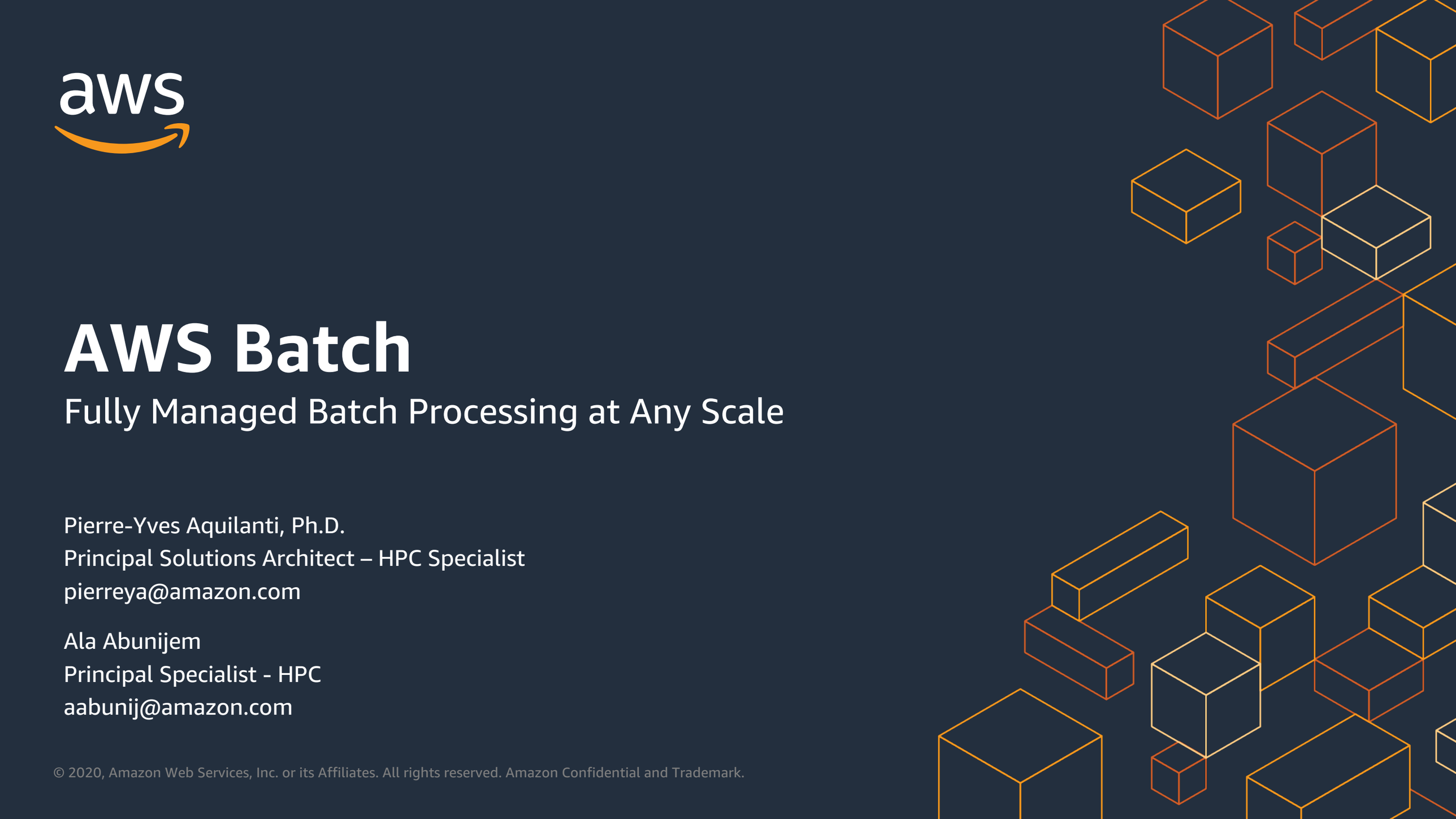


AWS Batch

Fully Managed Batch Processing at Any Scale

Pierre-Yves Aquilanti, Ph.D.
Principal Solutions Architect – HPC Specialist
pierreya@amazon.com

Ala Abunijem
Principal Specialist - HPC
aabunij@amazon.com

What is AWS Batch?

Job Scheduler

Orchestrator

Why AWS Batch?

Fully Managed

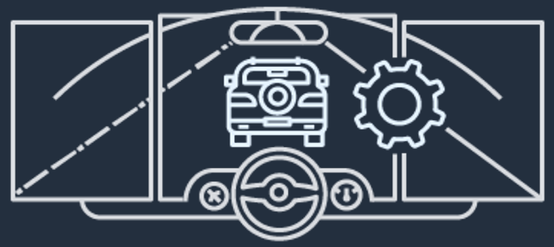
Integrated with AWS Services

Optimized Resource Provisioning

Cost Efficient

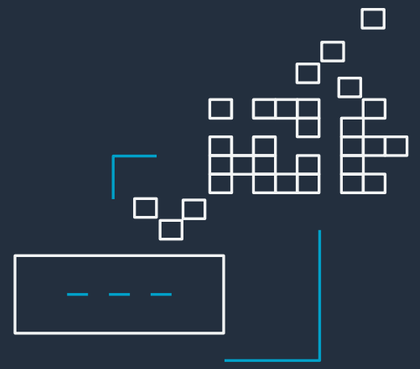
aws

Who Uses AWS Batch?



Autonomous vehicle
ML and simulation

Gene sequencing
& Drug Discovery



Big data

Machine Learning

Financial
risk analysis

EDA, CAD, FDC

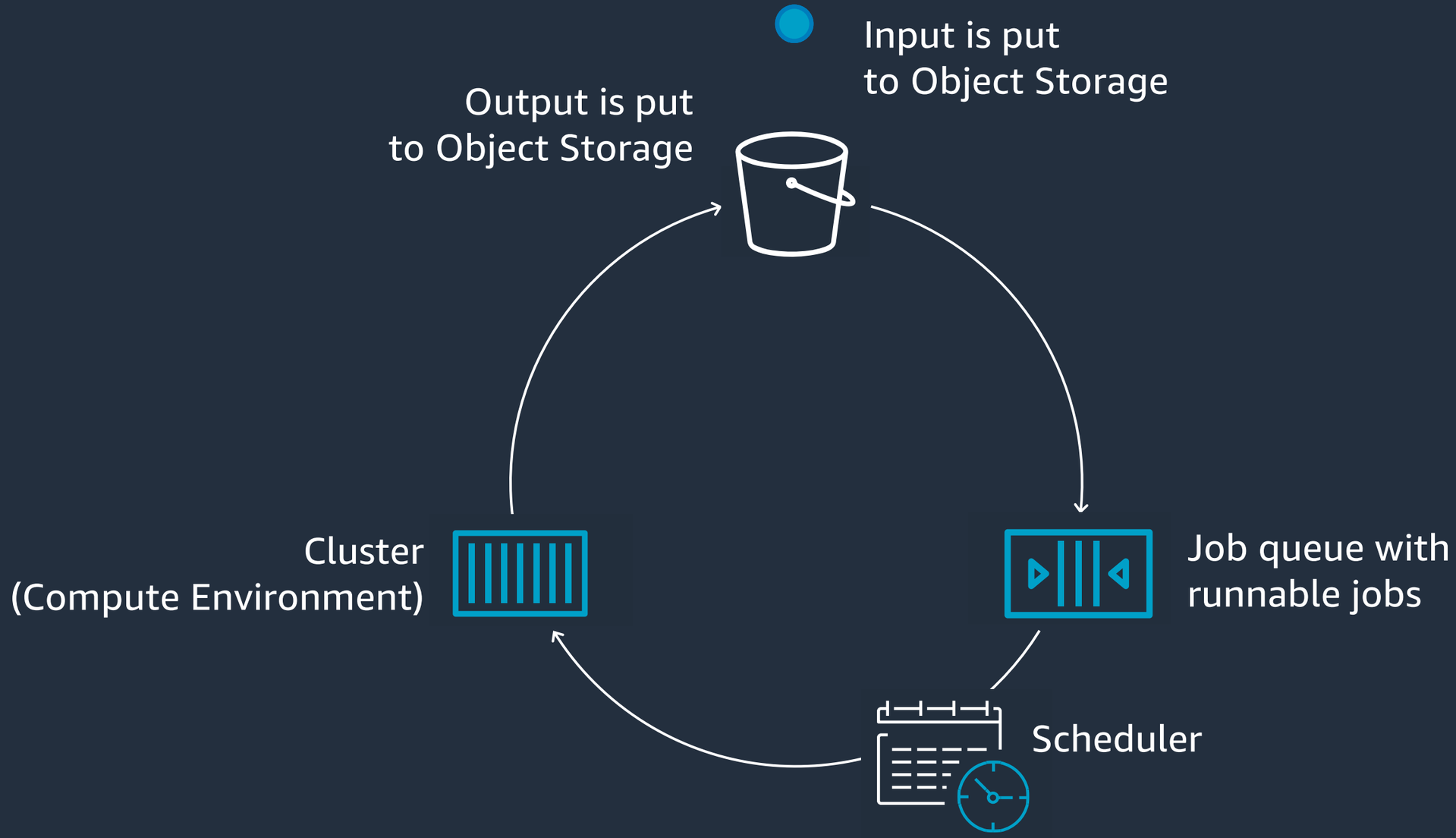
Renewable Energy,
Oil and gas exploration

