
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

CMP329-R

Deep dive on using AWS Batch with container-based clusters

Ala Abunijem (he/him)

Principal HPC Go to Market Specialist

AWS

Angel Pizarro (he/him)

Principal Developer Advocate, AWS Batch

AWS

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

I. What is AWS Batch?

II. Why AWS Batch?

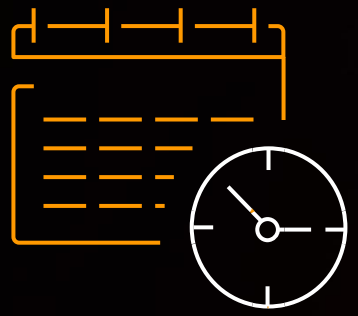
III. What’s new with AWS Batch

IV. Introducing fair-share scheduling

V. Tips to run at scale using AWS Batch

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is AWS Batch?



Job scheduler



Orchestrator

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Who uses AWS Batch?



Autonomous vehicle ML and simulation



Gene sequencing & drug discovery



Big data



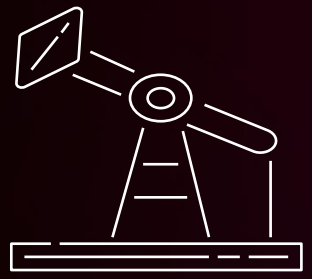
Machine learning



Financial risk analysis



EDA, CAD, FDC



Renewable energy, oil, and gas exploration



Weather simulation

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why AWS Batch?



Fully managed



Integrates with AWS services



Cloud native, optimizes resource provisioning



Cost efficient



High scalability

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Arm Accelerates Speed to Market by Migrating EDA Workflows to AWS

arm

Challenge

Arm wanted to modernize its offerings for intellectual property design because its on-premises infrastructure could not grow with the pace of its engineering requirements.

Solution

Arm uses AWS Batch and Amazon EC2 Spot Instances to optimize its compute—decreasing the turnaround time for verification jobs, increasing engineer productivity, and accelerating product speed to market.

Benefits

- Can run more than 53 million jobs per week
- Scaled up to 350,000 virtual CPUs
- Decreased turnaround time for verification jobs



> "Using AWS Batch facilitates selecting different instance types and mixing them together. That helps us to achieve the scalability that we need."
>
> —**Zhifeng Yun**, technical director, Arm Limited



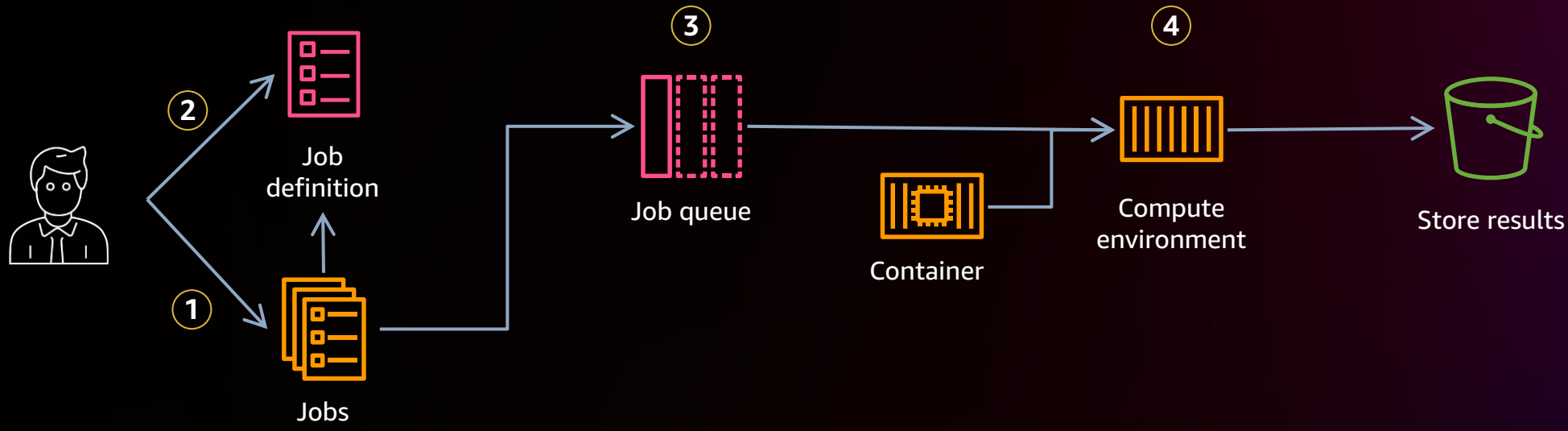
AWS Batch overview

1. **Job**
   Actual work request. Each job must reference a job definition, but many parameters may be overridden when submitted

2. **Job definition**
   Template that has common attributes (such as container image, IAM role, vCPU and memory requirements)

