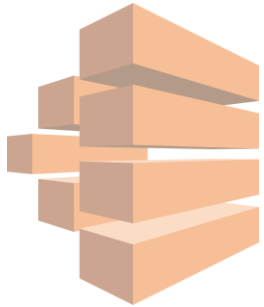
Running Cost Effective Batch Workloads with AWS Batch and Amazon EC2 Spot Instances

Chad Schmutzer, Solutions Architect – EC2 Spot
schmutze@amazon.com

October 2018

Agenda



- Amazon EC2 Spot Instances
- What is batch computing?
- AWS Batch overview and concepts
- Use cases
- Let’s take it for a spin!
- Q&A

Amazon EC2 Purchasing Options

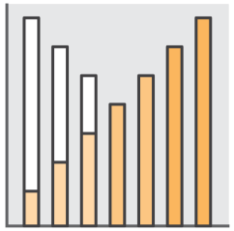
On-Demand

Pay for compute capacity by **the second** with no long-term commitments

Spiky workloads, to define needs

Reserved Instances

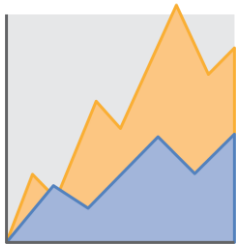
Make a 1- or 3-year commitment and receive a **significant discount** off On-Demand prices



Committed, steady-state usage

Spot Instances

Spare EC2 capacity at **savings of up to 90%** off On-Demand prices



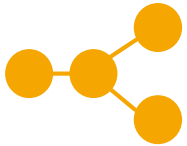
Fault-tolerant, flexible, stateless workloads

What are Amazon EC2 Spot Instances?

Spare EC2 capacity that AWS can reclaim with a two-minute notice

Low Cost

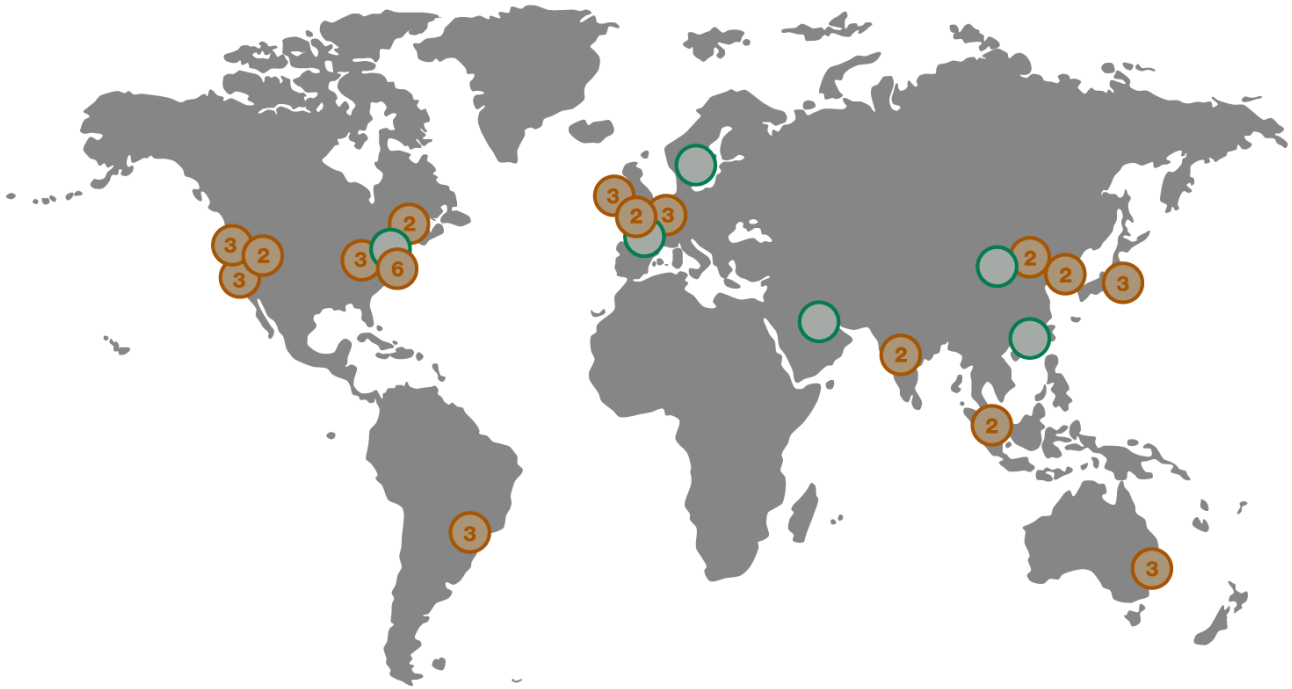
Faster Results



Easy Access

Resource Flexibility

Spot Instances: Spare Compute Capacity at *Scale*

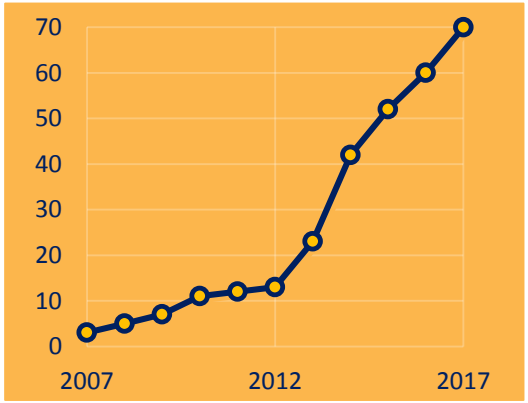


Supports all EC2 Instance Types and Available in all Public Regions and Availability Zones