Weather
simulation

aws

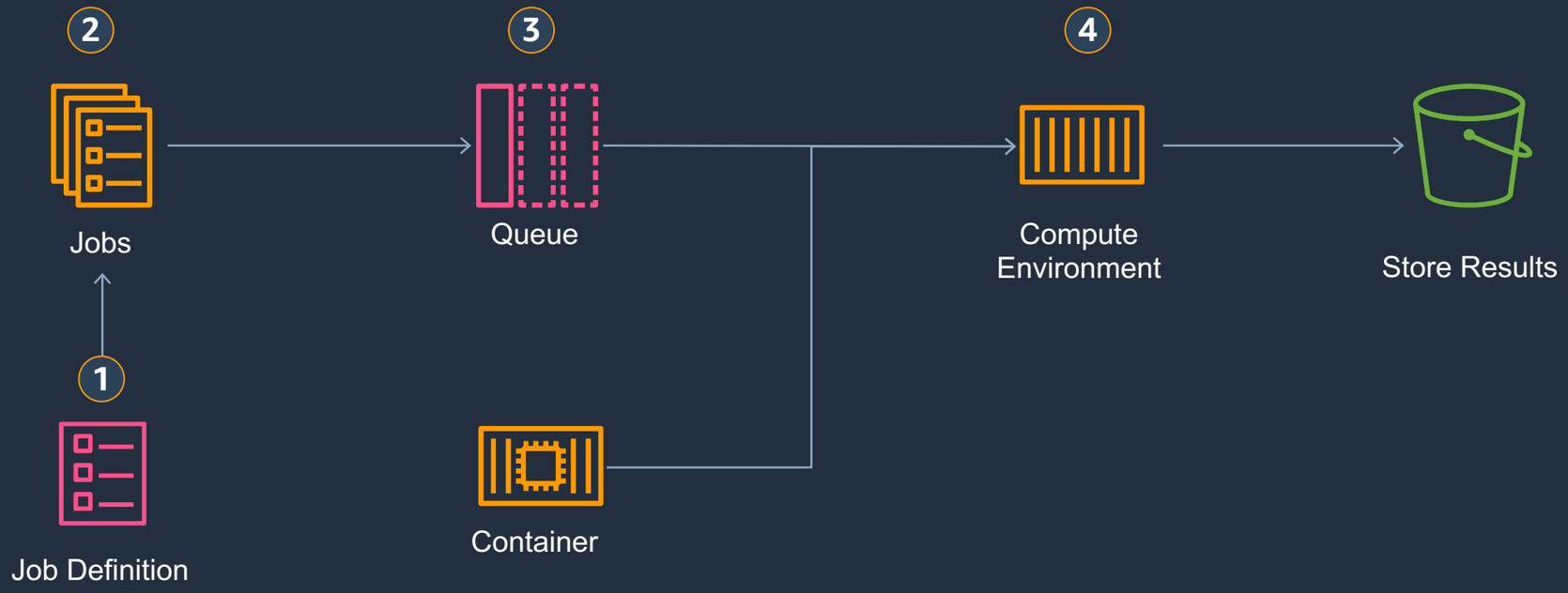
Batch Core Components

Batch Scheduling: Run Jobs Not Servers



AWS Batch overview

1. **Job Definition**
   Template that has common attributes (container image, IAM role, vCPU & memory requirements, ,…
2. **Job**
   Each job must reference a job definition, but many parameters may be overridden when submited
3. **Job Queue (JQ)**
   Queue determines priorities. Each JQ is connected to 1 or more CE
4. **Compute Environment (CE)**
   Resource Mix (defines On-demand vs. Spot and instance types. CE can be connected to more than one JQ



Job Definitions

AWS Batch **job definitions** specify how jobs are to be run.

Some attributes in a job definition:

- Container Image ← Amazon ECR, DockerHub, private registry or regular storage
- IAM role associated with the job ← Actions permitted/forbidden on services and resources
- vCPU and memory requirements ← Memory, swap memory, shared memory
- Volumes ← Mount points, docker volumes, tmpfs
- Environment variables ← Shell variable transmitted to the job (parameters)
- Retry strategy ← # of retries in case of failure, custom retries New

Job definitions are templates, parameters can be overridden

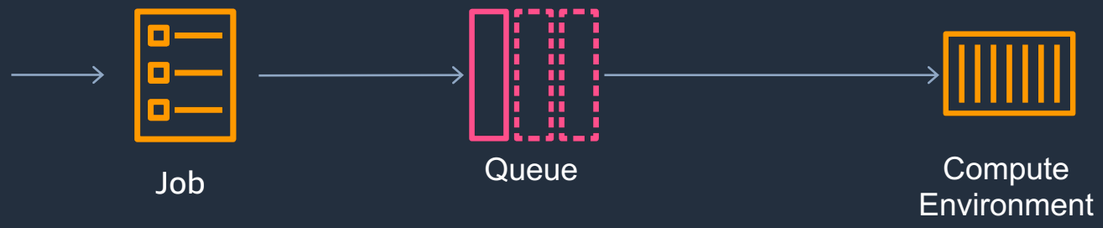
https://docs.aws.amazon.com/batch/latest/APIReference/API_RegisterJobDefinition.html

aws

Jobs

A job is the unit of work that will be processed by Amazon EC2 through AWS Batch.

Parameters through Job Definition or defined at submission time. Instances are selected based on CPU, Mem*, GPU.



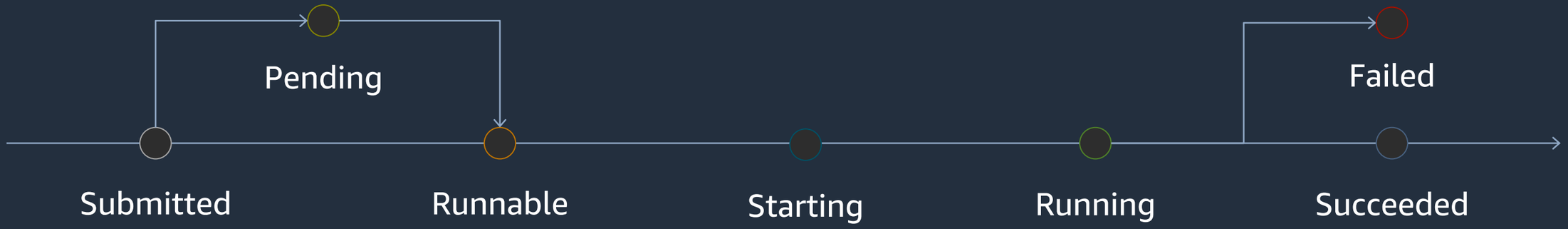
- Types of jobs:
  - Atomic: 1 or multiple jobs
  - Array: group of jobs with shared parameters (max 10k child jobs)
  - Multi-Node Parallel (MNP): MPI or NCCL
- Job dependencies: wait for a another job to complete

```
"dependsOn": [
      {
         "jobId": "IDJobA",
         "type": "SEQUENTIAL|N_TO_N"
      }
]
```