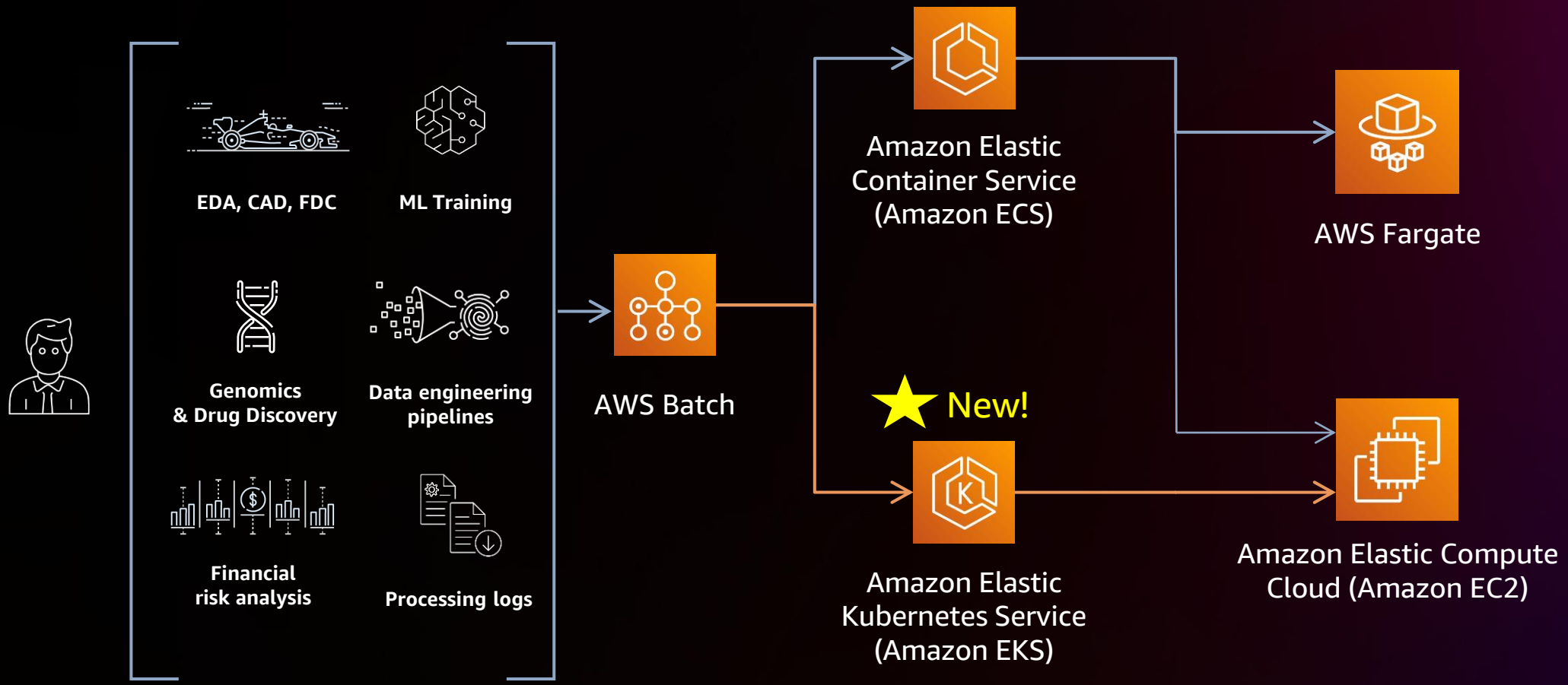
3. **Job queue (JQ)**
   Queue determines priorities. Each JQ is connected to one or more CE

4. **Compute environment (CE)**
   Resource mix (defines on-demand vs. Spot and other instance types); CE can be connected to more than one JQ



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Batch for Amazon Elastic Kubernetes Service (Amazon EKS)