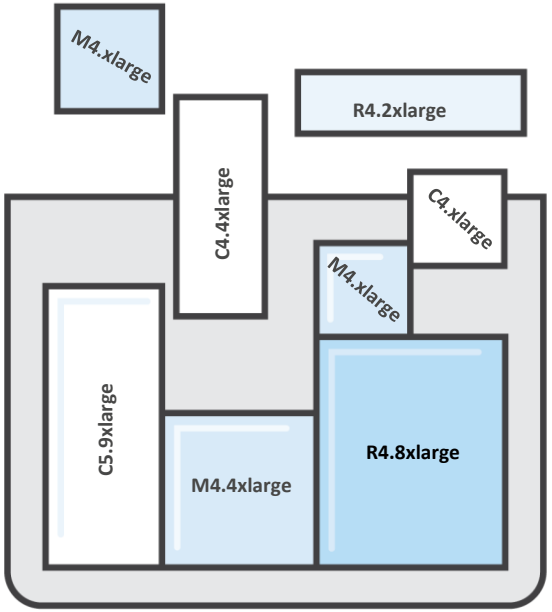
GovCloud Coming Soon!

Understanding how to use EC2 Spot pools

Each instance family, each instance size, in each Availability Zone, in every Region is a ***separate*** Spot pool



100s of Instance Options



Instance Flexibility and Diversification are crucial

What does it mean to be instance flexible?

Instance Size

- m4.xlarge capacity is different from m4.large. Use as many sizes as you can efficiently!

Instance Family

- If you're using the c4.large, you can almost certainly use m4.large and r4.large. They have the same number of vCPUs, just with some extra memory!

Availability Zone

- Availability Zones consist of one or more discrete data centers. us-east-2a and us-east-2b capacity is different! If your application can use multiple AZs, do it!

EC2 Spot Fleet

Automates the management of Spot instances – Simply tell Spot Fleet how much capacity you need and Fleet does the rest.

Fast Access to Capacity

- Launch Thousands of Spot Instances with single API call
- Maintain capacity or scale up or down based on metrics and thresholds you set

Optimize Price vs. Availability

- Find the lowest priced horsepower that works for you or diversify your fleet to increase availability
- Auto-attach to a Load Balancer

Customize Based on Application

- Create your own capacity unit based on your application requirements

Spot Instances Use Cases

Big data




FINRA has **saved up to 50%** from its on-premises solution, increased elasticity/scalability, and accelerated reprocessing requests from months to days with EC2 Spot Instances

Containers & test/dev




Yelp runs **millions of tests** every day with EC2 Spot Instances. Yelp improved test result response time from **2 days to 30 minutes** and has also delivered a large reduction in execution costs with Spot.

HPC & batch



TLG Aerospace saw a **75% reduction in the cost** per CFD simulation with Amazon EC2 Spot Instances. They were able to pass those savings along to their customers and be more competitive.

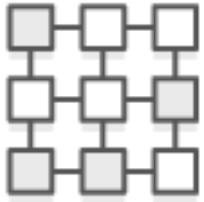
Stateless web services




AdRoll have been able to seamlessly scale their infrastructure, better serve customers across the globe, and **reduce our fixed costs by 75% and operational costs by 83%.**with AWS solution, including EC2 Spot Instances