Expressing dependencies

* 32MB of memory reserved with ECS_RESERVED_MEMORY
https://docs.aws.amazon.com/batch/latest/userguide/memory-management.html#ecs-reserved-memory
https://docs.aws.amazon.com/batch/latest/userguide/job_definitions.html

Jobs states

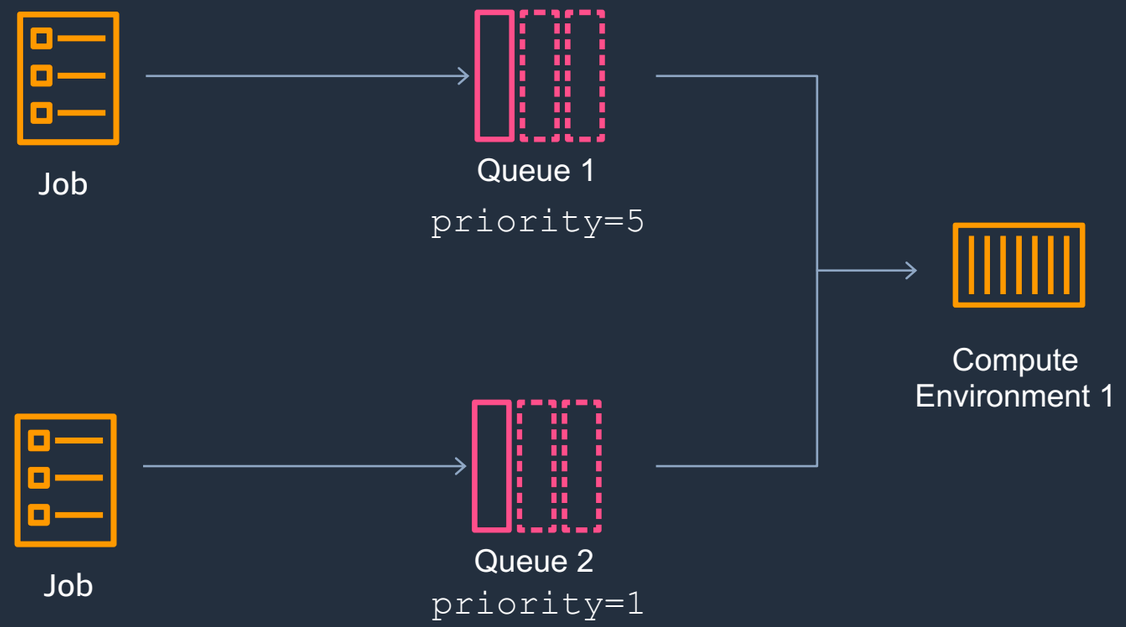


- SUBMITTED: accepted into the queue, but not yet evaluated by the scheduler for execution
- PENDING: the job has dependencies on other jobs which have not yet completed
- RUNNABLE: the job is evaluated by the scheduler and is ready to run
- STARTING: the job is in the process of being scheduled to a compute resource
- RUNNING: the job is currently running
- SUCCEEDED: the job has finished with exit code 0
- FAILED: the job finished with a non-zero exit code, was cancelled or terminated

Job Queues (JQ)

Job Queues are were jobs are submitted and reside throughout their lifetime

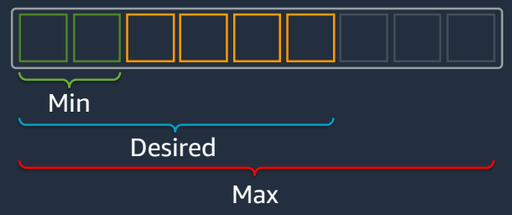
- A single queue can connect to 1 or a set of Compute Environments (CEs) and can share a CE with other queues
- Some parameters
  - Priority: scheduling priority to assign a job to a CE shared with multiple JQs in ascending order
  - CE Order: placement in descending order (0 first)



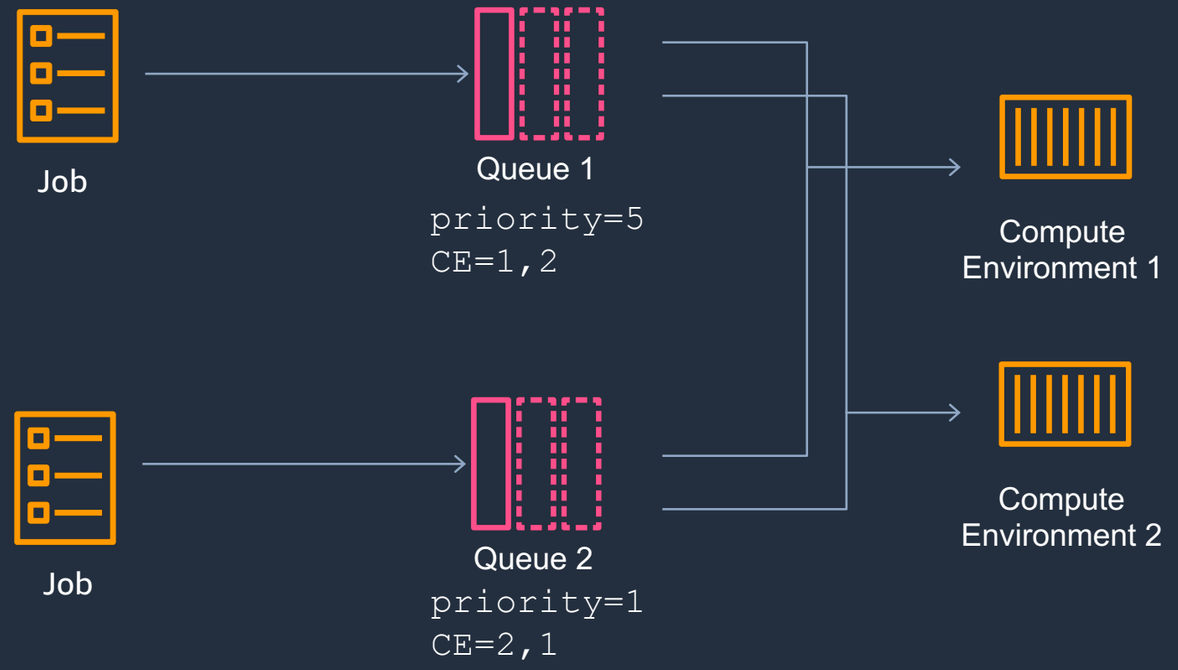
Compute Environments (CEs)

