fact that job definitions cannot be deleted — only *deactivated*.

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).

DRAGEN Compute Environments

[10] → aws batch describe-compute-environments --output table

dragen-spot, dragen-on-demand

The above are two of the compute environments currently associated with **dragen-queue.** They 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: all of these compute environments specify the use of f1.2xlarge instances since f1.4xlarges are newer and currently unavailable for use with batch. **This was changed in Aug 2019, so f1.4xlarge are now available.

See the specs: <https://aws.amazon.com/ec2/instance-types/f1/>

Dragen currently uses only 1 FPGA under-the-hood, the additional speed-up on f1.4x or f1.16x is due to the additional vCPUs.

Submit your first Batch job!

[11] → aws batch help

[12] → aws batch submit-job help

Obtain the following .json for use in job submission:

s3://drageneval/test/batchWEStest.json

We'll use this to avoid —putting —a —ridiculous —amount —of —tags —and —commands —during —job —submission. Look at the .json and familiarize yourself with the dragen commands.

If you ever wanted to generate a job submission input json yourself, you could use the following, which includes container overrides:

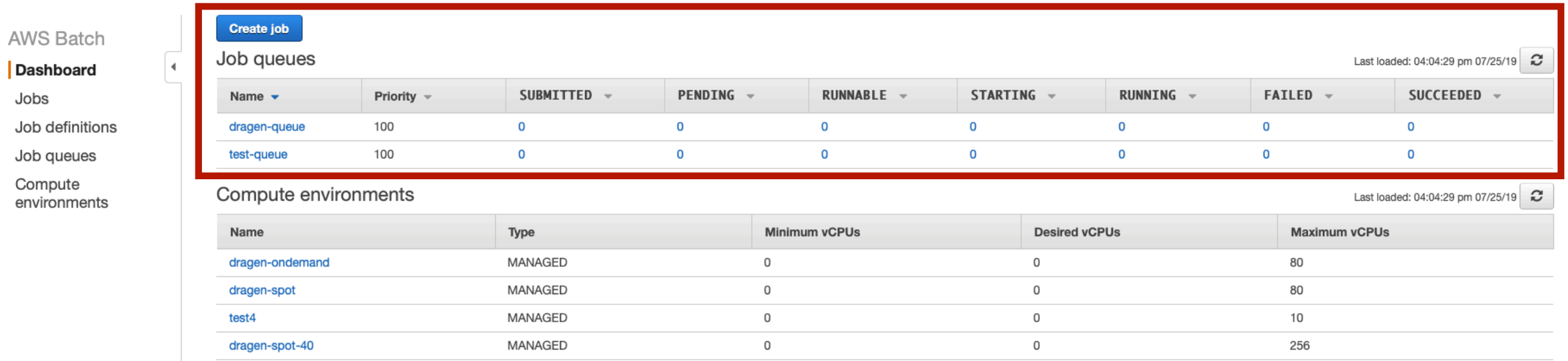
[13] → aws batch submit-job --generate-cli-skeleton

Let's actually submit our WES test to batch!

[14] → aws batch submit-job --cli-input-json file://<PATH>

What's Happening?

Check in the Batch Dashboard:



Or…

[13] → aws batch describe-jobs --jobs <JOBID 2ndJOBID>

An Aside About Boto3

There are two ways to submit jobs:

1. *via AWSCLI: aws batch submit-job (useful for one or two jobs)*
2. *via Python's boto3 library (useful for scale up)*

***I have resources for this elsewhere.**
You can ask to see the tutorial directory in my scratch space*

AWS/Illumina Quickstart

Take a look at how the stack of resources was set-up:

[11] → aws s3 ls s3://aws-quickstart/quickstart-illumina-dragen/templates/

[12] → aws s3 cp s3://aws-quickstart/quickstart-illumina-dragen/templates/dragen-master.yaml -

As you percolate on what you've learned…

Part II of this guide is meant as a (simple-as-possible) introduction to AWS Batch and Dragen. If any slide in Part II takes you longer than 10 minutes to figure out — or if you want to have a conversation about how to expand on what we've learned here — feel free to contact me.

Part II, Created May 2019, Last updated Oct 2019
Contact: Noora Siddiqui, nooras@bcm.edu