GARVAN INSTITUTE

GENOME.ONE

CONTAINERISED BIOINFORMATICS PIPELINE ON AWS

JOINT PILOT PROJECT BETWEEN GARVAN AND GENOME.ONE

PRESENTED BY LIVIU CONSTANTINESCU

Learn more about AWS at - http://amzn.to/2yXbKsW.
The Garvan Institute of Medical Research and Genome undertook a joint pilot project with AWS to optimise and customise our genomic analysis. We have run over 4,500 genomes through a new genomic pipeline, developed to the Broad Institute Best Practices leveraging Amazon's ECS, SQS, RDS, CloudFormation, CloudWatch and Spot Instances to optimise specifically for the cloud. This session will present our architecture, learnings, and the cost reductions we've achieved.

XTen Sequencing Sites

Around the World

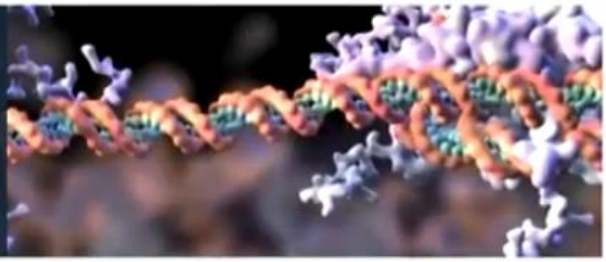
Garvan (via Genome.One) is the Largest Genome Sequencing Facility in the Southern Hemisphere

Our researchers are leaders in biomedical and clinical sciences. We have one of the first clinical genomics enterprises in the world and strengths in key enabling technologies such as data mining and software engineering. Using our Illumina XTen sequencers, a person's entire DNA sequence — three billion letters of genetic code — now can be read in Australia in just a few days for about $1500.

The Problem of Scale

And the Importance of Cost Optimisation

Genome.One owns **12** high-throughput **next-generation sequencers**, and analyses up to **300 genomes** every week.



Faced with an increasingly commoditised market, and competitors with much larger teams, we need to **lower costs** in all aspects of our enterprise while maintaining our very high levels of **accuracy**, **traceability**, **documentation**, **consistency**, **functionality** and **quality**. Further, we need to be able to pivot quickly, and innovate fast, to stay at the forefront of genomics research in Australia, and the world.

We've tackled this by adopting **agile**, **lean** approaches at every level of our business, and building strong in-house **software development** capabilities. Now we are shifting towards a "software first, hardware last" model for our **bioinformatics pipelines**, applying the same rigorous, change-managed **continuous improvement** model to our infrastructure, science and compute that we do to the software we develop in-house.

Software First, Hardware Last

04

Hardware is big, slow-moving, and quickly out of date.

It represents a high transaction cost, and depreciates in value rapidly post-purchase. This applies to everything from our genetic sequencers to the machines that analyse the data that comes out of them. Treating our hardware infrastructure the same way we do our code offers us numerous advantages:

- Cost Reduction
- Faster Analysis Speed
- Risk Reduction, by Removing Errors and Security Violations
- Aside from that, it also led to a:

20 x

cost reduction in compute

A Containerised Bioinformatics Pipeline

05

The Processing Pipeline

- We sequence whole human genomes at scale (up to 18,000 per year)
- Each genome produces ~80 GB of raw data (in FASTQ format).
- We need to run every such genome through an analysis process yielding BAM files (~160GB) and finally gVCFs (~8GB).
- Each genome can be processed independently, and is thus a great candidate for computing in parallel at scale.
- Continuous improvement also drives further increases in data size.

What we Built

- We created a **Docker image** that:
  - Downloads raw data from NCI or Amazon S3.
  - Processes this data.
  - Uploads the results to NCI or Amazon S3.
- It is optimised for c3.8xlarge instance (using 320GB of ephemeral storage)
  - Runs a genome in approximately 20 hours (including data transfer time).
  - Each stage is maxing out either CPU or RAM.
  - c3.8xlarge on demand price is $2.117/h (~$40/genome).
    - Infrastructure as Code allows us to take advantage of the Spot price, however, which hovers around 35 – 50c (~$7-10/genome).

This Pipeline's First Research Application

Introducing ISKS and the Medical Genome Reference Bank

ISKS (n=1,000) – Young, Affected

MGRB (n=4000) – Elderly, Healthy

Developing A Method of Identifying Individuals at High Risk of Sarcoma

Sarcomas are rare and deadly cancers that are usually diagnosed at an advanced stage or following metastasis. Earlier diagnosis of sarcoma is expected to lead to greatly improved patient survival, but the rarity of the cancer in the general population makes rapid diagnosis extremely challenging.

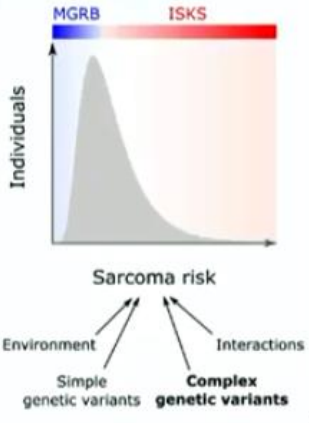
Multiple lines of evidence suggest that sarcoma risk has a substantial genetic component: sarcomas overwhelmingly affect the young, sarcoma survivors are at increased risk of second cancers. The two research cohorts used in this study (MGRB and ISKS) represent the extremes of sarcoma risk.