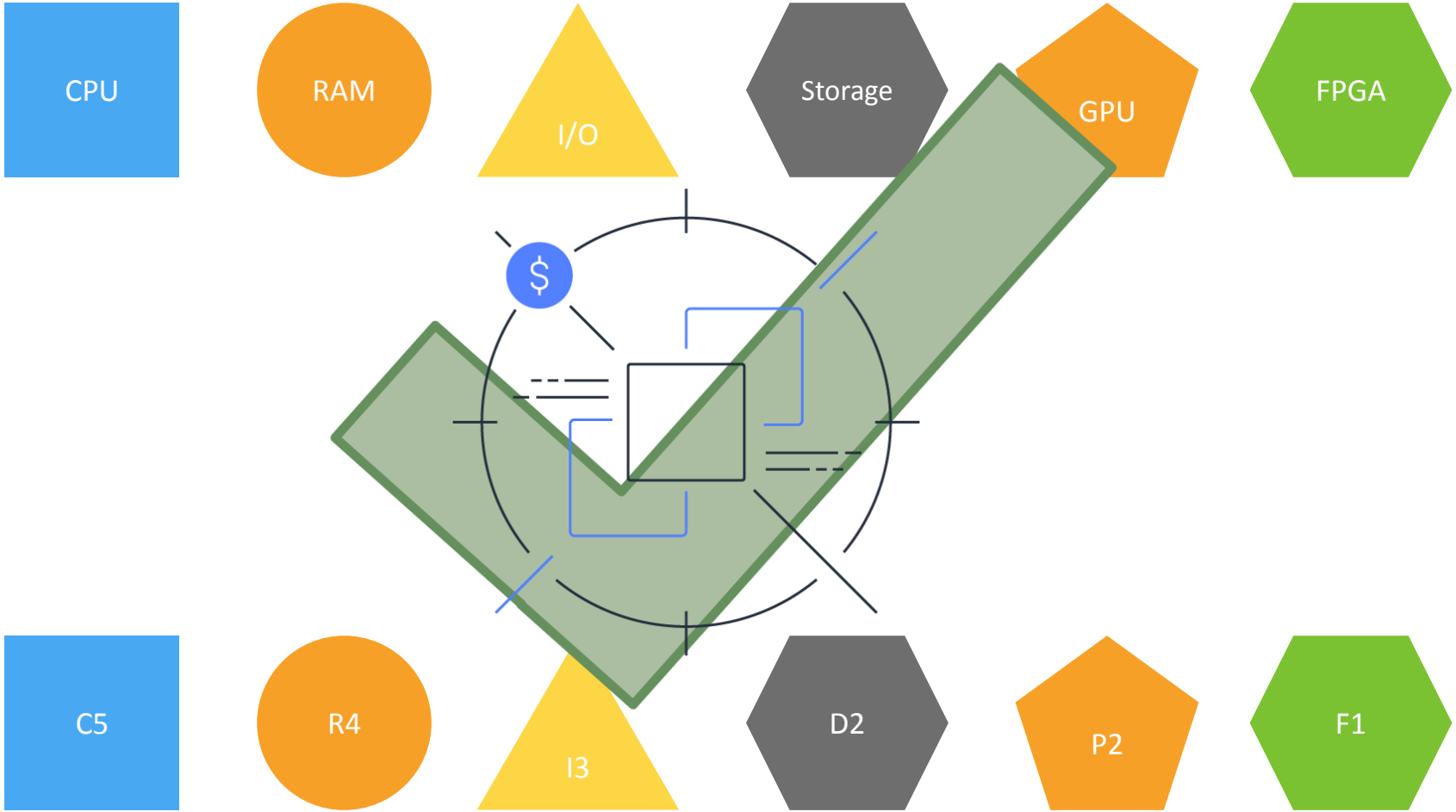
What is batch computing?

Run jobs asynchronously and automatically across one or more computers.



Jobs may have dependencies, making the sequencing and scheduling of multiple jobs complex and challenging.

How does batch computing work in the cloud?


CPU
RAM
I/O
Storage
GPU
FPGA
$
C5
R4
I3
D2
P2
F1

However, batch computing could be easier…

AWS Components:

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

Introducing AWS Batch

Fully 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 EC2 Spot

Introducing AWS Batch

- Fully-managed batch primitives
- Focus on your applications (shell scripts, Linux executables, Docker images) and their resource requirements
- We take care of the rest!

AWS Batch Concepts

- Jobs
- Job Definitions
- Job Queue
- Compute Environments
- Scheduler

Jobs

Jobs are the unit of work executed by AWS Batch as containerized applications running on Amazon EC2.

Containerized jobs can reference a container image, command, and parameters or users can simply provide a .zip containing their application and we will run it on a default Amazon Linux container.

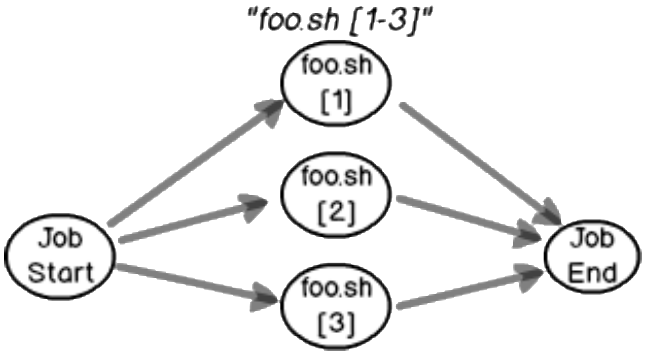
```
$ aws batch submit-job --job-name variant-calling
--job-definition gatk --job-queue genomics
```

