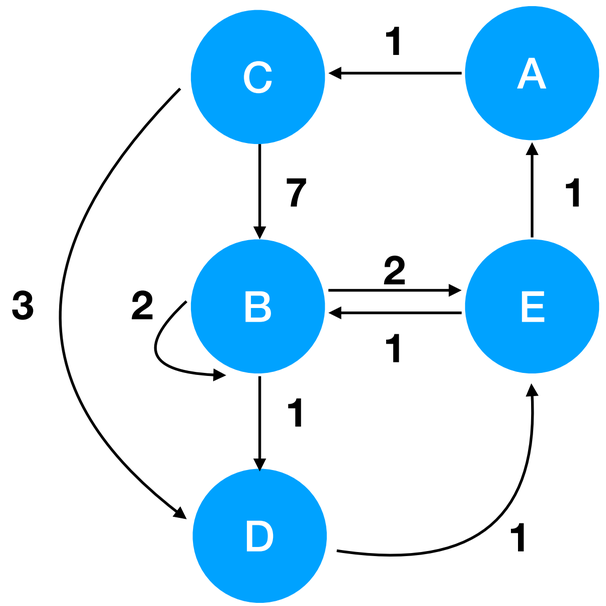
Running containerized C++ processes in AWS Batch

David Garcia
@dassun
Barcelona C++ Group

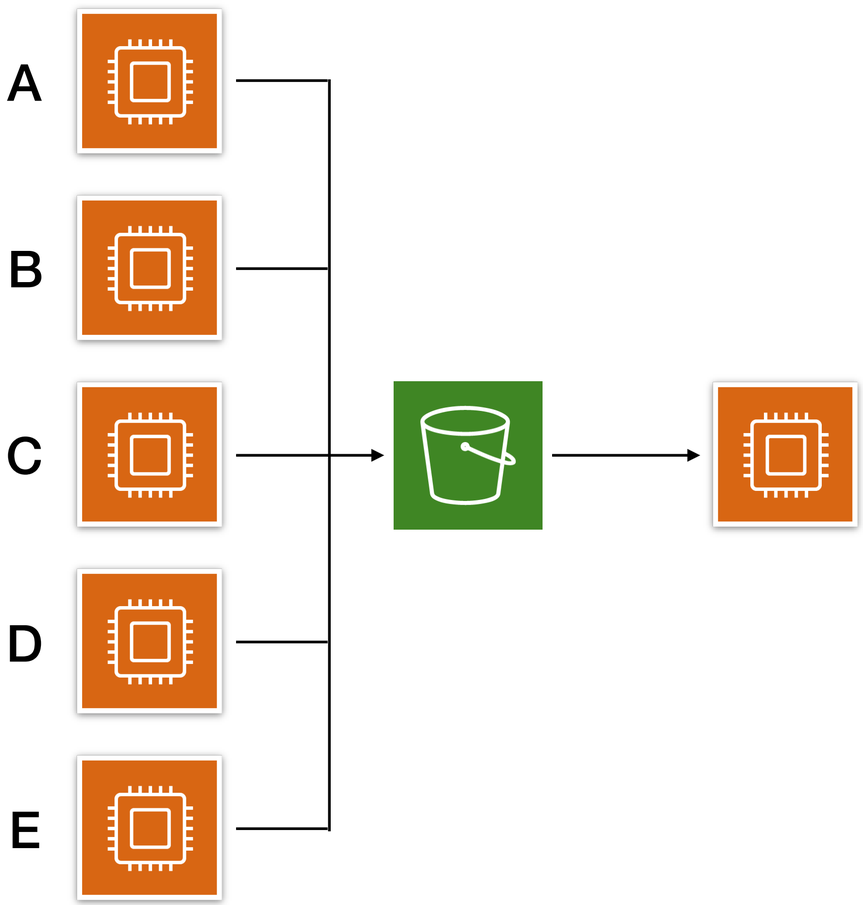
The problem

- Run a distributed C++ app in the cloud
- Calculate paths from all the nodes to all the nodes of a network
- Using Boost Dijkstra implementation
- But distributed



Distributed

- Each instances compute all the paths from an origin and write the solution to disk
- A new instance reads the results and aggregates them



Why…?