They will be made available to all Australian scientists, via the SGC web portal at: sgc.garvan.org.au

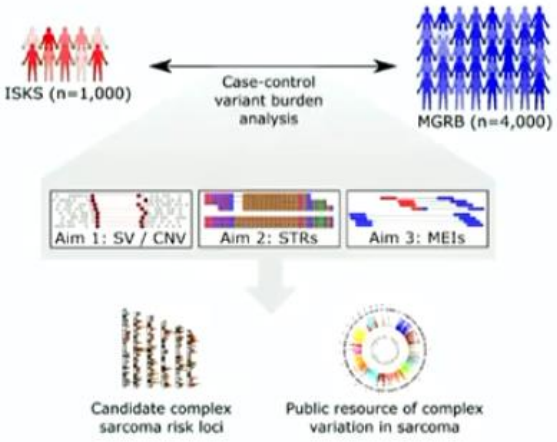
The Benefit to Australia

Major Advances in Sarcoma Research, and a Universal Control Cohort for Genomics Studies

a) Risk in the population



b) Project outline



Two Main Analyses will Follow:

MGRB will be analysed to:

- Assess the genomic patterns associated with healthy old age.
- Provide a control sequencing cohort for disease linkage studies.

MGRB will be compared with ISKS to:

- Determine loci and genes involved in Sarcoma Risk.

The MGRB & ISKS projects together represent 10 million AUD of research investment.

Our Process, Step By Step

08

How we Turned our Container into a Change Managed, Optimised & Repeatable Pipeline

An Investment in the Future

We never intended this architecture to be running 24/7, and neither did we intend it to be dedicated to this specific research study. Nor, for that matter, do we want to be limited to a single instance.

Like all other code in the Genome.One Ecosystem, and the container the pipeline is based on, our entire compute architecture is source controlled, change managed, and deployable at the click of a button.

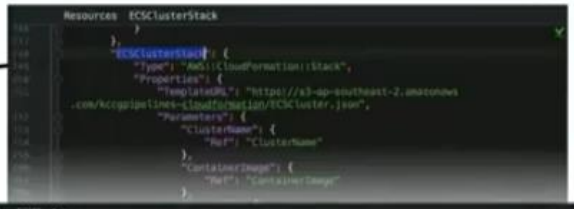
For this, we used AWS CloudFormation.

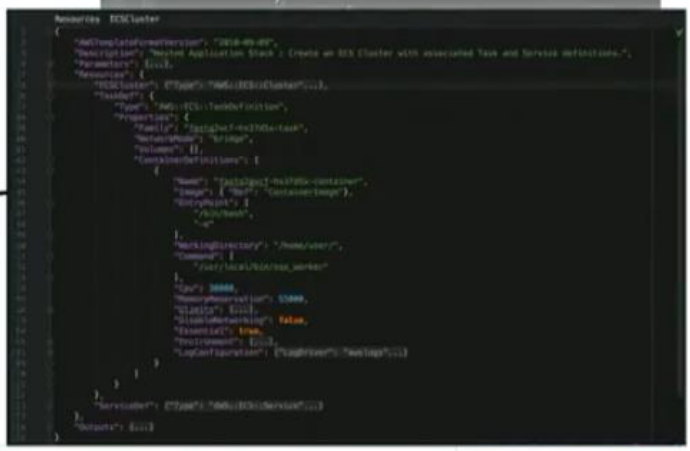
Deployment Automation

09

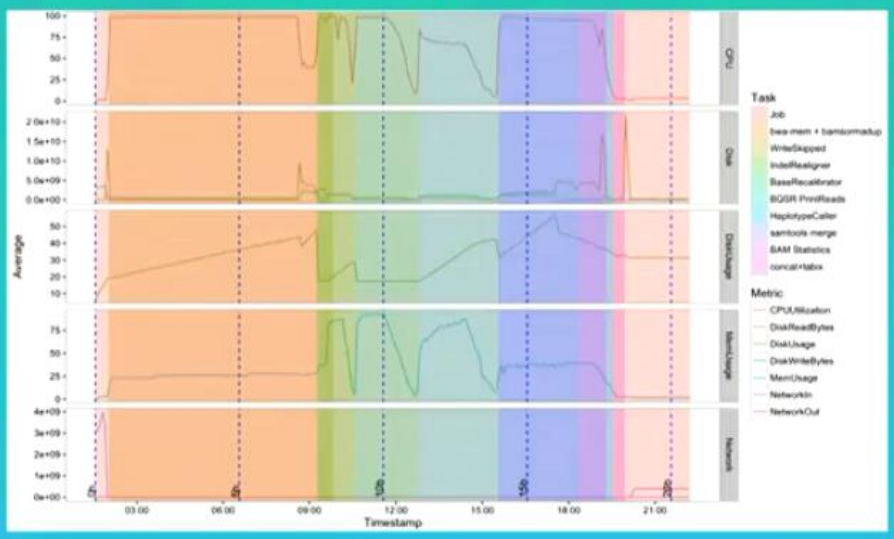
Using Nested Templates in AWS CloudFormation