Easily run massively parallel jobs

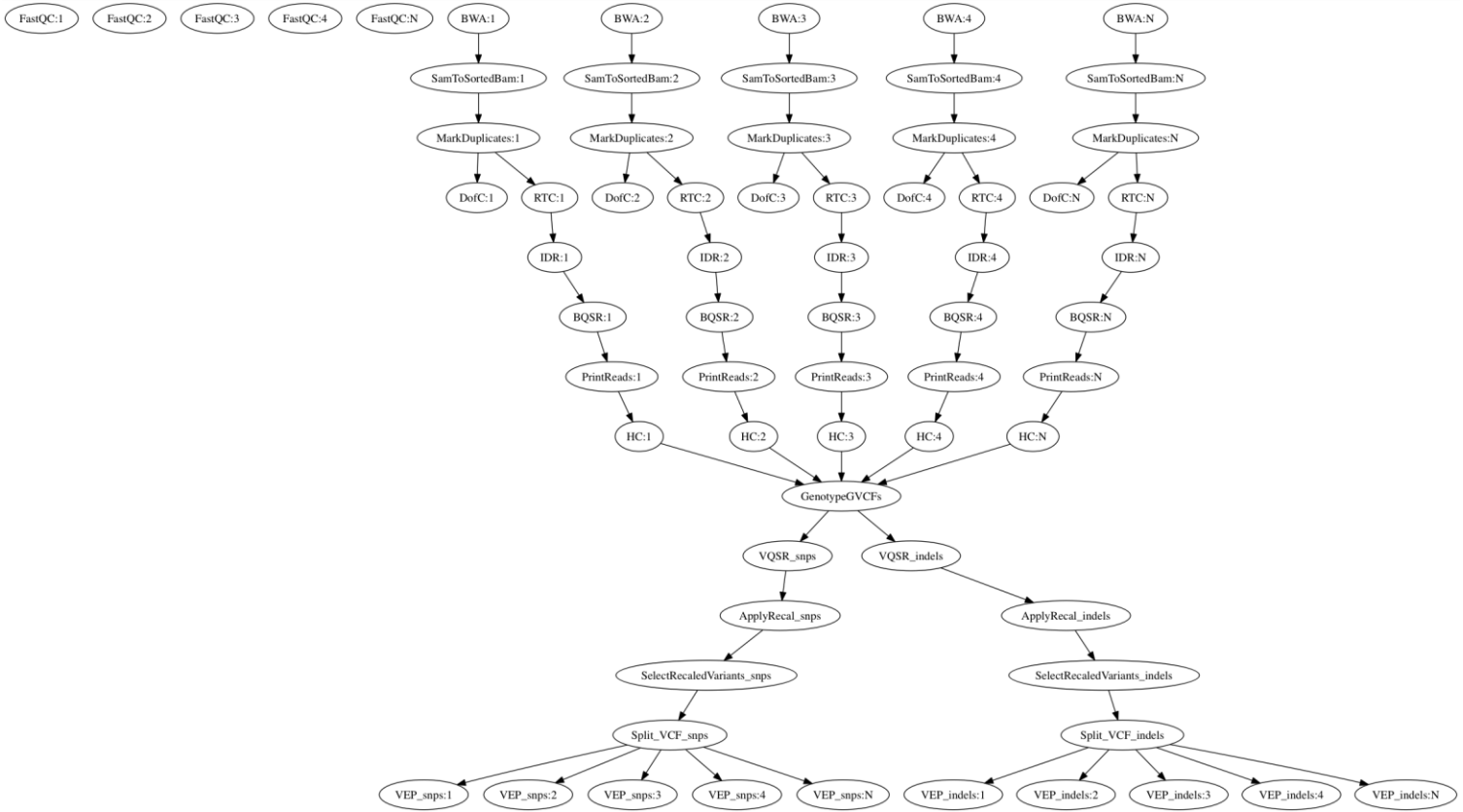
Users can submit a large number of independent “**simple jobs,**” or “**array jobs**” that run many copies of an application against an array of elements.

Array jobs are an efficient way to run:

- Parametric sweeps
- Monte Carlo simulations
- Processing a large collection of objects

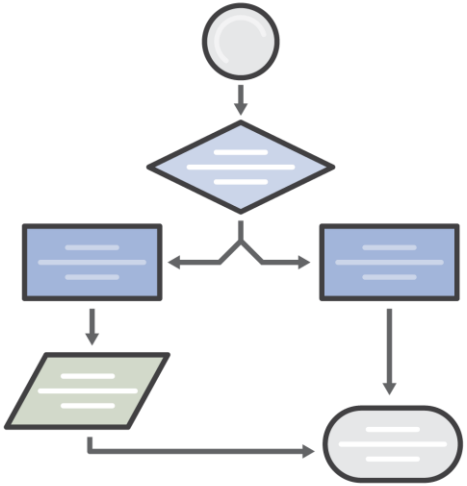


Workflows and Job Dependencies



Workflows, Pipelines, and Job Dependencies

Jobs can express a **dependency** on the successful completion of other jobs or specific elements of an array job.



Use your preferred workflow engine and language to submit jobs. Flow-based systems simply submit jobs serially, while DAG-based systems submit many jobs at once, identifying inter-job dependencies.

```
$ aws batch submit-job –depends-on 606b3ad1-aa31-48d8-92ec-f154bfc8215f ...
```

Job Definitions

Similar to ECS Task Definitions, AWS Batch **Job Definitions** specify how jobs are to be run. While each job must reference a job definition, many parameters can be overridden.