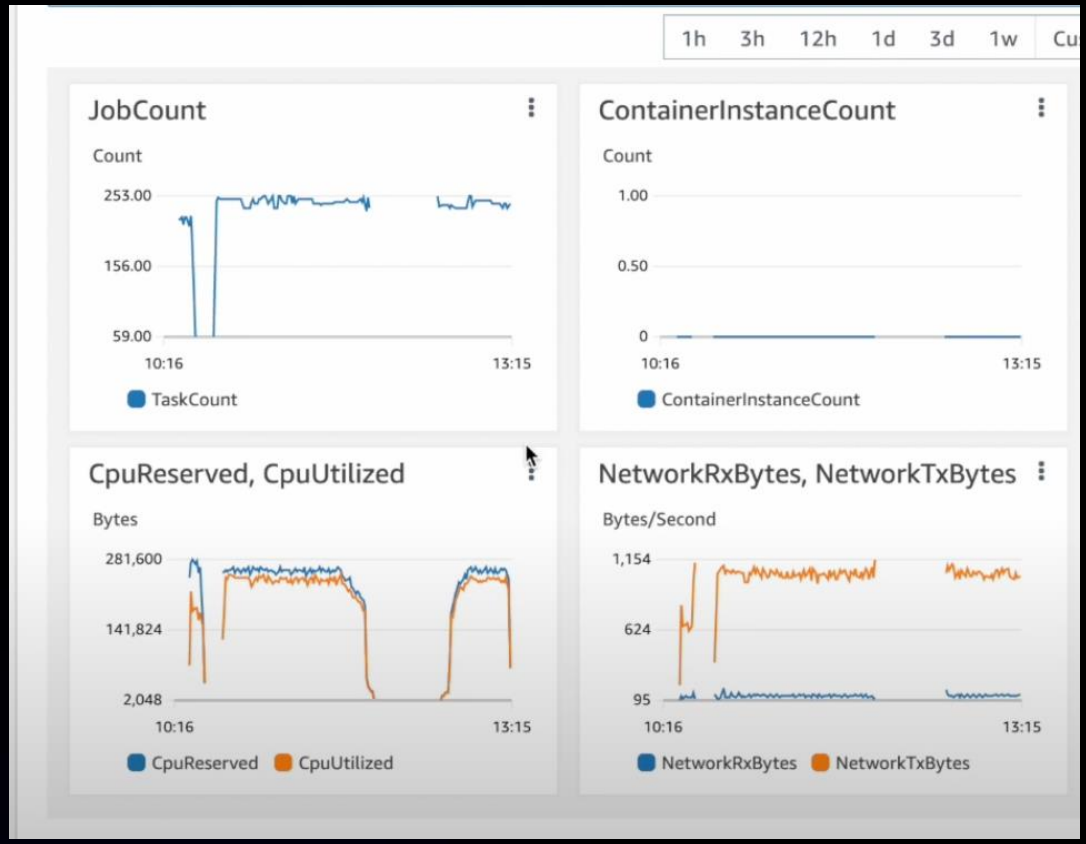
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What’s new with AWS Batch

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

New usability features

- Container Insights in the Console
- Dynamic updates to Compute Environments
- 7-day job history retention



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Batch provides EKS-specific API parameters

- EKS compute environments and job definitions have separate parameters from ECS
- AWS Batch supports EKS versions 1.21+ but 1.22+ recommended
- You can scope a DaemonSet to Batch nodes using tolerations

```
{
    "jobDefinitionName": "",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "serviceAccountName": "myBatchEksServiceAccount",
            "hostNetwork": true,
            "containers": [
                {
                    "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                    "command": [
                        "echo",
                        "Hello KubeCon!"
                    ],
                    "resources": {
```

```
tolerations:
- key: "batch.amazonaws.com/batch-node"
  operator: "Exists"
```

```
                    "securityContext": {}
                }
            ],
            "volumes": []
        }
    }
    "retryStrategy": {},
    "propagateTags": true,
    "timeout": {
        "attemptDurationSeconds": 0
    },
    "tags": {
        "KeyName": ""
    },
    "schedulingPriority": 0,
}
```

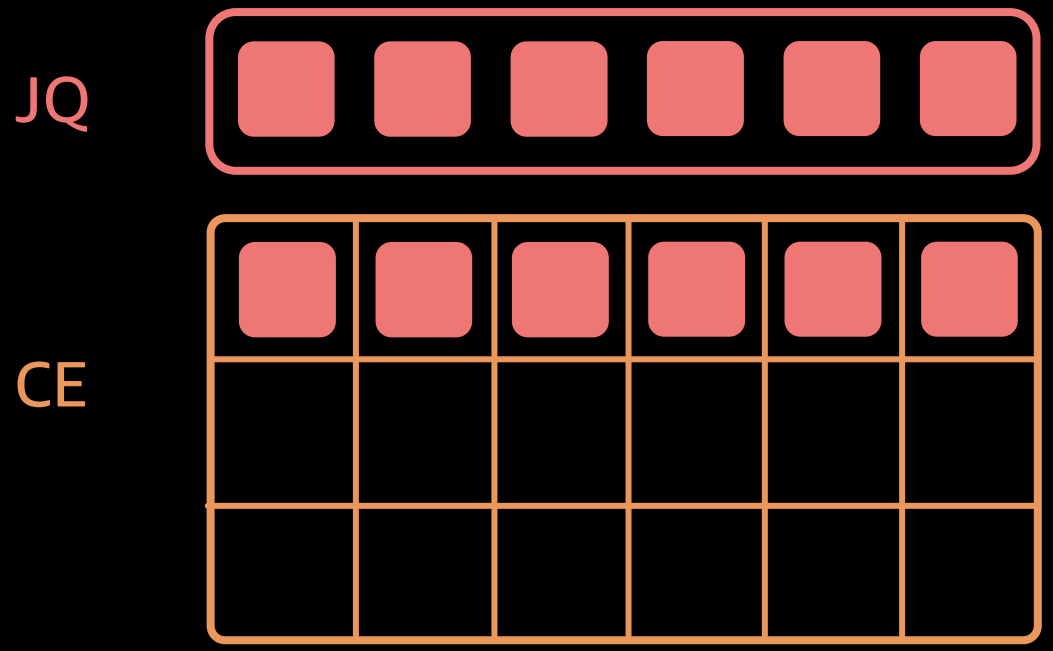
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deep dive into fair share scheduling

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fair share scheduling attributes