- The cloud: flexibility, millions of nodes :)
- AWS: popular option (39% market share)

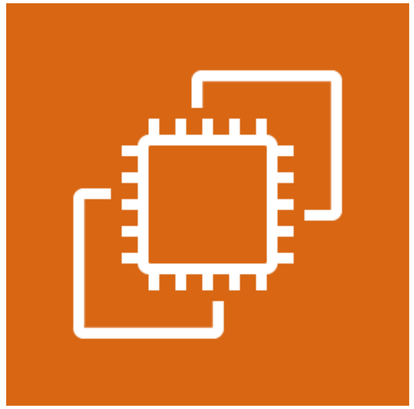
aws

Microsoft Azure

Google Cloud

How: AWS EC2

- Spin up **machines** in the cloud
- Install software on them
- Execute the software
- Consolidate results
- https://aws.amazon.com/ec2/



How: AWS HPC

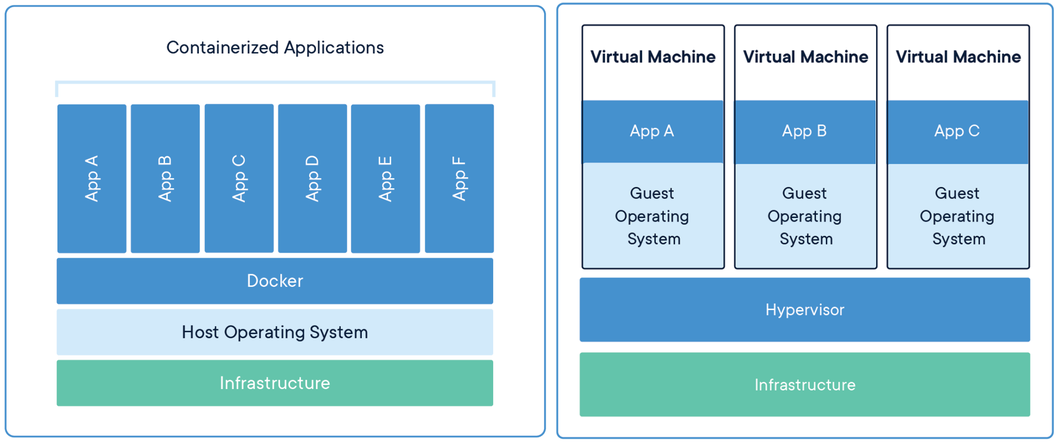
- Create a **cluster**
- Install software in master
- Execute a distributed app (using MPI probably)
- Consolidate results
- https://aws.amazon.com/hpc/

How: AWS Batch

- Create a container to run the task
- Create a parallel batch job using the container
- Consolidate results
- No infrastructure to manage
- https://aws.amazon.com/batch/

Containers

- A standardised unit of software
- Runs a single process
- Lightweight: they share a base OS (vs VM)
- But still offers isolation
- Text and images from https://www.docker.com/resources/what-container



Docker

- A container type and platform (others: rkt from CoreOS , LXC…)
- Docker because AWS compatibility

Docker commands

- build: build an image from a Dockerfile
- run: run a command in a new container
- image ls: list images
- image rm: remove one or more images
- container ls: list containers
- container prune: removes stopped containers

Create a Container

- The app will run in Linux (Amazon Linux, Ubuntu, CentOS…)
- Do you have Linux?
  - Yes, native user, HDMI output fails
  - No, I will use a VM (oh! Multipass)
  - No, I will use a container for compilation (and install Docker Desktop)

A container to compile

- Create a container to do the compilation

```
docker build -t batch-talk/cpp-build-base:0.1.0 . -f docker/base-builder/Dockerfile
```