Some of the attributes specified in a job definition:

- IAM role associated with the job
- vCPU and memory requirements
- Mount points
- Container properties
- Environment variables

```
$ aws batch register-job-definition --job-definition-name gatk
--container-properties ...
```

Job Queues

Jobs are submitted to a **Job Queue**, where they reside until they are able to be scheduled to a compute resource. Information related to completed jobs persists in the queue for 24 hours.

```
$ aws batch create-job-queue --job-queue-name genomics
--priority 500 --compute-environment-order ...
```

Compute Environments

Job queues are mapped to one or more **Compute Environments** containing the EC2 instances used to run containerized batch jobs.

Managed compute environments enable you to describe your business requirements (instance types, min/max/desired vCPUs, and **EC2 Spot** max price as a % of **On-Demand**) and we launch and scale resources on your behalf.

You can choose specific instance types (e.g. c4.8xlarge), instance families (e.g. C4, M4, R3), or simply choose “optimal” and AWS Batch will launch appropriately sized instances from our more-modern instance families.

Compute Environments

Alternatively, you can launch and manage your own resources within an **Unmanaged** compute environment. Your instances need to include the ECS agent and run supported versions of Linux and Docker.

```
$ aws batch create-compute-environment --compute-
environment-name unmanagedce --type UNMANAGED ...
```

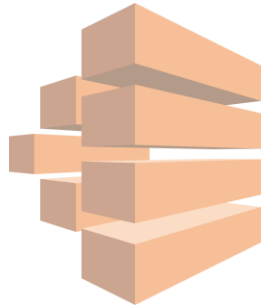
AWS Batch will then create an Amazon ECS cluster which can accept the instances you launch. Jobs can be scheduled to your Compute Environment as soon as your instances are healthy and register with the ECS Agent.

AWS Batch Concepts

The **Scheduler** evaluates when, where, and how to run jobs that have been submitted to a job queue.

Jobs run in approximately the order in which they are submitted as long as all dependencies on other jobs have been met.

Job States



Jobs submitted to a queue can have the following states:

SUBMITTED: Accepted into the queue, but not yet evaluated for execution

PENDING: Your job has dependencies on other jobs which have not yet completed

RUNNABLE: Your job has been evaluated by the scheduler and is ready to run

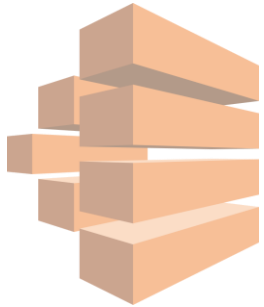
STARTING: Your job is in the process of being scheduled to a compute resource

RUNNING: Your job is currently running

SUCCEEDED: Your job has finished with exit code 0

FAILED: Your job finished with a non-zero exit code or was cancelled or terminated.

AWS Batch Actions



Jobs:

SubmitJob

ListJobs

DescribeJobs

CancelJob

TerminateJob

Job Definitions:

RegisterJobDefinition

DescribeJobDefinitions

DeregisterJobDefinition

Job Queues:

CreateJobQueue

DescribeJobQueues

UpdateJobQueue

DeleteJobQueue

Compute Environments:

CreateComputeEnvironment

DescribeComputeEnvironments

UpdateComputeEnvironment

DeleteComputeEnvironment

AWS Batch Actions

CancelJob: Marks jobs that are not yet STARTING as FAILED.

TerminateJob: Cancels jobs that are currently waiting in the queue. Stops jobs that are in a STARTING or RUNNING state and transitions them to FAILED.

Requires a “reason” which is viewable via DescribeJobs

```
$ aws batch cancel-job --reason “Submitted to wrong queue”
--jobId= 8a767ac8-e28a-4c97-875b-e5c0bcf49eb8
```

AWS Batch Pricing

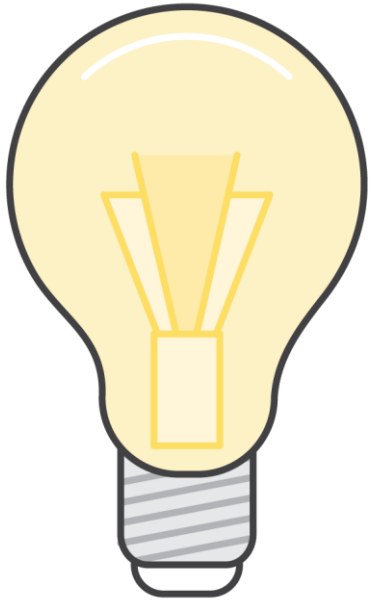
There is no charge for AWS Batch; you only pay for the underlying resources that you consume!

Service Comparisons

AWS Batch is one of many complementary services:

- **CfnCluster:** Elastic HPC cluster that is ideal for tightly-coupled, latency sensitive applications, or when customers would like to use an OSS or commercial job scheduler
- **Glue/DataPipeline:** ETL to/from relational databases with known schemas
- **EMR –** Managed MapReduce clusters using Hadoop/Spark for large-scale data processing
- **ElasticSearch -** Perform Web-Crawling on Social Media sites and populate results to a searchable dataset
- **Lambda –** Run short duration functions without provisioning or managing servers
- **Step Functions/SWF –** Design and orchestrate workflows, with support for branching and callouts to other AWS services.