Compute Environments contain the underlying resources that are used to run jobs

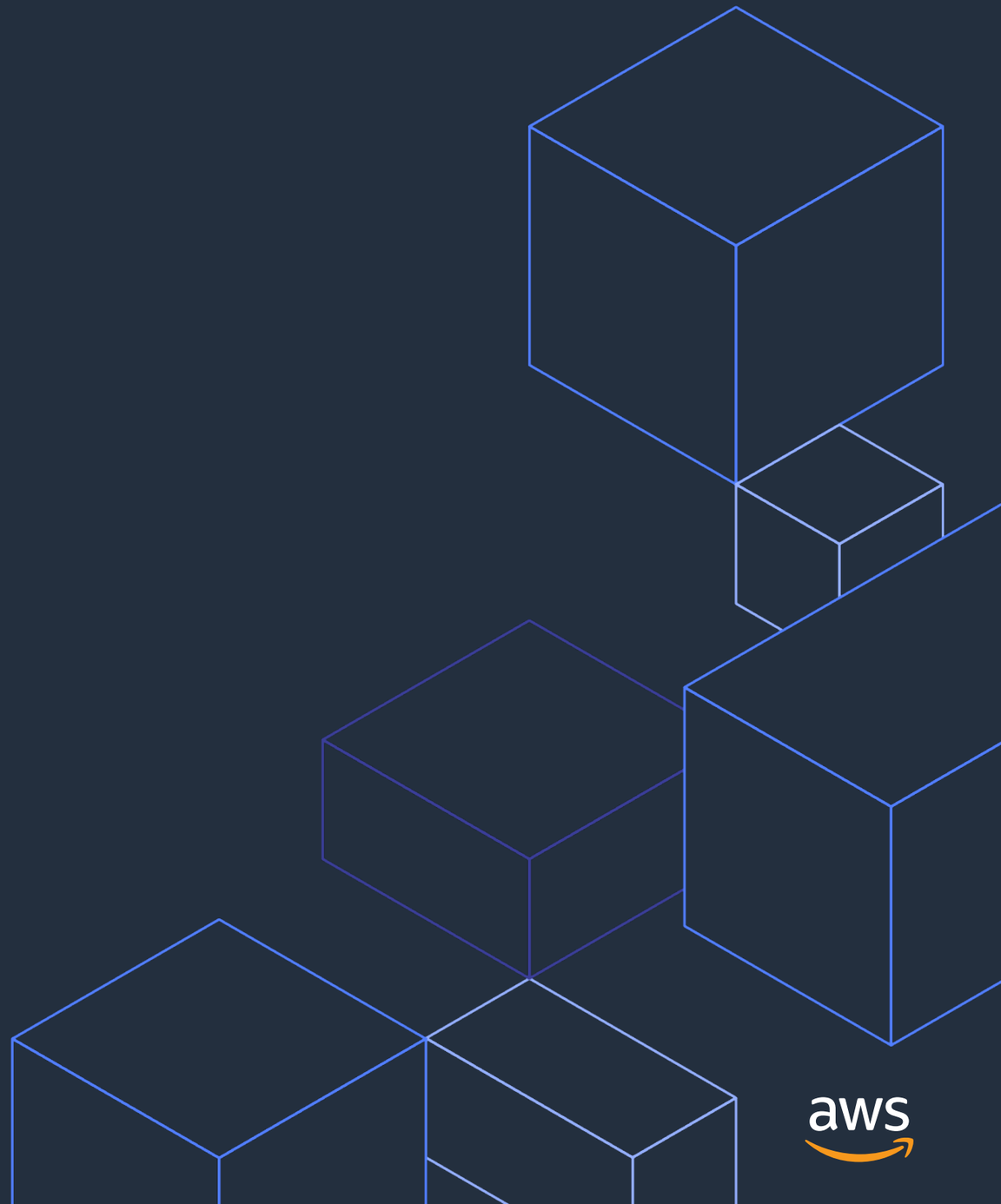
- Types
  - Managed: AWS scales and configures underlying instances (recommended)
  - Unamanaged: Customers control and manage instance configuration, provisioning and scaling
- Parameters
  - Scaling:



  - Instances types: instance families or specific instances on which jobs will be running



Job Dependencies & Workflows

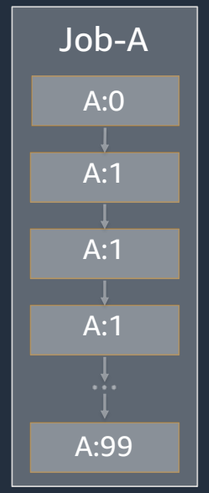


Workflows

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

Dependency on one job:

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

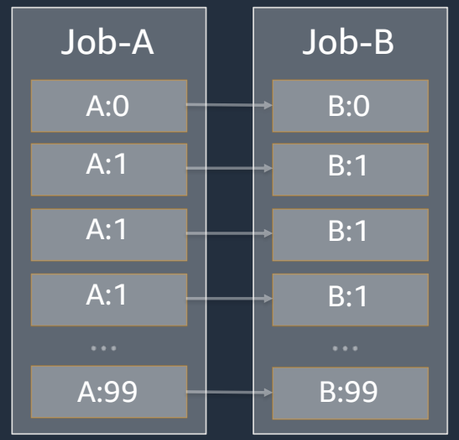


Sequential dependency within an array job:

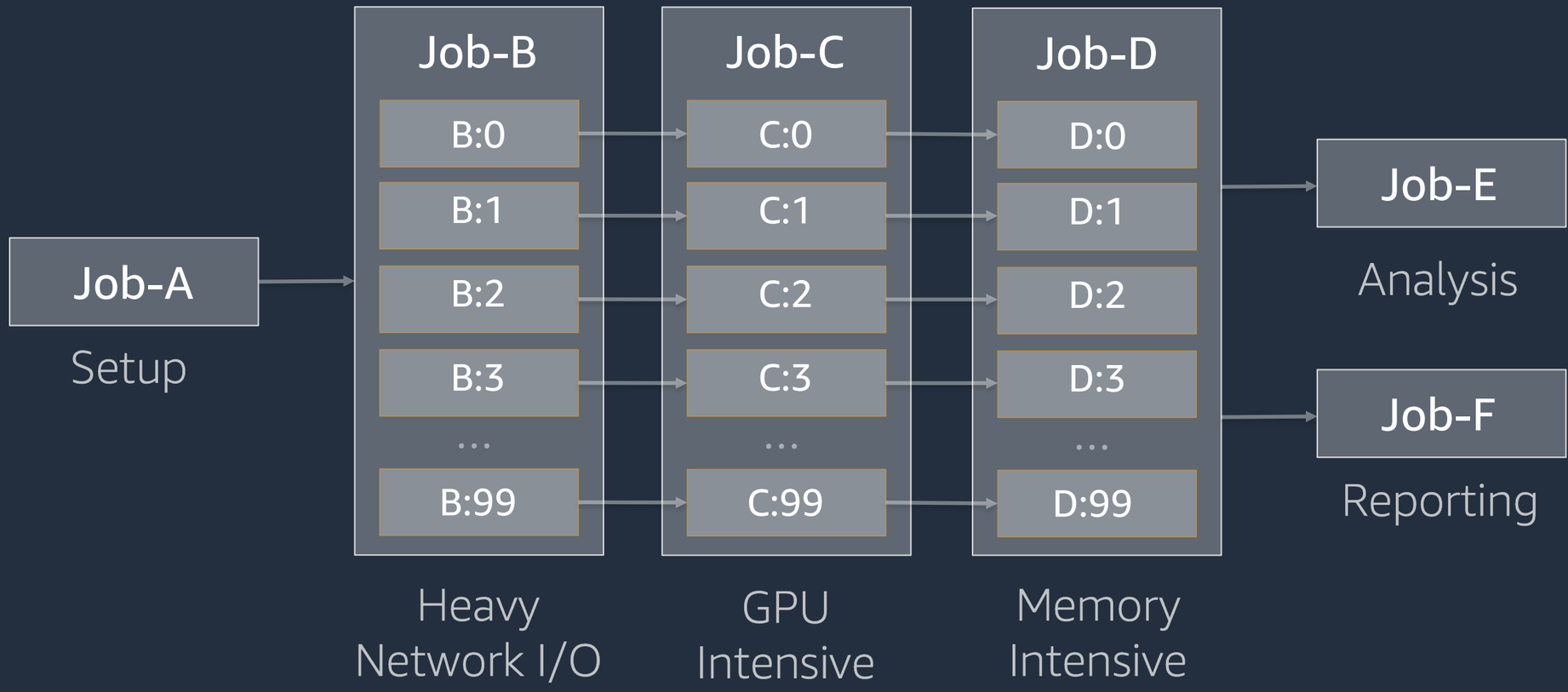
```
$ aws batch submit-job --array-properties size=100 \
    --depends-on type=SEQUENTIAL
```

Sequential dependency within an array job:

```
$ aws batch submit-job --array-properties size=100 \
    --depends-on jobId=7a6225f0-a16e-4241-9103-192c0c68124c,type=N_TO_N
```