- `shareIdentifier`
  - A logical grouping for requests, like project, group, or department

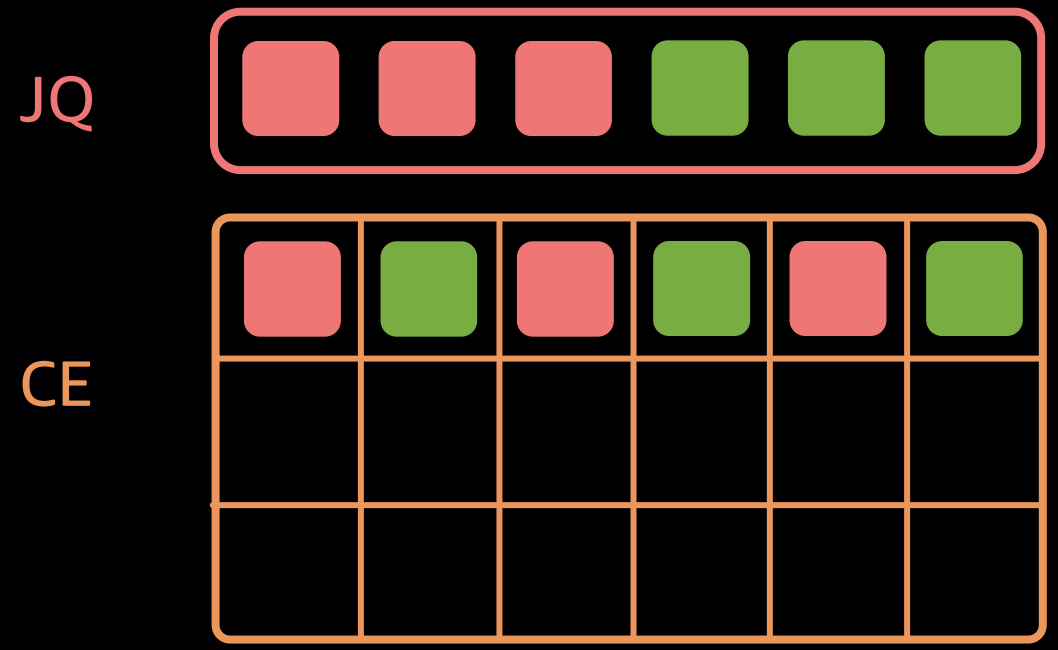



```
{
  "fairsharePolicy": {
    "shareDistribution": [
      {
        "shareIdentifier": "pink”
      }
    ]
  }
}
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fair share scheduling attributes

- `shareIdentifier`
  - A logical grouping for requests, like project, group, or department

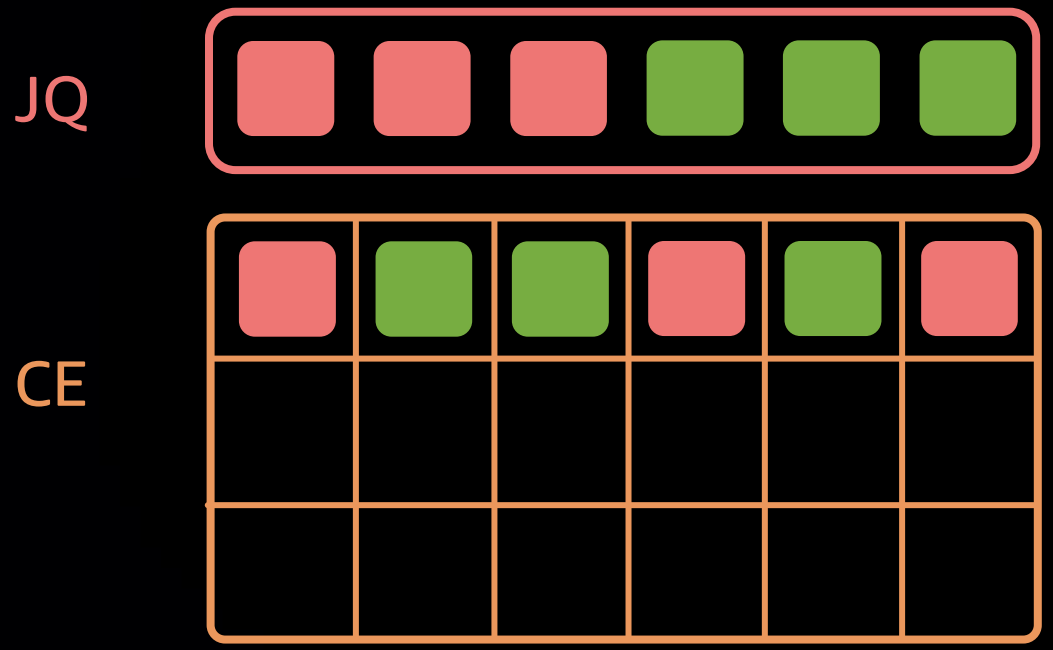



```
{
  "fairsharePolicy": {
    "shareDistribution": [
      {
        "shareIdentifier": "pink”
      },
      {
        "shareIdentifier": ”green”
      },
    ]
  }
}
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fair share scheduling attributes