Some ideas for inspiration



Typical AWS Batch Job Architecture

Input Files

S3 Events Trigger
Lambda Function
Submits Batch Job

AWS Batch Execution

Queue of
Runnable Jobs

AWS Batch
Compute Environments

AWS Batch
Scheduler

AWS Batch Job Output

Job Definition

IAM Role for
Batch Job

Application
Image

Job Resource Requirements
and other parameters

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

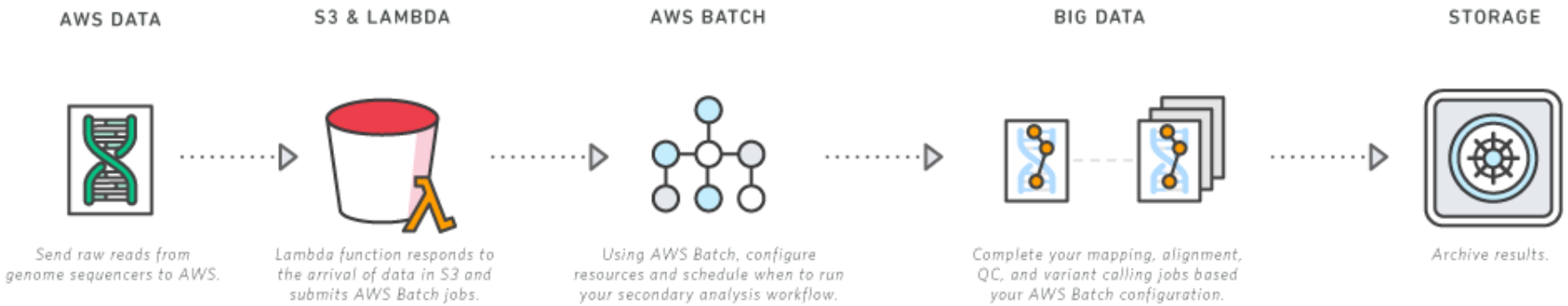
Resource Optimized

- Budget Constraints
- Multiple Job Queues, priorities, sharing compute environments
- Existing compute resources that are available / underutilized (RI, SF, etc.)