Workflow Example



Other Workflow Management Options

AWS Step Functions

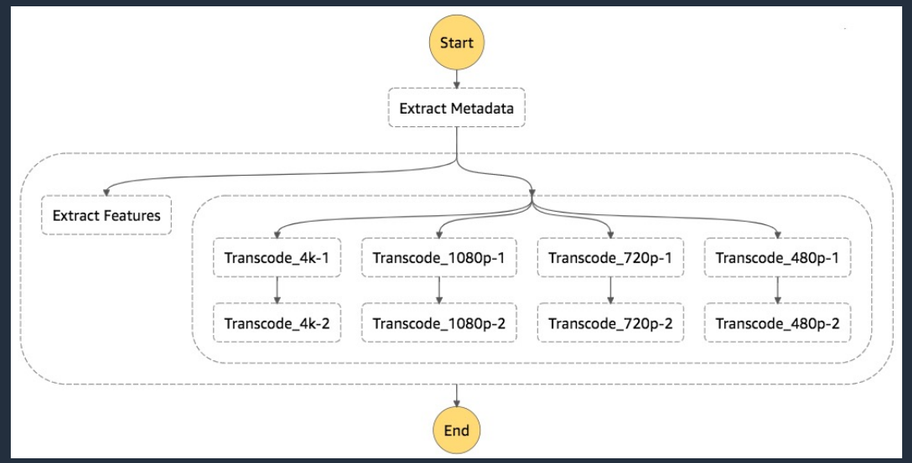
Workflow orchestrator to run a series of discrete steps described by a state machine.

Tasks

Parallel steps

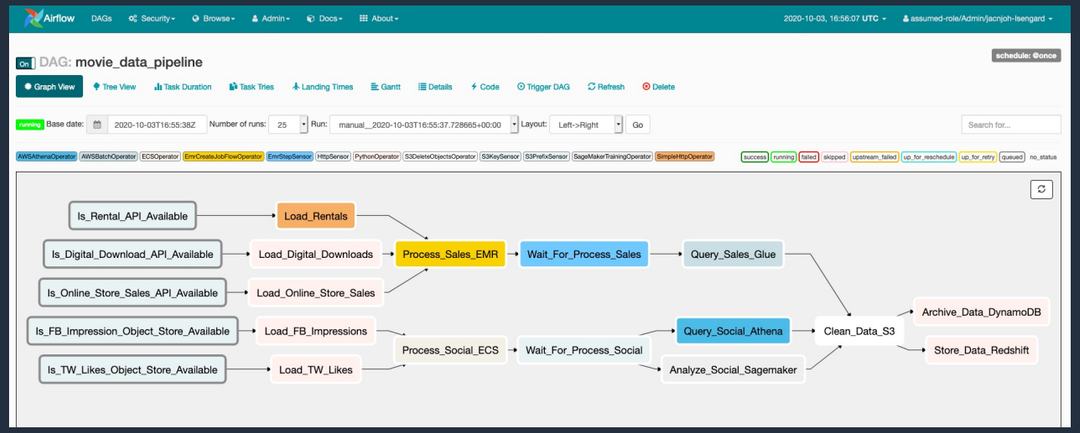
Branching

Callback wait



Amazon Managed Workflows for Apache Airflow

Managed workflow orchestration service for Apache Airflow to setup and operate pipelines.



AWS Batch Backends

Fargate vs. EC2 with Batch

EC2

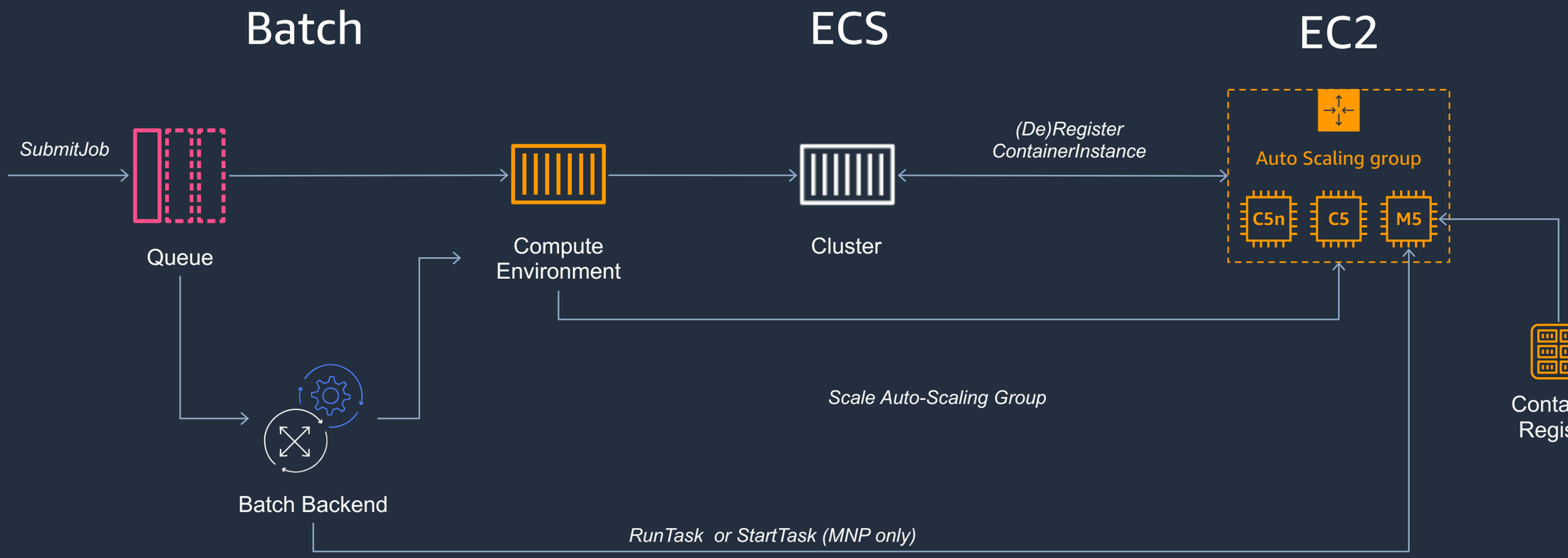
- Large workloads that require high scalability (> 1M +)
- Higher job throughput
- Customizable Compute Environments
- Somewhat serverless
- Can launch any size instance

Fargate

- Small jobs that require quick response time
- Limited job throughput
- Easy to use with limited expertise with AWS and AWS Batch
- Pure serverless environment
- Limited to smaller instances < 12/16 vCPUs (Q4/2021)

aws

Compute high level structure



Allocation strategies

- `BEST_FIT` (default in the CLI)

  Pick the least number of instances that can fit the jobs requirements at the lowest cost regardless of the type and size within your selection of instances. Will diversify across instance families.

- `BEST_FIT_PROGRESSIVE` (default for OD in the console)

  Same as best fit but will select instances of the same family in priority, then look at other families if $/vCPUs & requirements cannot be met.

- `SPOT_CAPACITY_OPTIMIZED` (default for Spot in the console)

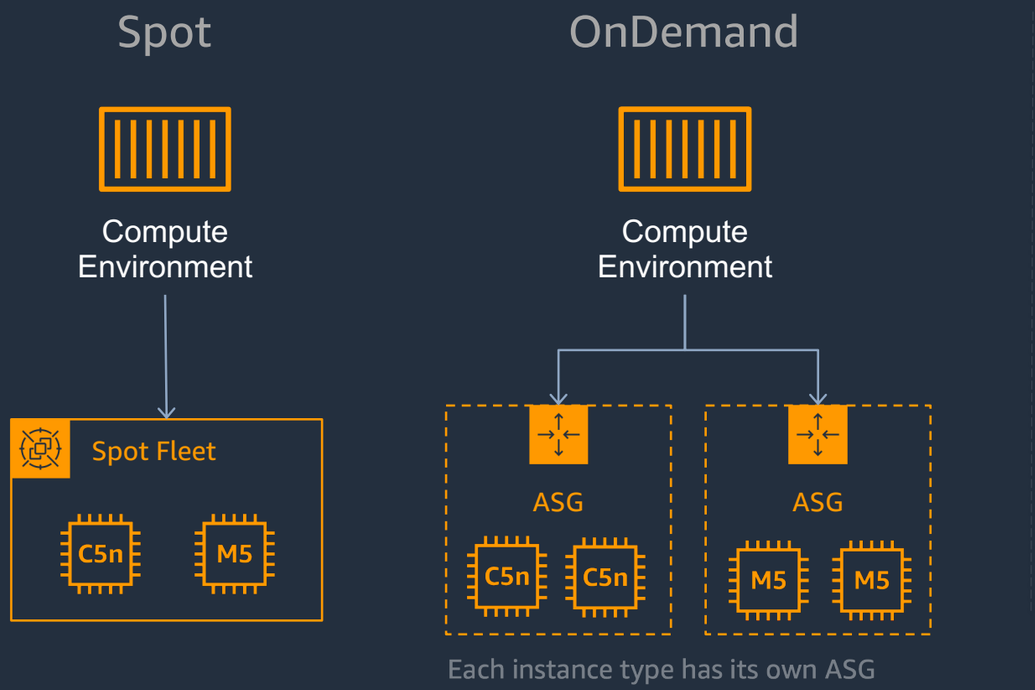
  Pick instances in chosen families and focus on pools with lower chances of interruptions based on historical data.