- **kccgpipelines-master.json**
  - Network.json
    - Nat.json
    - S3endpoint.json
  - Bastion.json
  - DB.json
  - SQS.json
  - **ECSCluster.json**
  - App.json
  - ECSCluster-monitor.json
- DirectConnect.json
  - DataMover.json





Compute Optimisation



- Every stage of the pipeline has differing compute needs.
- Each stage is maxing out either CPU or RAM

Smart Queuing System

Amazon ECS

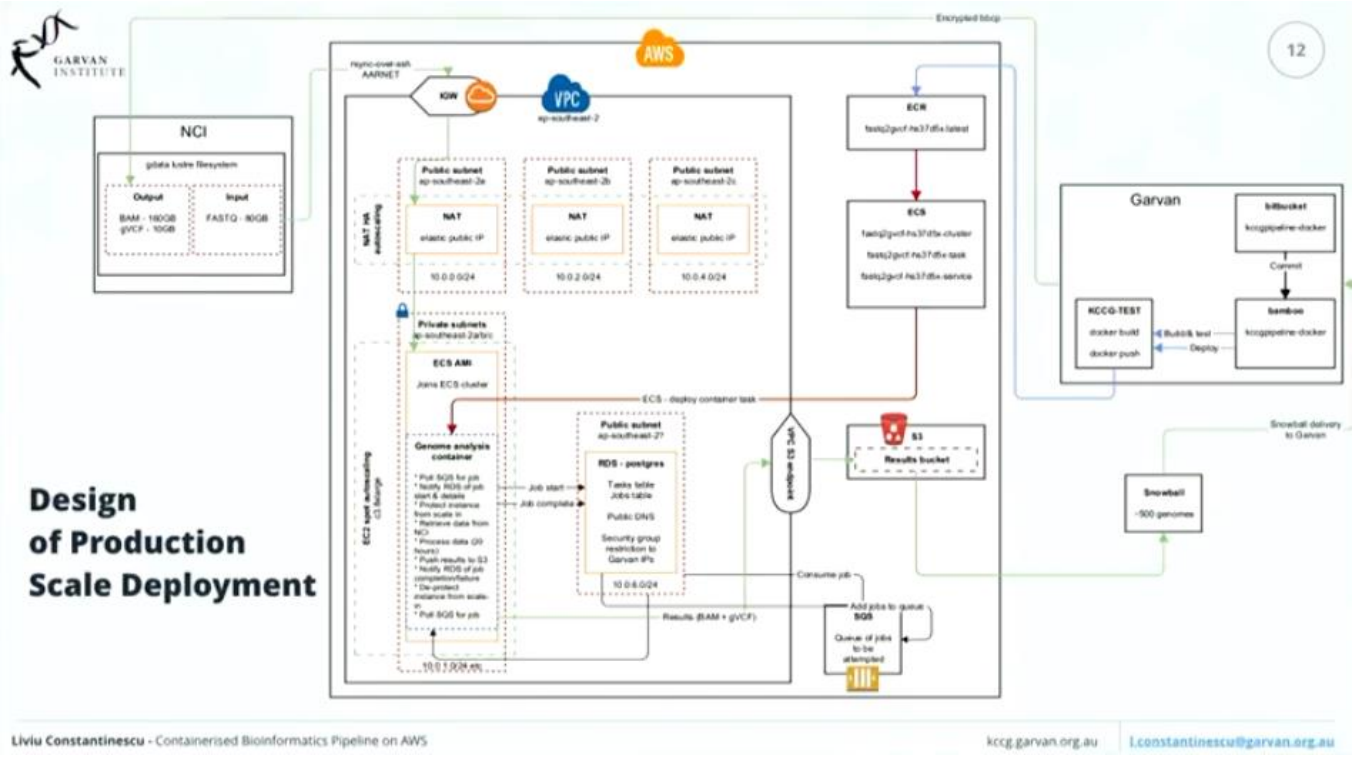
- Create an Amazon ECS cluster to run 1 container per instance.
  - Create an Amazon EC2 autoscaling group to run containers on.

Amazon RDS

- Serves a simple task management database:
  - 1 genome == 1 task
  - 1 job == 1 attempt at completing a task

Amazon SQS

- If a task does not have a queued or running job, create a new job and submit to an Amazon SQS queue.
  - When tasks are in the Amazon SQS queue, scale up the Amazon EC2/ECS cluster.
  - When containers are not working and the Amazon SQS queue is empty, scale down the Amazon EC2/ECS cluster.


GARVAN INSTITUTE
12
Design of Production Scale Deployment
NCI
AWS
VPC
ECR
ECS
Garvan
bitbucket
bamboo
KCCG-TEST
S3
Results bucket
Snowball
SQS
RDS - postgres
Genome analysis container
ECS AMI
Public subnet
Private subnets
NAT
Liviu Constantinescu - Containerised Bioinformatics Pipeline on AWS
kccg.garvan.org.au
l.constantinescu@garvan.org.au