- `shareIdentifier`
  - A logical grouping for requests, like project, group, or department
  - `weightFactor` will adjust priority within a share identifier, lower value gets more resources
  - `shareDecaySeconds` will adjust weights based on historical data

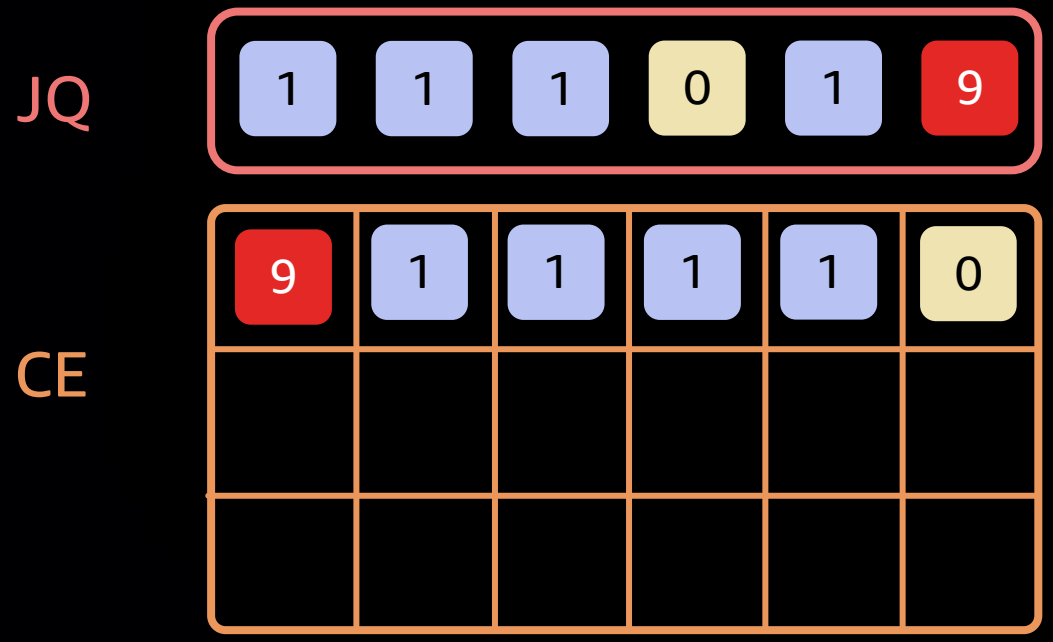



```
{
  "fairsharePolicy": {
    "shareDistribution": [
      {
        "shareIdentifier": "pink",
        “weightFactor”: 1.0
      },
      {
        "shareIdentifier": ”green”,
        “weightFactor”: 0.5
      },
    ]
  }
}
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fair share scheduling attributes

- `shareIdentifier`
  - A logical grouping for requests, like project, group, or department
  - `weightFactror` will adjust priority within a share identifier
  - `shareDecaySeconds` will adjust weights based on historical data
- `schedulingPriority`
  - Job definition level re-ordering across share identifiers



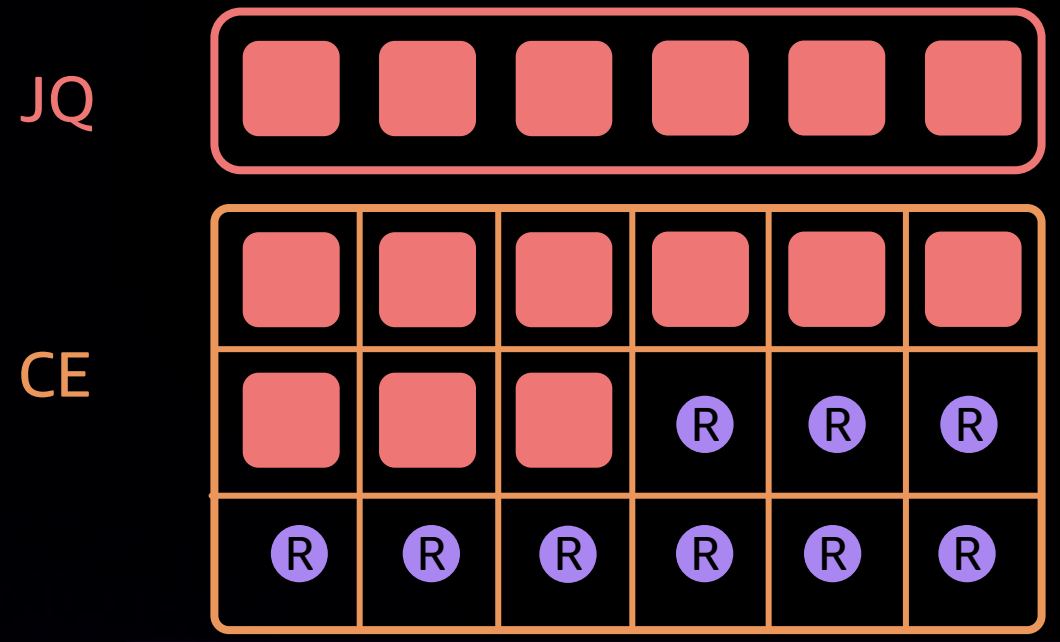