https://aws.amazon.com/blogs/compute/optimizing-for-cost-availability-and-throughput-by-selecting-your-aws-batch-allocation-strategy/

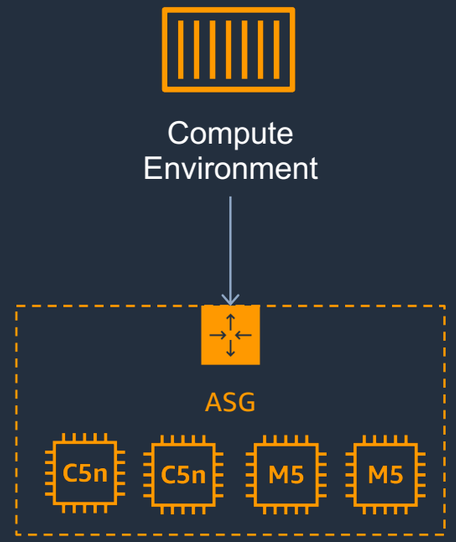
aws

Allocation strategies underneath

BEST_FIT



BEST_FIT_PROGRESSIVE
SPOT_CAPACITY_OPTIMIZED



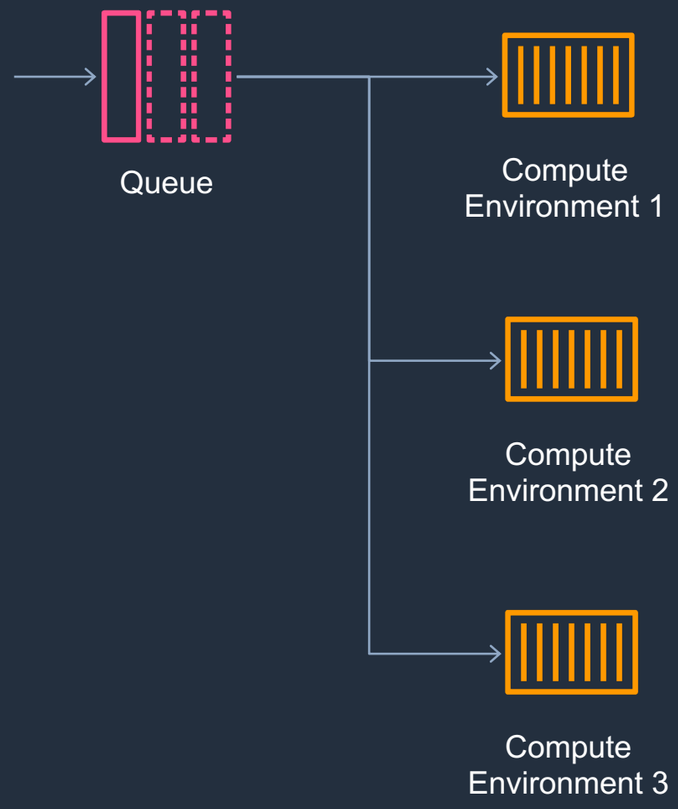
How AWS Batch scales

How AWS Batch scales CEs

- When is scaling triggered
  - The first time a job is submitted to a queue
  - Every 2 minutes
  - When a MNP job is submitted
  - User calls Create/Update/Delete CE
  - Backend action

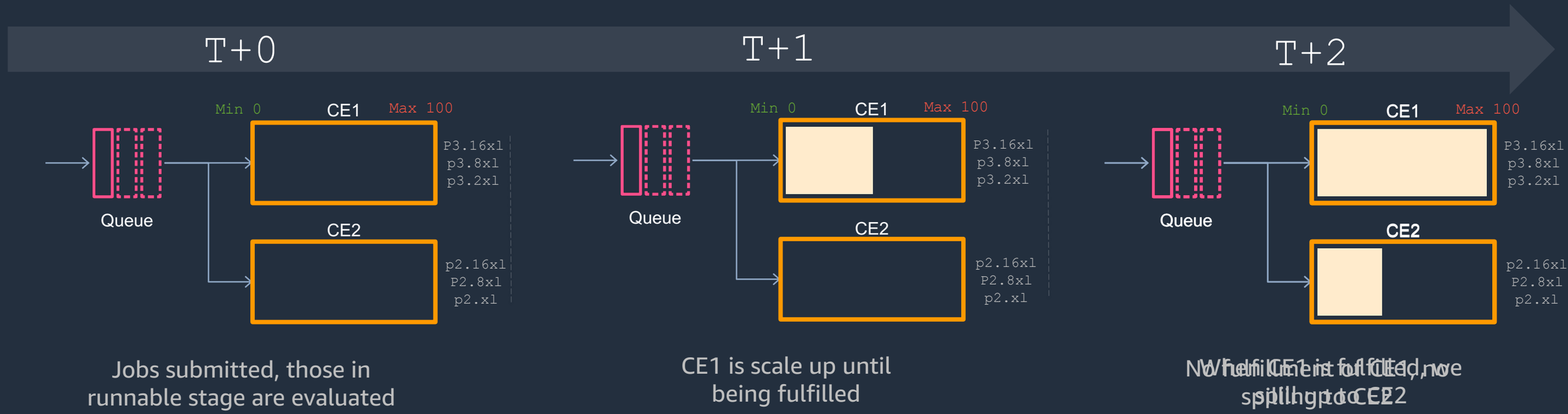
- How is scale up is conducted
  1. For each queue
  2. Consolidate view runnable jobs by properties
  3. Pack jobs in resources chunks to maximize vCPUs packing
  4. Select instances type(s) by lowest $ and vCPU/Memory/GPU packing, use larger instances if possible
  5. Provide list of instances ordered by $ to the ASG

- For scale down, AWS Batch explicitly terminate instances, it is not done through the ASG



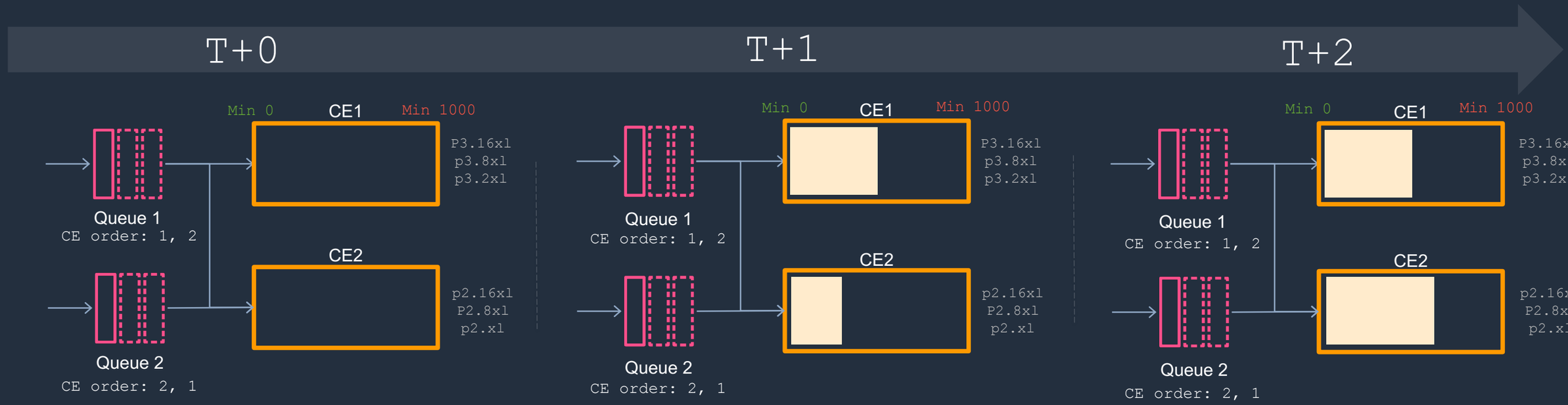
CEs in-order scaling example

- CEs are scaled in-order by the JQ
- Switching to the next CE occurs when *maxvCPU* is met in the current CE
- CEs can be scaled concurrently to meet capacity requirements