GARVAN INSTITUTE
13
Information Radiators
Displaying Pipeline Information at-a-Glance
Pipeline Monitoring Dashboard
857
0
30.2
126
122
122
0
121
173
33
25.25
30.74
0.93
0.499
0.522
Complete End-to-End Reporting
Connecting our existing suite of CI/CD tools (Atlassian's Bamboo/Bitbucket, with sample flow monitored in Atlassian JIRA) to the system was easy and effective, via some simple dashboarding tools and Atlassian JIRA, alongside the AWS CLI.
Total Control of Billing
By using billing tags and automatically assigning these to all resources via the AWS CloudFormation scripts that create them, plus specifying our desired range of Spot prices, we maintained complete control over what we were paying, when.
Shared Vision and Transparency
We continuously monitored run time, the number of tasks in the queue, the prices of the instances we needed on Spot, instances being used, data in the Amazon S3 buckets. We tied alerts to all of these.
Liviu Constantinescu - Containerised Bioinformatics Pipeline on AWS
kccg.garvan.org.au
l.constantinescu@garvan.org.au

State of the Pipeline

Overview

- 4,500 whole genomes processed.
- Raw data currently staged on NCI.
- Data pulled up to Amazon EC2 for processing, results deposited into an Amazon S3 bucket.
- Processing occurs in a c3.8xlarge autoscaling group of max 150 instances with max spot bid @ $2.90. Costing for Phase 1 to follow.
- After project is finished, results are pulled back down to NCI using datamover nodes in a DirectConnect VPC.
- Phase 1 data is already up on sgc.garvan.org.au, accessible for free.

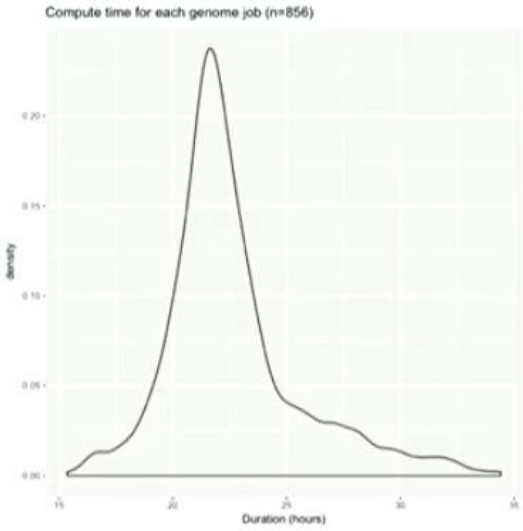
Egress Options

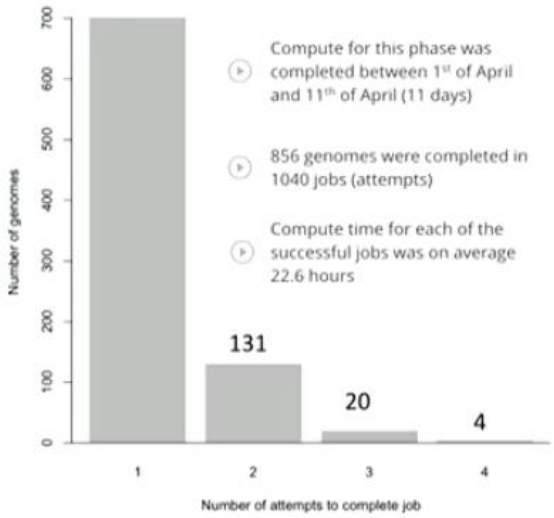
- Direct Connect from AWS and NCI.
- Direct Connect between AWS and Garvan (via UNSW).
- Use AWS Snowball to deliver the genomes on a physical device.
- Partial egress waiver available through AWS for scientific studies like this one.

- Phase 2 metrics are 2859 high-quality samples, with roughly 70 million loci genotyped in each. Roughly 3 million CPU-hours, 1.1 Petabytes data.

Phase 1 Reliability & Speed Metrics

Reasons for Retries: Accidental terminations (~20), Spot terminations, Container termination (due to pipeline errors).
Pipeline errors were largely OOM errors in the HaplotypeCaller stage, likely due to recent real world data being larger than the test data.



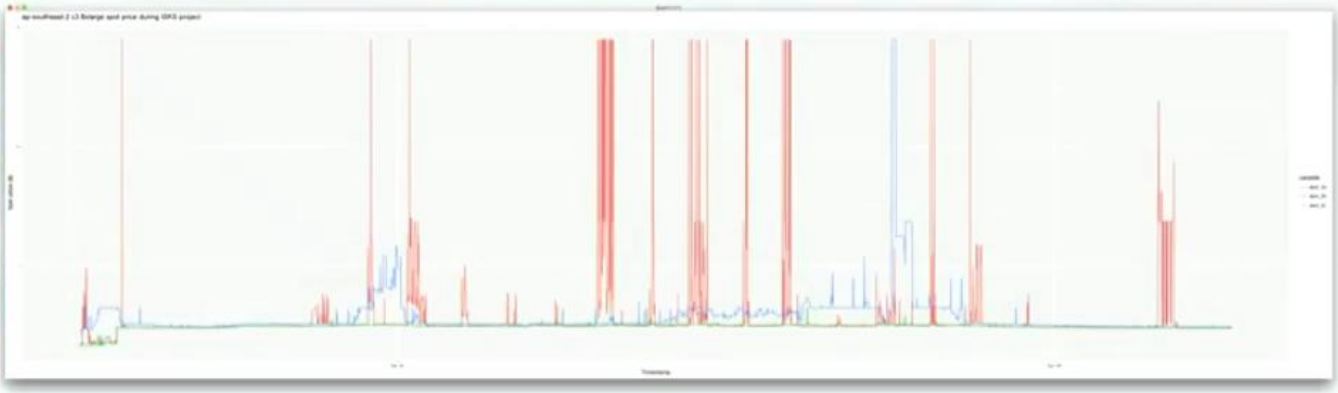


- Compute for this phase was completed between 1st of April and 11th of April (11 days)
- 856 genomes were completed in 1040 jobs (attempts)
- Compute time for each of the successful jobs was on average 22.6 hours

Spot Price Fluctuation Observed

16

(Some terminations here and there, mainly in AZ 2A)



- In our trials, spot prices scaled comfortably to about 150 c3.8xlarge instances and 200 r3.8xlarge instances (about 10000 cores total), before prices were significantly affected.