```
# register job definition
{
  "jobDefinitionName": ”myJobDef",
  "schedulingPriority": 1,
  ...
}
# submit job
{
  ...
  "schedulingPriorityOverride”: 10
  ...
}
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fair share scheduling attributes

- `shareIdentifier`
  - A logical grouping for requests, like project, group, or department
  - `weightFactror` will adjust priority within a share identifier
  - `shareDecaySeconds` will adjust weights based on historical data
- `schedulingPriority`
  - Job definition level re-ordering across share identifiers
- `computeReservation`
  - Always save some capacity for other share identifiers not yet submitted

$$\left(\frac{computeReservations}{100}\right)^{activeShareIds}$$



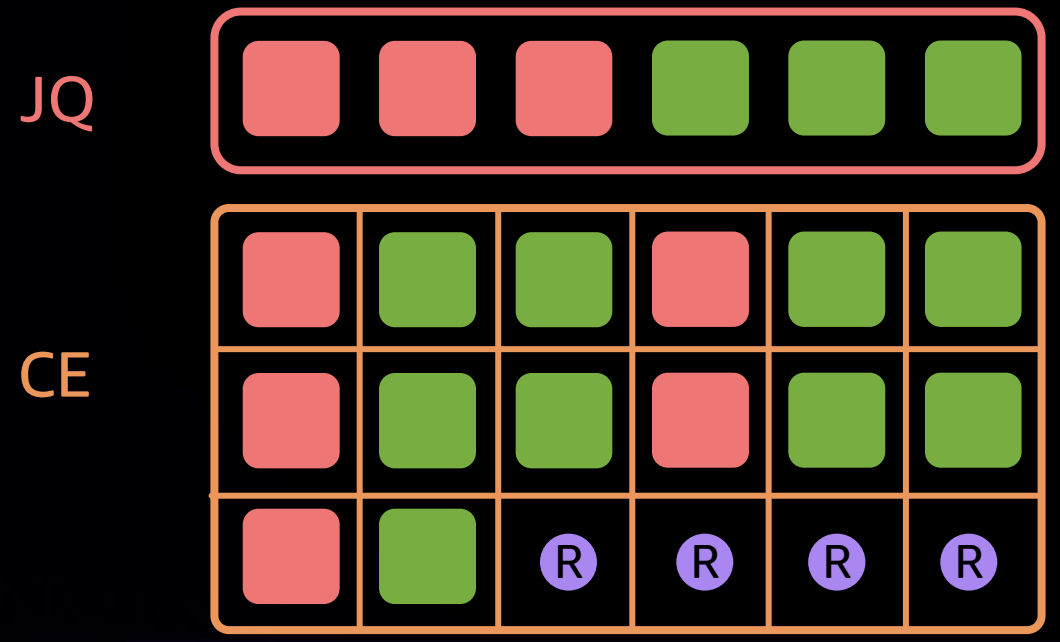

```
{
  "fairsharePolicy": {
    ”computeReservation: 50,
    "shareDistribution": [
      {
        "shareIdentifier": "pink”,
        “weightFactor”: 1.0
      },
      {
        "shareIdentifier": ”green”,
        “weightFactor”: 0.5
      }
]}}
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fair share scheduling attributes

- `shareIdentifier`
  - A logical grouping for requests, like project, group, or department
  - `weightFactror` will adjust priority within a share identifier
  - `shareDecaySeconds` will adjust weights based on historical data
- `schedulingPriority`
  - Job definition level re-ordering across share identifiers
- `computeReservation`
  - Always save some capacity for other share identifiers not yet submitted

$$\left(\frac{computeReservations}{100}\right)^{activeShareIds}$$



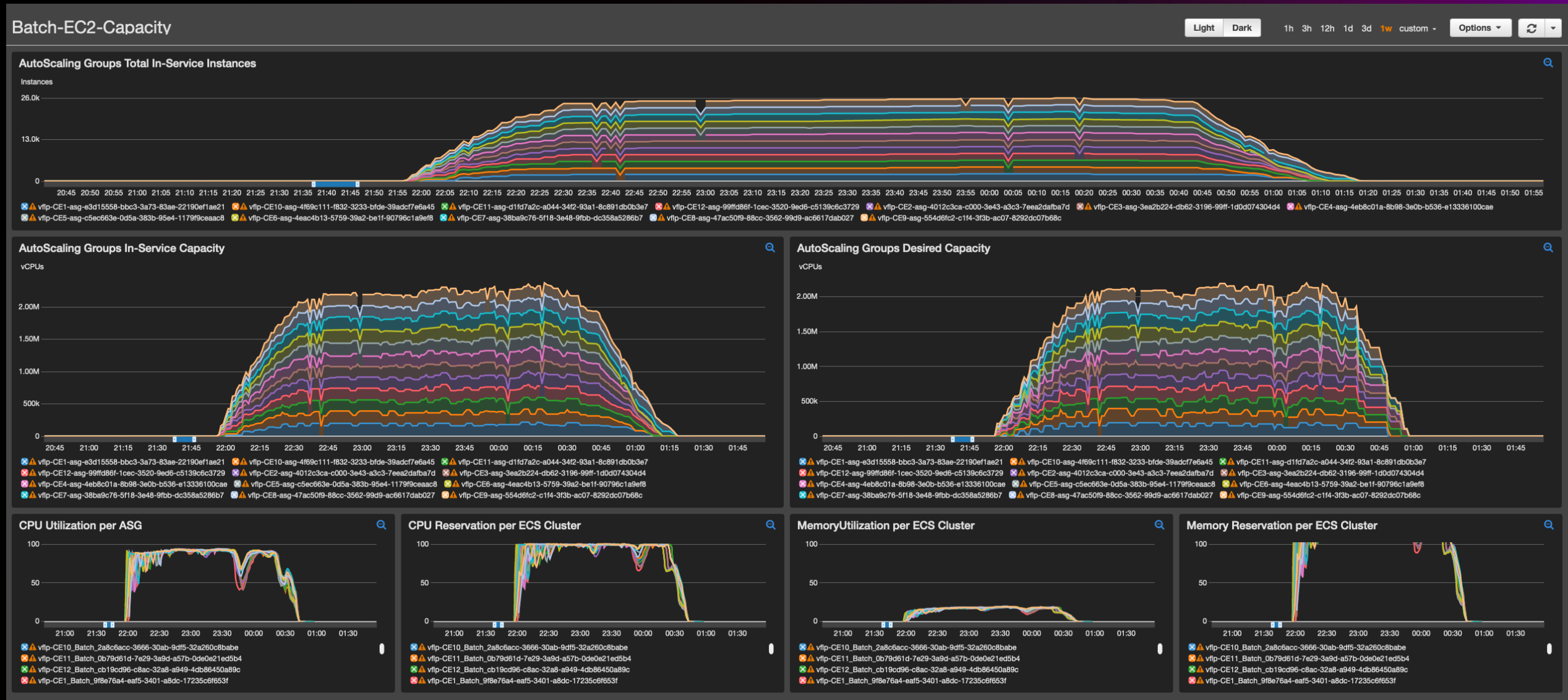