CE / JQ Interleaving technique

- This technique helps for large capacity acquisition and fast scale-up
- All JQs attached to each CEs (jobs can be schedules on each CE)
- JQ to CE order defined in a rolling fashion, each JQ scales a CE



Jobs are distributed to both queues at the same time

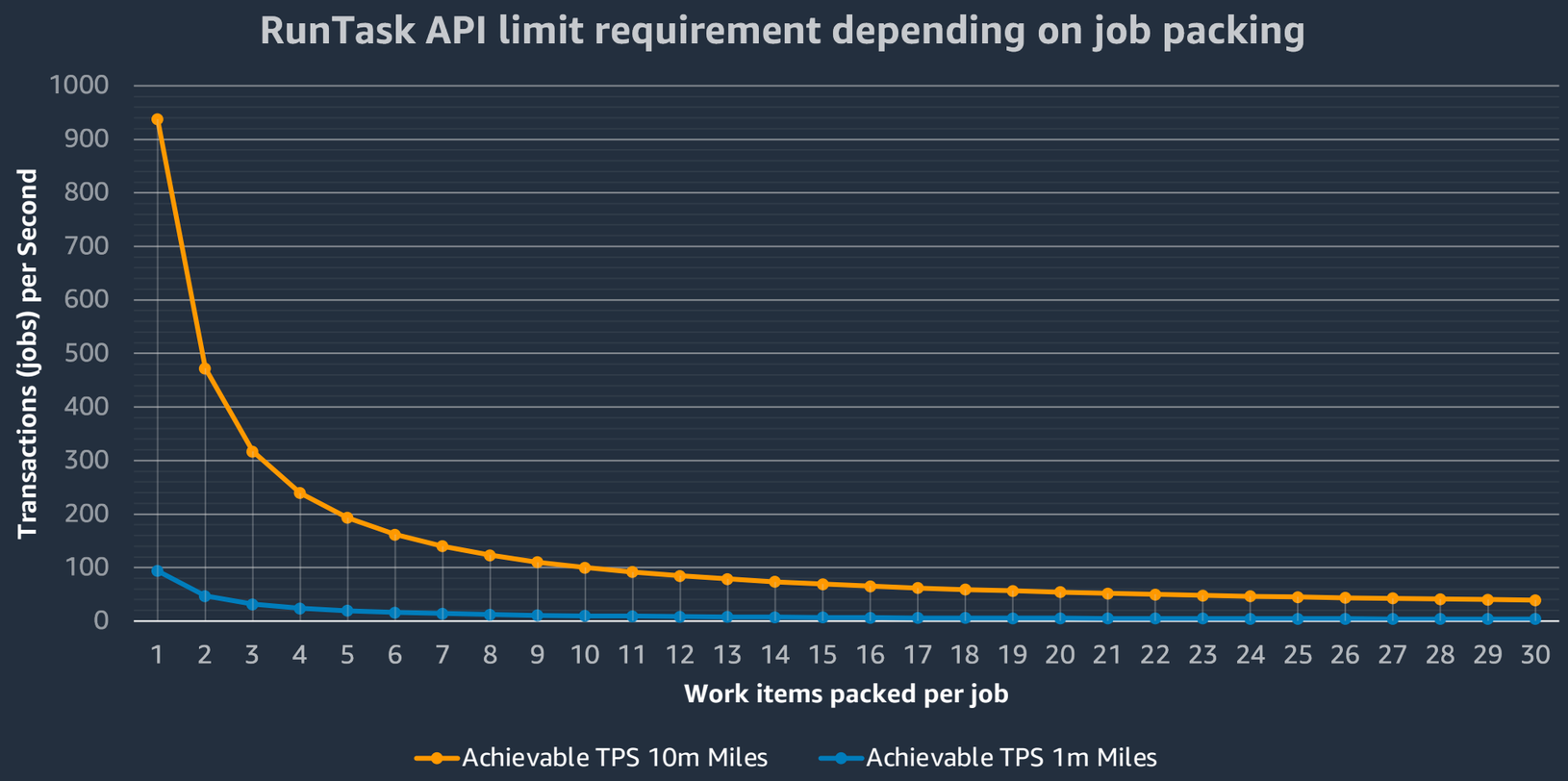
Each queue scale the CEs in order, order varies by queue

Jobs can be scheduled on both CEs to be executed

Optimizations for Job Throughput

Job packing

- Short jobs (<30 sec) are not the best match for AWS Batch
  - Pressure on the scheduler
  - RunTask API limit may not be enough even if increased
- Alternative is to job-pack
  - Dynamic setting
  - Help to balance throughput with RunTask API limits