NAT Network Throughput

17

We observed significant spend on data mover instances, and will Optimise this in Future

- NAT Gateway pricing sub-optimal here.
- Instead, used c3.large instances, but maxed out at ~4GB/s data throughput.
- Upgrading to c3.8xlarge saw bandwidth increase to 20GB/s or higher.
- Due to the significant costs involved, we suggest that anyone building a similar architecture use a Squid Proxy instead.
- See: https://aws.amazon.com/articles/5995712515781075

Per-Genome Costs Observed

- Compute
  - Total for 856 samples = $15,093
    - $17.6 per genome
    - $12,409 on c3.8xlarge spot instances for compute
    - $2,476 on NAT instances (could be improved, via Squid Proxy solution)
- Data egress
  - Amazon S3 -> Direct Connect -> UNSW -> NCI
  - Roughly $8.7 per genome
- Amazon S3 storage
  - Total for 856 samples = $1,954 ($2.2 per genome)
- Grand total estimate == $28.5 USD (no GST)
  - $37.33 AUD per genome.

Further Evolution

Our Roadmap for Refining the Bioinformatics Pipeline

Phase 3 – More Instance Types

We'd like to optimise container versions for multiple instance types, and use SpotFleet to deploy based on what's economical/available. Our pipeline's output ought to scale linearly with CPU cores, given sufficient RAM. We'll also investigate the use of Amazon EBS over ephemeral storage.

Phase 2 – Adjust Data Metrics

We can reduce the frequency of Out of Memory errors with some more work on the pipeline's optimisation, to account for the spikes in RAM usage we observed.

Complete – Containerised AWS Pipeline

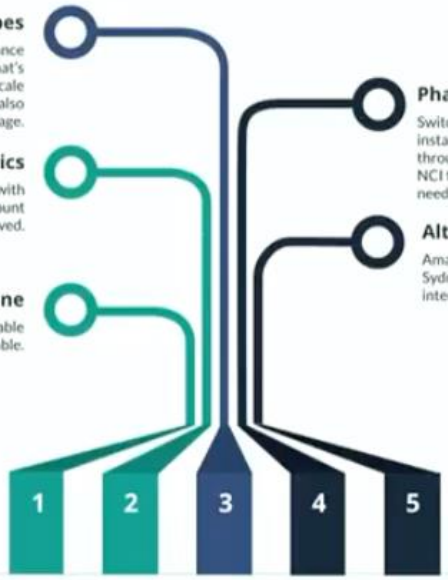
Pipeline works end-to-end, optimised for the cloud, and able to produce a high-quality result within the budget available.

Phase 4 – Optimise NAT Instances

Switch over to a Squid Proxy model over our current NAT instances, or simply find a cheaper instance type with higher throughput. Alternatively, we can preload all raw data from NCI to Amazon S3 (over Direct Connect) and likely avoid needing (expensive) NAT instances altogether.