```
{
  "fairsharePolicy": {
    ”computeReservation: 50,
    "shareDistribution": [
      {
        "shareIdentifier": "pink”,
        “weightFactor”: 1.0
      },
      {
        "shareIdentifier": ”green”,
        “weightFactor”: 0.5
      }
]}}
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Batch best practices

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

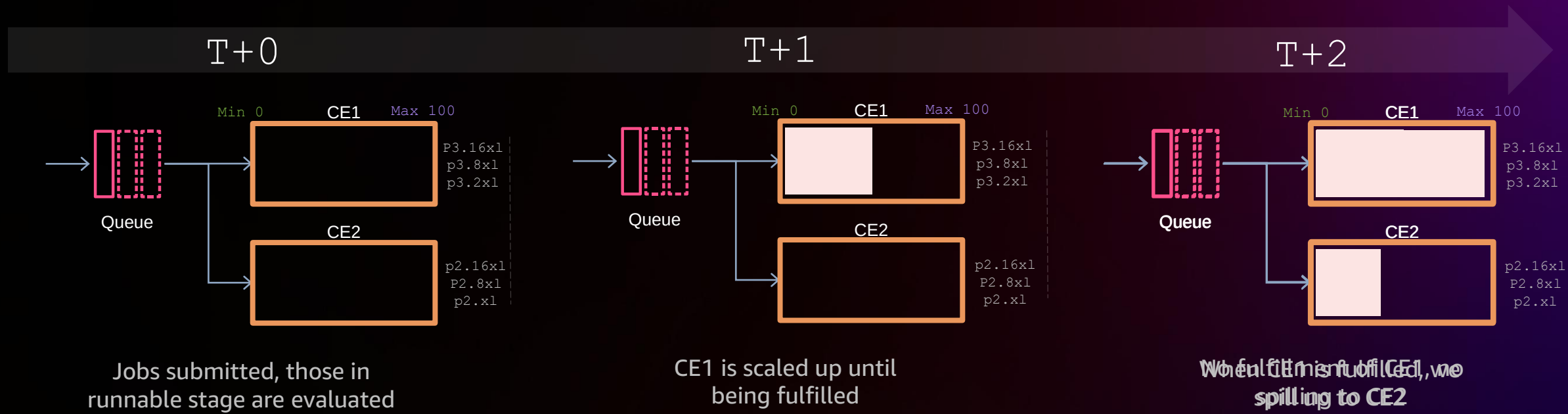
High scalability strategies



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CEs in-order scaling example