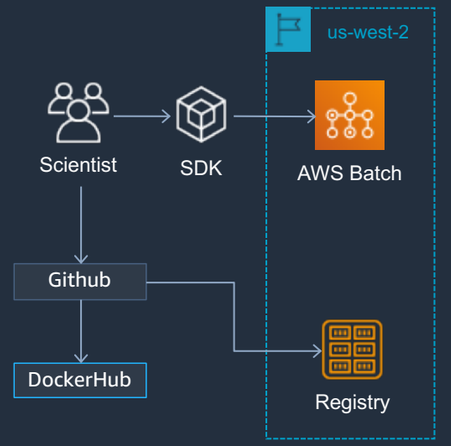


Containers structure and storage

```
ubuntu@ip-172-31-79-173:~$ docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
nvc                                 1.15.1              6ddfe8ed            12 days ago         32.9GB
ubuntu@ip-172-31-79-173:~$ docker history 6ddfe8edbeb3
IMAGE               CREATED             CREATED BY                                      SIZE
6ddfe8edbeb3        12 days ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENTRYPOINT ["/scripts/ent…   0B
<missing>           12 days ago         |3 archive_dir=D driv ve…                      77B
<missing>           12 days ago         /bin/sh -c #(nop) WORKDIR /d                    0B
<missing>           12 days ago         /bin/sh -c #(nop)  USER driveconst              0B
<missing>           12 days ago         /bin/sh -c #(nop) COPY file:741dcfa00aff76a0…   2.16kB
<missing>           12 days ago         |3 archive_dir=D driv ve…                      152MB
<missing>           12 days ago         |3 archive_dir=D driv ve…                      205MB
<missing>           12 days ago         |3 archive_dir=D driv ve…                      530MB
<missing>           12 days ago         /bin/sh -c #(nop)  ARG                          0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV DDS_CONFIG=/d …          0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV DDS_DOMAIN=1066          0B
<missing>           12 days ago         |2 archive_dir=D 008 drivesim_ve…              0B
<missing>           12 days ago         |2 archive_dir=D 008 drivesim_ve…              887B
<missing>           12 days ago         |2 archive_dir=D 008 drivesim_ve…              0B
<missing>           12 days ago         /bin/sh -c #(nop) COPY dir:c874f713e6155ef38…   26.8GB
<missing>           12 days ago         /bin/sh -c #(nop)  ARG archive_dir              0B
<missing>           12 days ago         |1 ion=1.15.1-internal /bin/sh …               2.11kB
<missing>           12 days ago         |1 ion=1.15.1-internal /bin/sh …               784B
<missing>           12 days ago         |1 ion=1.15.1-internal /bin/sh …               2.33kB
<missing>           12 days ago         |1 ion=1.15.1-internal /bin/sh …               336kB
<missing>           12 days ago         /bin/sh -c #(nop)  ENV HOME=/c                  0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV GID=1000                 0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV UID=1000                 0B
<missing>           12 days ago         /bin/sh -c #(nop)  ENV USER=d                   0B
<missing>           12 days ago         /bin/sh -c #(nop)  LABEL organization=          0B
<missing>           12 days ago         /bin/sh -c #(nop)  ARG sion                     0B
<missing>           2 months ago        /bin/sh -c echo "                               9.03kB
<missing>           2 months ago        /bin/sh -c echo "PS1=\"\\[\\033[01;31m\\][\…   3.22kB
<missing>           2 months ago        /bin/sh -c echo "PS1=\"\\[\\033[01;33m\\][\…   2.31kB
<missing>           2 months ago        /bin/sh -c #(nop) COPY file:ecccda9e8f2129f1…   139B
<missing>           2 months ago        /bin/sh -c #(nop)  ENV _CAPABIL…                0B
<missing>           2 months ago        /bin/sh -c #(nop) COPY file:052cbddc3360723a…   114B
<missing>           2 months ago        /bin/sh -c #(nop)  ENV LD_LIBRARY_PATH=/usr/…   0B
<missing>           2 months ago        /bin/sh -c touch /tmp/temp.cpp &&    emcc /t…   7.75MB
<missing>           2 months ago        /bin/sh -c #(nop)  ENV EM_NODE_JS=/3rd/emsdk…   0B
```

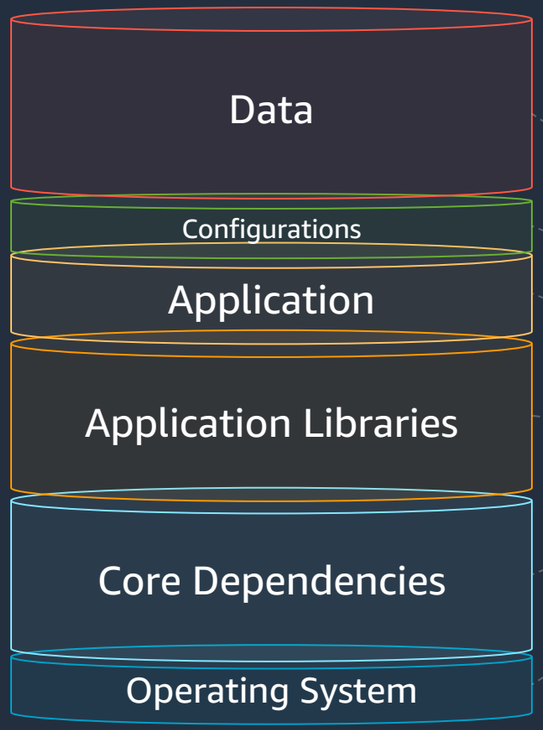
Layers created on RUN, COPY and ADD

- Prefer ECR or S3 for Batch
  - DockerHub throttles under load, private registries can suffer



- Fewer even layers is better
  - Docker requests layers in parallel (1 layer = 1 request)
  - Even layers provide a better load distribution

Using machine images and containers



Data Type	Size	Change Frequency
Input Data	20 GB (r+w)	Runtime — 20 min per job
Configurations	3 MB	Container — Weekly
Application	1 GB	Container — 5 min
Application Libraries	4 GB	machine images — Weekly
Core Dependencies	5 GB	machine images — Biweekly
Operating System	500 MB	machine images — Monthly

aws

Debug and data capture

Debugging Batch

- Job in runnable status (#1 error)

  log driver if using a non-ecs, insufficient resources (job shape large), NAT, EC2 limits

- Debugs steps to follow

  1. AWS Batch dashboard
     → state of the jobs is expected, do the CE scales?
  2. EC2 Panel – EC2 instances
     → are instances booting?
  3. EC2 Panel – Auto-scaling Groups
     → check history, can you acquire instances?
  4. ECS Cluster
     → Can instances register?
  5. CloudTrail
     → Look at ECS, then EC2
  6. Connect to the instances (ssh or else)
     → check the ECS + Docker logs, increase CE `minvCPU`

```
Pattern:
  detail-type:
  - 'AWS API Call via CloudTrail'
  source:
  - 'aws.ecs'
  detail:
    eventName:
    - 'RunTask'
    - 'RegisterContainerInstance'
    - 'DeregisterContainerInstance'
    eventSource:
    - 'ecs.amazonaws.com'
```

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/logs.html
https://docs.aws.amazon.com/batch/latest/userguide/troubleshooting.html#job_stuck_in_runnable

CloudWatch Events: Batch Jobs & EC2 Instances

- AWS Batch jobs: job transition between states

```
EventPattern:
  source:
      - "aws.batch"
  detail-type:
      - "Batch Job State Change"
```

```
# Transform the CloudWatch Event
item = {
    'JobId': event['detail']['jobId'],
    'States': {event['detail']['status']: event['time']},
    'Region': event['region'],
    'JobQueue': event['detail']['jobQueue'],
    'JobName': event['detail']['jobName'],
    'JobDefinition': event['detail']['jobDefinition'],
    'LastEventTime': event['time'],
    'LastEventType': event['detail']['status'],
    'ExpirationTime': int(time.time() + item_expiration_days)
}
```

- EC2 instances: life cycle & states

```
EventPattern:
  source:
      - "aws.ec2"
  detail-type:
      - "EC2 Instance State-change Notification"
```

```
item = {
    'InstanceId': event['detail']['instance-id'],
    'Events': {
        # remove dashes to make the entry valid
        event_formatted: event['time']
    },
    'Region': event['region'],
    'LastEventTime': event['time'],
    'LastEventType': event_formatted,
    'ExpirationTime': int(time.time() + item_expiration_days)
}
```

CloudTrail API Calls: ECS Container Instances & RunTask

- RunTask API Calls

```
Pattern:
  detail-type:
  - 'AWS API Call via CloudTrail'
  source:
  - 'aws.ecs'
  detail:
    eventName:
    - 'RunTask'
    eventSource:
    - 'ecs.amazonaws.com'
```

```
item = {
    'Region': detail['awsRegion'],
    'LastEventTime': detail['eventTime'],
    'LastEventType': detail['eventName'],
    'JobId': environment['AWS_BATCH_JOB_ID'],
    'JobName': environment['AWS_BATCH_JQ_NAME'],
    'CEName': environment['AWS_BATCH_CE_NAME'],
    'JobAttempt': environment['AWS_BATCH_JOB_ATTEMPT'],
    'ECSCluster': detail['requestParameters']['cluster'].split("_Batch")[0],
    'TaskDefinition': detail['requestParameters']['taskDefinition'],
    'ExpirationTime': int(time.time() + item_expiration_days),
    'Events': {detail['eventName']: detail['eventTime']}
}
```

- ECS instances registration & deregistration

```
Pattern:
  detail-type:
  - 'AWS API Call via CloudTrail'
  source:
  - 'aws.ecs'
  detail:
    eventName:
    - 'RegisterContainerInstance'
    - 'DeregisterContainerInstance'
    eventSource:
    - 'ecs.amazonaws.com'
```

```
# Transform the CloudTrail event
item = {
    'Region': detail['awsRegion'],
    'ECSCluster': detail['requestParameters']['cluster'].split("_Batch")[0],
    'InstanceId': detail['responseElements']['containerInstance']['ec2InstanceId'],
    'LastEventTime': detail['eventTime'],
    'LastEventType': detail['eventName'],
    'ContainerInstanceId': detail['responseElements']['containerInstance']['container
    'ExpirationTime': int(time.time() + item_expiration_days),
    'Events': {detail['eventName']: detail['eventTime']}
}
```

Thank You