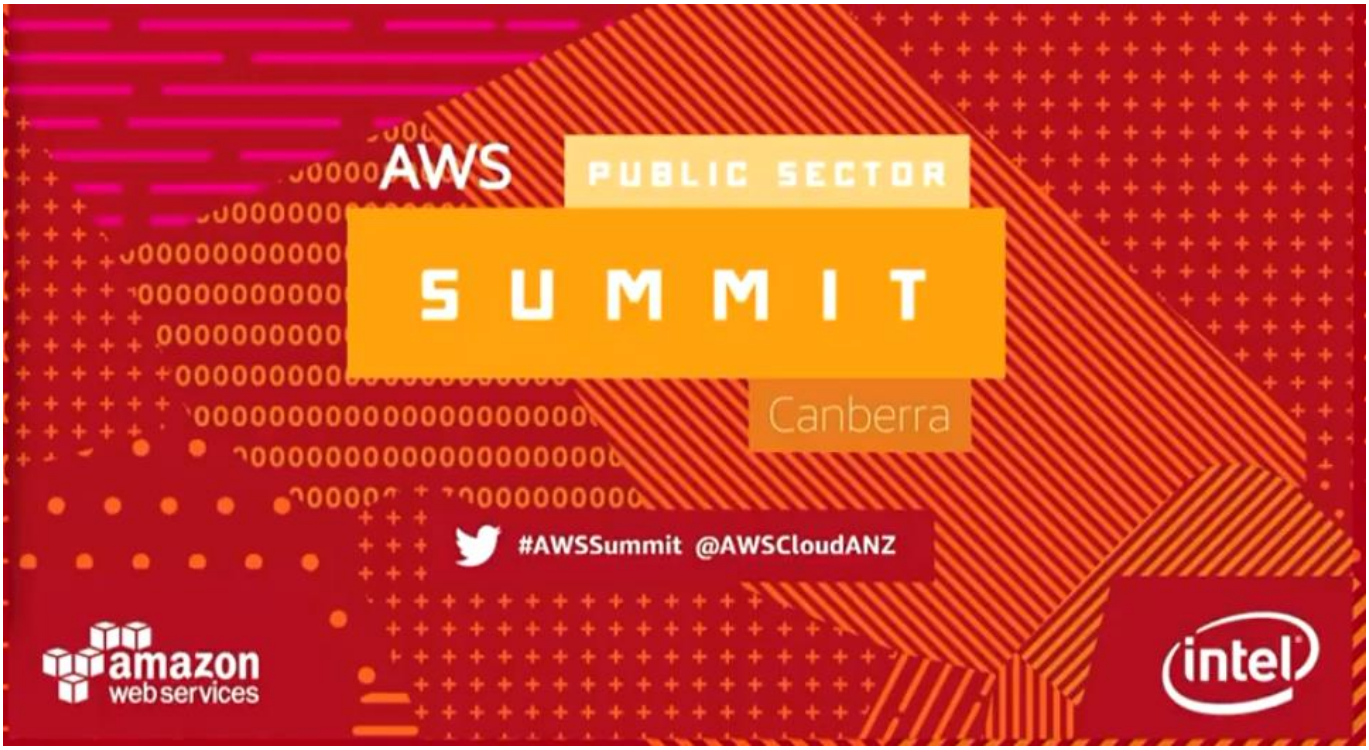
Alternative – AWS Batch

Amazon's new service, AWS Batch, was not available in Sydney when we built this pipeline, and we are very interested in its potential to bring even more gains.

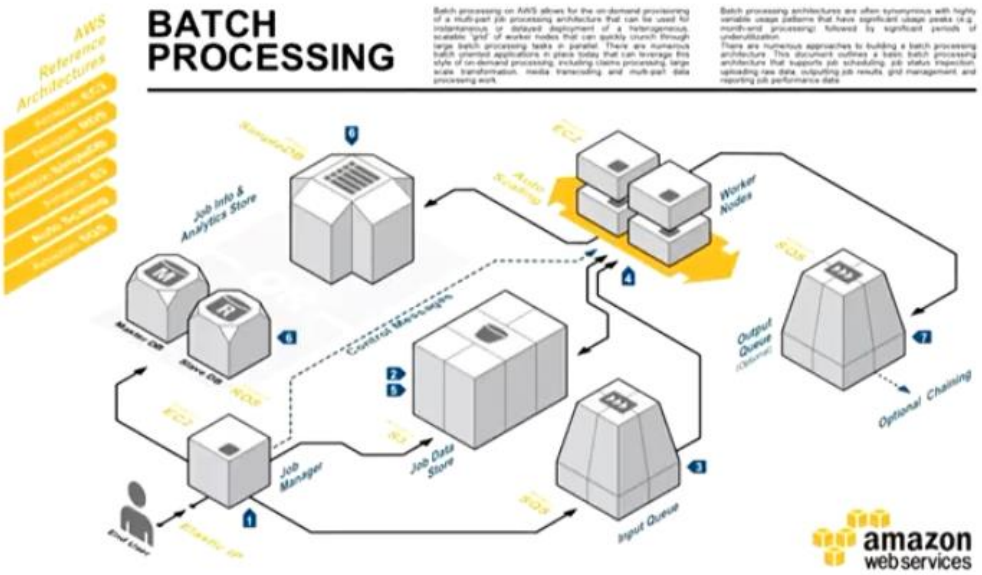


Special Thanks

- Our sponsors and collaborators (shown to left).
  - Our sequencing staff at Garvan and Genome.One.
- NSW OHMR
  - For funding the research study.
- Aspree
  - For providing samples.
- The 45 and Up Study
  - For providing samples.
- Amazon Web Services
  - For Jamie and Ben's valuable assistance completing this pilot study.
  - For defraying some of the costs of developing the pipeline, and of data egress for the study.



Batch computing could be easier…



AWS Components:

- Amazon EC2
- Spot Fleet
- Auto Scaling
- Amazon SNS
- Amazon SQS
- CloudWatch
- Lambda
- Amazon S3
- DynamoDB
- API Gateway
- …