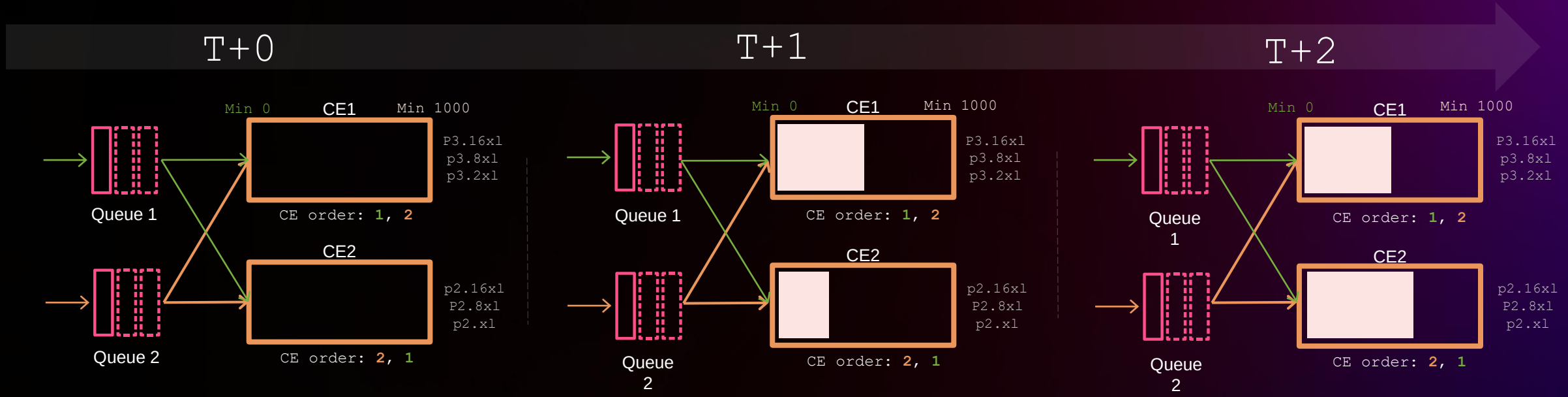
- CEs are scaled in order by the JQ
- Switching to the next CE occurs when *maxvCPU* is met in the current CE
- CEs can be scaled concurrently to meet capacity requirements



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CE/JQ interleaving technique

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



Jobs are distributed to both queues at the same time

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

Jobs can be scheduled on both CEs to be executed

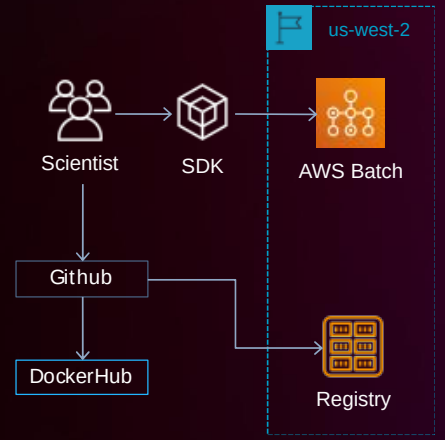
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Container 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                dri    _ve…   152MB
<missing>           12 days ago         |3 archive_dir=D                dri    _ve…   205MB
<missing>           12 days ago         |3 archive_dir=D                dri    _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

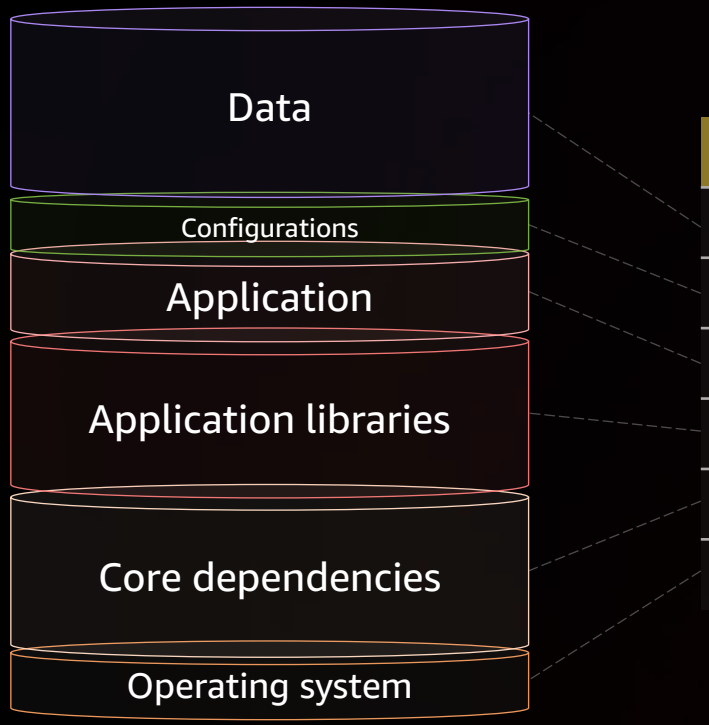
- Prefer Amazon ECR or Amazon S3 for Batch
  - DockerHub throttles under load, private registries can suffer



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

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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 and 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

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Ala Abunijem
aabunij@amazon.com

Angel Pizarro
pizarroa@amazon.com



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.