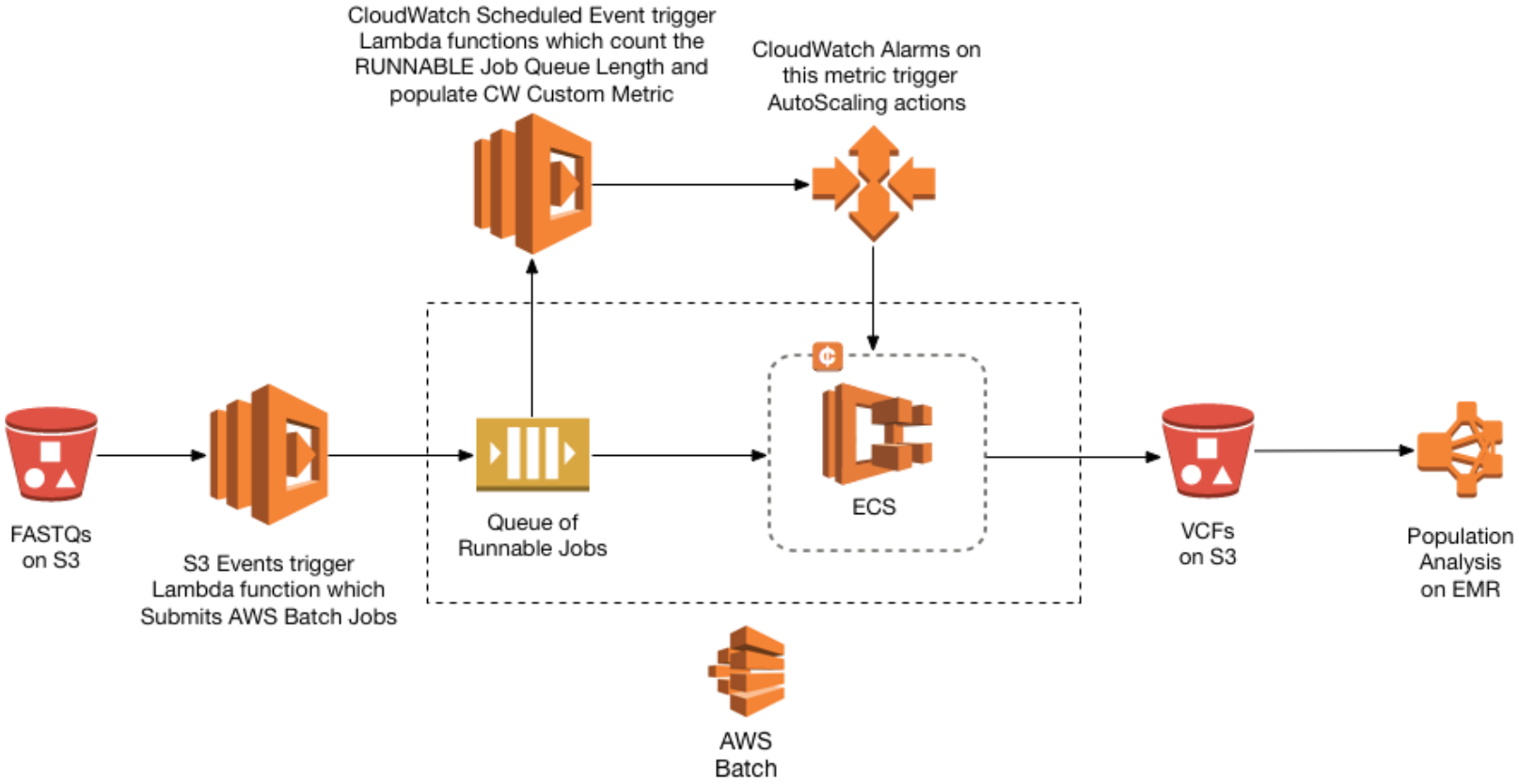
Time Optimized

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

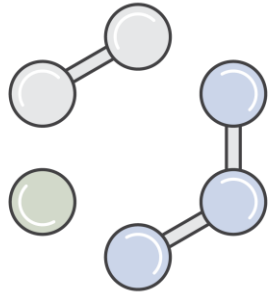
DNA Sequencing

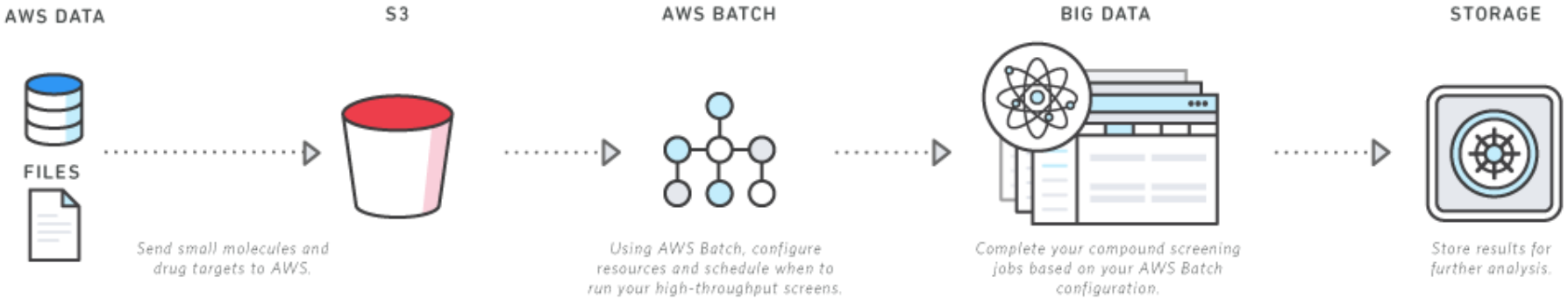


Genomics on Unmanaged Compute Environments

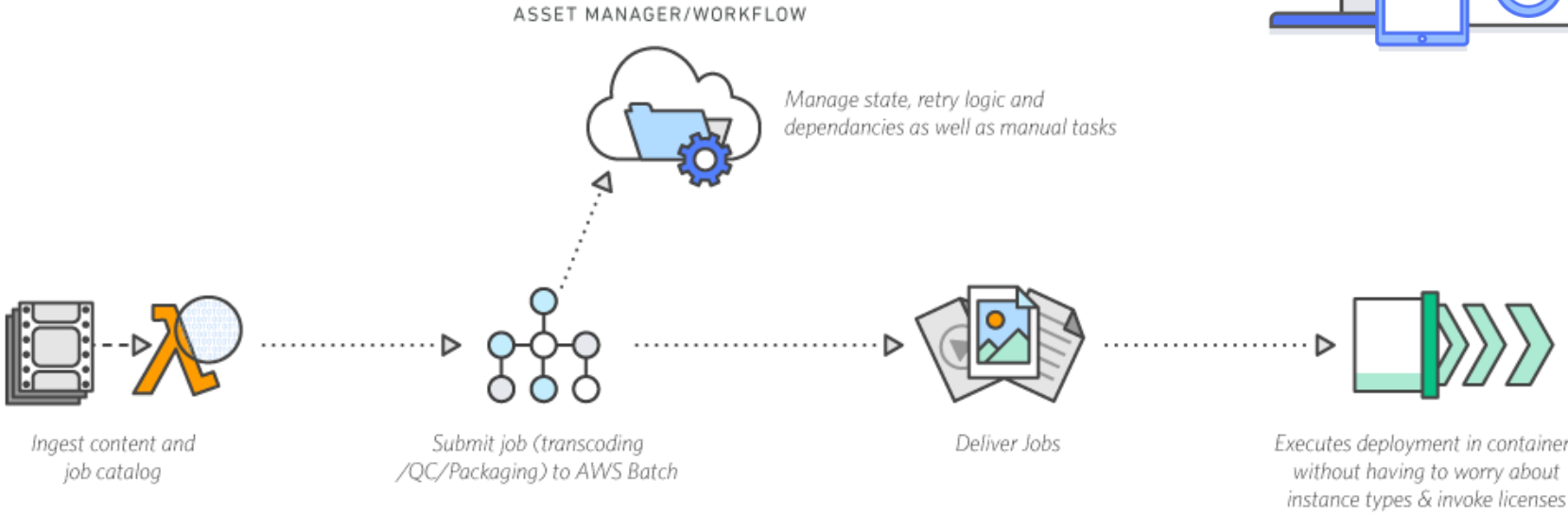


Computational Chemistry

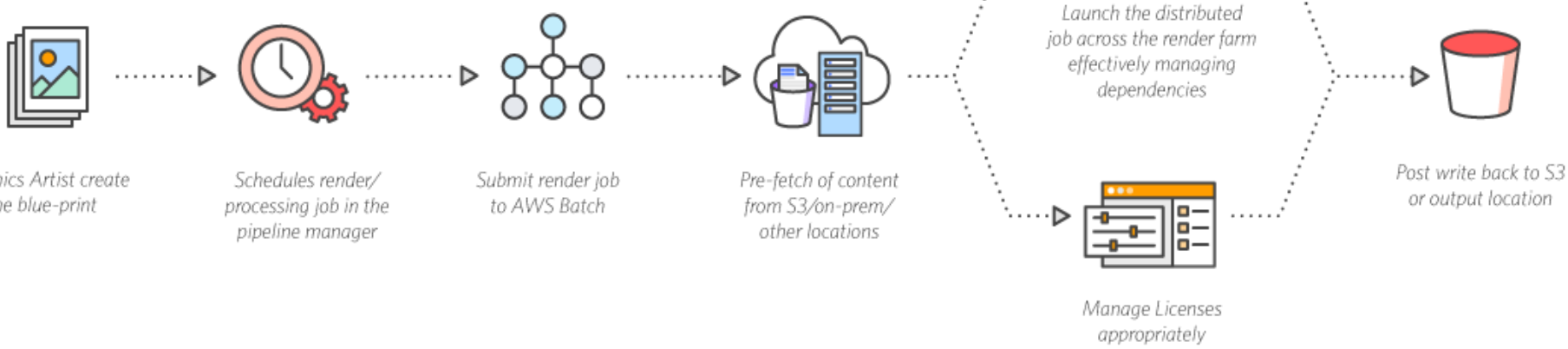




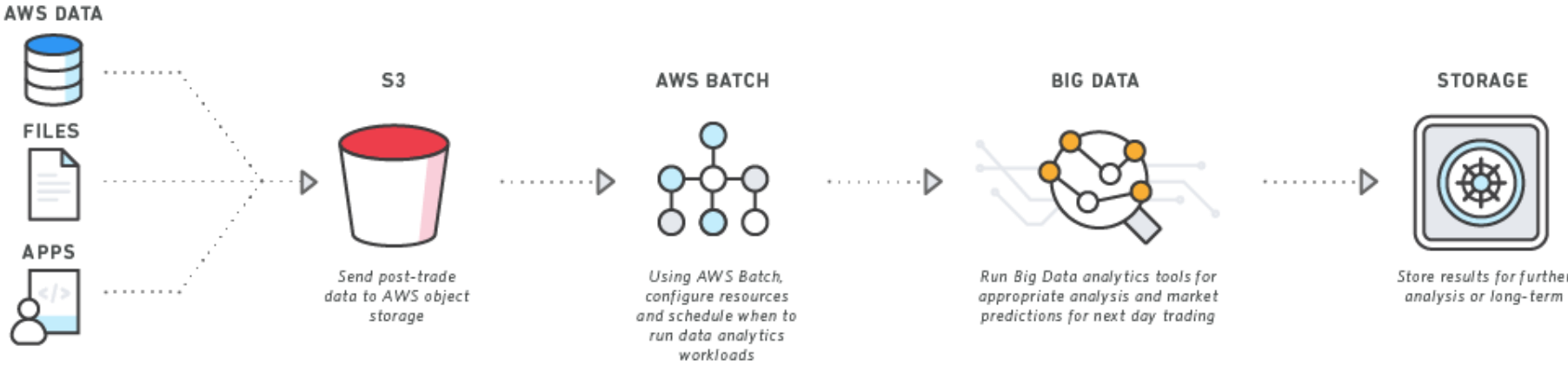
Media Encoding/Transcoding



Animation and Visual Effects Rendering



Financial Trading Analytics



Would you like to see a demo?

Fully Managed

Integrated with AWS

Cost-optimized Resource Provisioning

Any questions?

Thank You!

- Get started
  - With Amazon EC2 Spot Instances
  - With AWS Batch

- Demos
  - https://github.com/awslabs/ec2-spot-labs/tree/master/sqs-ec2-spot-fleet-autoscaling
  - https://github.com/awslabs/ec2-spot-labs/tree/master/ec2-spot-aws-batch