Introducing AWS Batch

Managed

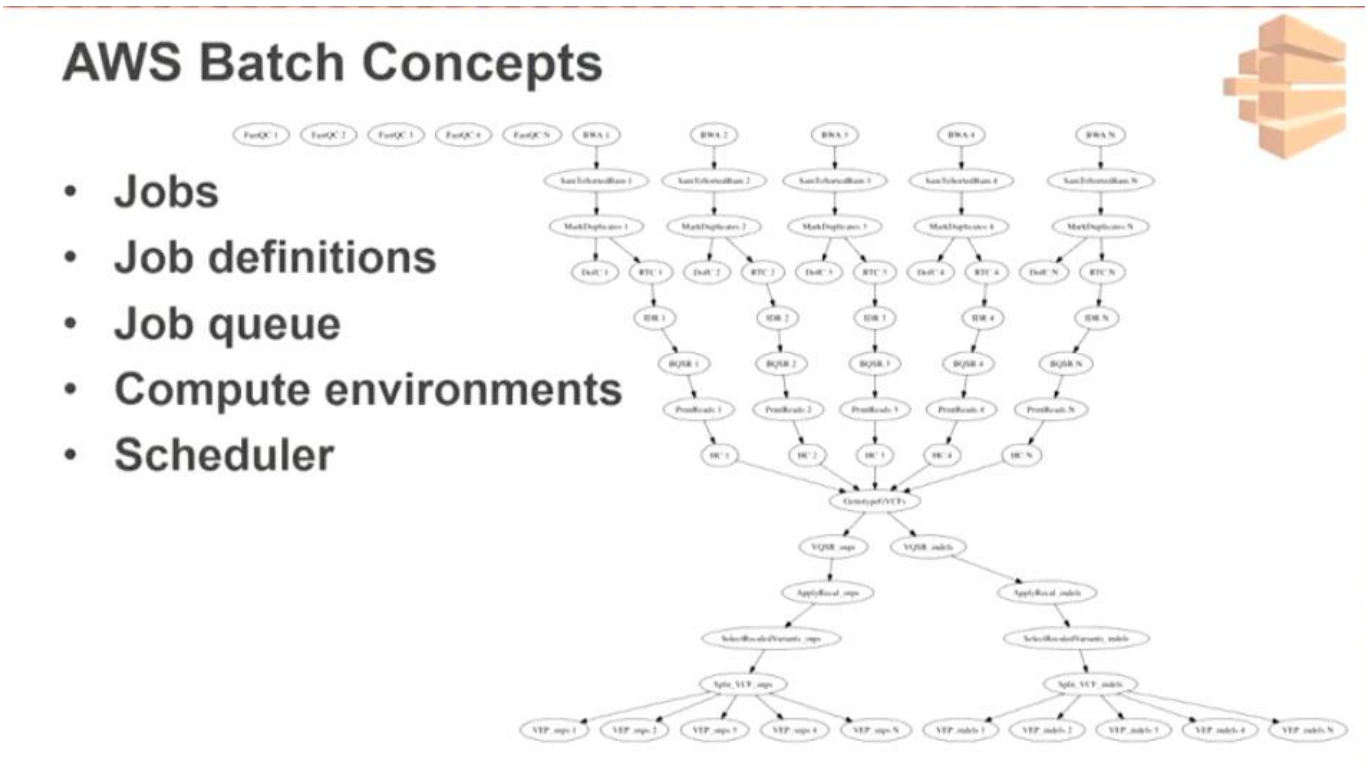
No software to install or servers to manage. AWS Batch provisions, manages, and scales your infrastructure

Integrated with AWS

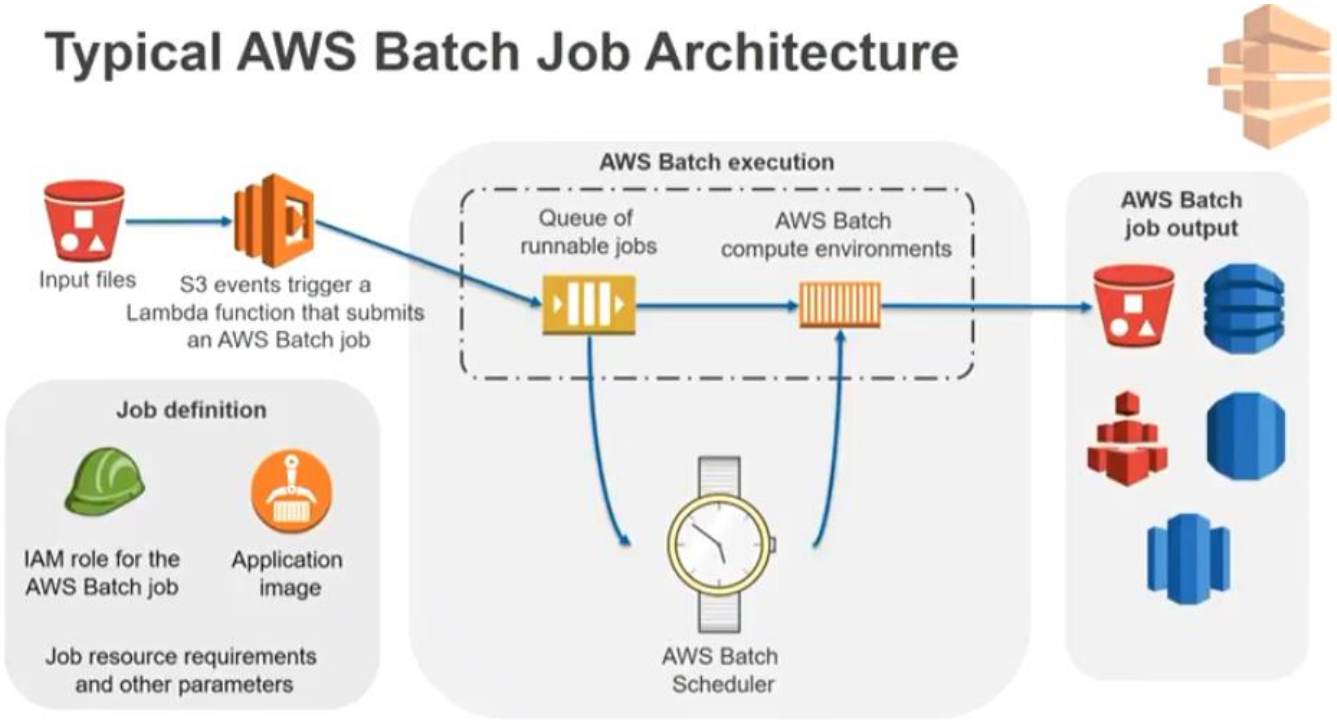
Natively integrated with the AWS platform, AWS Batch jobs can easily and securely interact with services such as Amazon S3, DynamoDB, and Rekognition

Cost-optimized resource provisioning

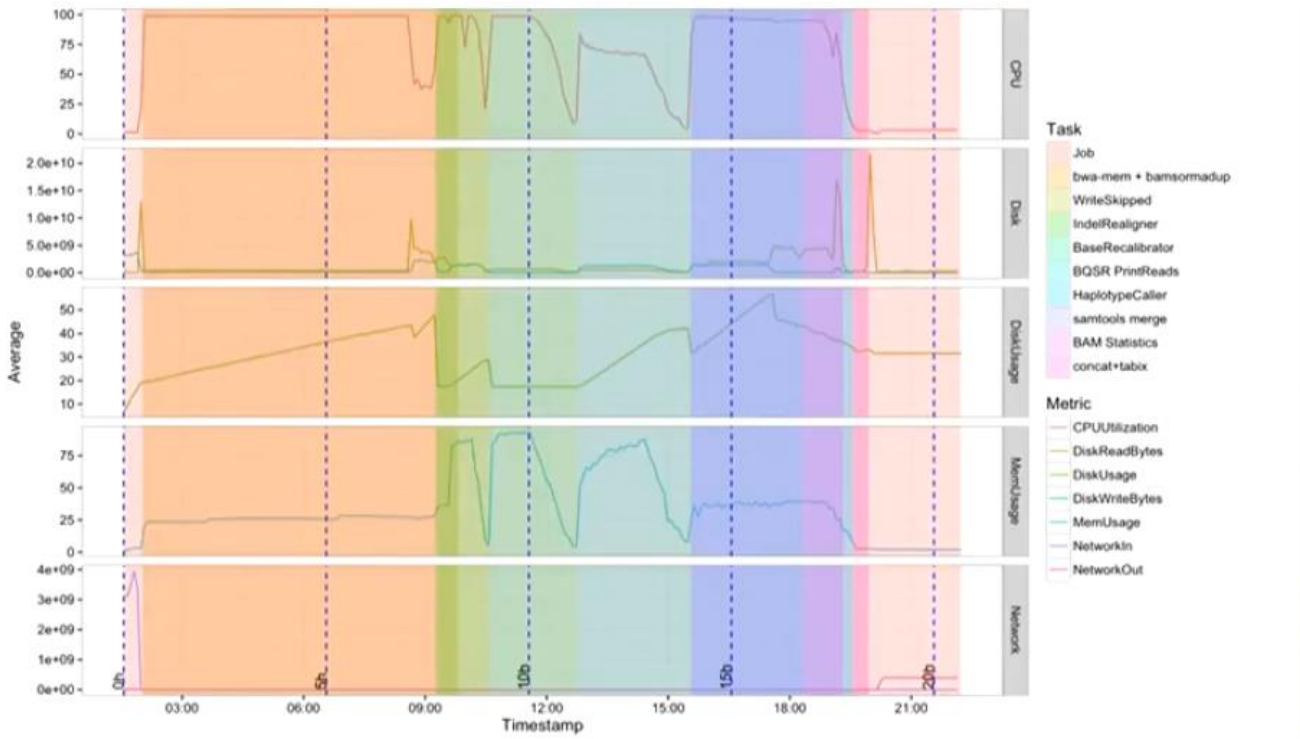
AWS Batch automatically provisions compute resources tailored to the needs of your jobs using Amazon EC2 and Spot Instances



With AWS Batch, you create Jobs. With a job definition, you define what vCPUs it's going to use, what image it needs to use, what storage to mount, and all other needed job characteristics. When you submit a job to a Job Queue, it will be automatically taken up by the Scheduler and ran in the appropriate compute environment.



Input files being put into S3 will trigger a lambda function that submits an AWS Batch Job in a runnable queue, the Scheduler will then take up those jobs and put them into compute environments that manages the computing needs for the job. The results are then stored again in S3 where they can be used for further analysis



Since each application has specific compute requirements like memory, CPU, or I/O, they can be optimized. By breaking down the specific requirements into specific jobs, we can use AWS Batch to schedule them into computing environments that match the specific computing needs of each job.

Common AWS Batch Configurations

You can achieve different objectives via AWS Batch through service configuration and solution architectures:

Cost-optimized

- Minimize operational overhead
- Work can happen any time over a multi- hour period (or a weekend)
- Monte-Carlo simulations or bulk loan application processing

Resource-optimized

- Budget constraints
- Multiple job queues, priorities, sharing compute environments
- Existing compute resources that are available / underutilized (RI, SF, etc.)

Time-optimized

- Workloads with firm deadlines
- Queue w/ primary compute environment using RIs and fixed capacity and a secondary Spot CE
- Financial settlement

Thank you!