- Simple: https://medium.com/@mfcollins3/shipping-c-programs-in-docker-1d79568f6f52
- Deluxe: https://devblogs.microsoft.com/cppblog/build-c-applications-in-a-linux-docker-container-with-visual-studio/

```
# Dockerfile
FROM ubuntu:latest

RUN apt-get update && apt-get install -y build-essential git cmake autoconf libtool pkg-config libboost-all-dev
```

Main container

- Create our container

```
docker build -t batch-talk/shortest-path-s3:1.0.0 . -f docker/app-builder-s3/Dockerfile
```

- Multi Stage builds: https://docs.docker.com/develop/develop-images/multistage-build/

```
# Dockerfile
FROM batch-talk/cpp-build-base:0.1.0 AS build
WORKDIR /src
COPY shortest-path shortest-path
RUN cd shortest-path && cmake . && make
FROM ubuntu:latest
RUN apt-get update && \
    apt-get install -y awscli
WORKDIR /opt/shortest-path
COPY --from=build /src/shortest-path/bin/shortest-path ./
COPY --from=build /src/shortest-path/s3/shortest-path-s3.sh ./
CMD ["./shortest-path-s3.sh"]
```

Our container

- Runs the shortest path app and log the results to a file
- If a S3 bucket has been configured, copy the result there, if not, show the log file

```
#!/bin/bash
./shortest-path "$@" >
"$AWS_BATCH_JOB_ID"_"$AWS_BATCH_JOB_ARRAY_INDE
X"_shortest-path.log
if [ -z ${BUCKET_NAME+x} ]; then
    echo "Missing bucket name. Log file says:"
    more
"$AWS_BATCH_JOB_ID"_"$AWS_BATCH_JOB_ARRAY_INDE
X"_shortest-path.log;
else
    aws s3 cp
"$AWS_BATCH_JOB_ID"_"$AWS_BATCH_JOB_ARRAY_INDE
X"_shortest-path.log s3://"$BUCKET_NAME"/;
fi
```

Testing it

- Run

```
docker run --env
AWS_BATCH_JOB_ARRAY_INDEX=1 --env
AWS_BATCH_JOB_ID=10 batch-talk/
shortest-path-s3:1.0.0
```

```
::::::::::::::
10_1_shortest-path.log
::::::::::::::
Path from B to A. Cost: 3
E
A
Path from B to B. Cost: 0
B
Path from B to C. Cost: 4
E
A
C
Path from B to D. Cost: 1
D
Path from B to E. Cost: 2
E
```

Results container

- Create the container

```
docker build -t batch-talk/
shortest-path-s3:1.0.0 . -f
docker/app-builder-s3/Dockerfile
```

```
# Dockerfile
FROM ubuntu:latest
RUN apt-get update && \
    apt-get install -y awscli
WORKDIR /src
COPY shortest-path shortest-path
CMD ["shortest-path/s3/shortest-
path-post-s3.sh"]
```

Results container

- List the result files in the bucket but the real implementation will aggregate them

```
#!/bin/sh
if [ -z ${BUCKET_NAME+x} ]; then
    echo "Missing bucket name."
else
    aws s3 ls s3://"$BUCKET_NAME"/
fi
```

AWS Batch

- Submit a job, using a job definition, into a queue. Jobs are run in a compute environment
  - Job: A unit of work
  - Job Definition: how jobs are to be run
  - Job Queues: queue with jobs waiting for execution
  - Compute Environment: set of managed or unmanaged compute resources that are used to run jobs

AWS Resources

- We will need an AWS Account
- AWS CLI installed

- AWS Resources used
  - One S3 Bucket
  - One Policy
  - Two Roles
  - Two Amazon ECR repositories
  - One AWS Batch Compute environment
  - One AWS Batch Job queue
  - Two Job definitions and two jobs
  - Logs in AWS CloudWatch

S3 Bucket

- S3 is an object storage service
- A Bucket is a container for objects stored in Amazon S3
- We will write our results in a bucket

```
aws s3api create-bucket \
--acl private \
--bucket ${AWSBUCKETNAME} \
--region ${AWSREGION} \
--create-bucket-configuration
LocationConstraint=${AWSREGION}
```

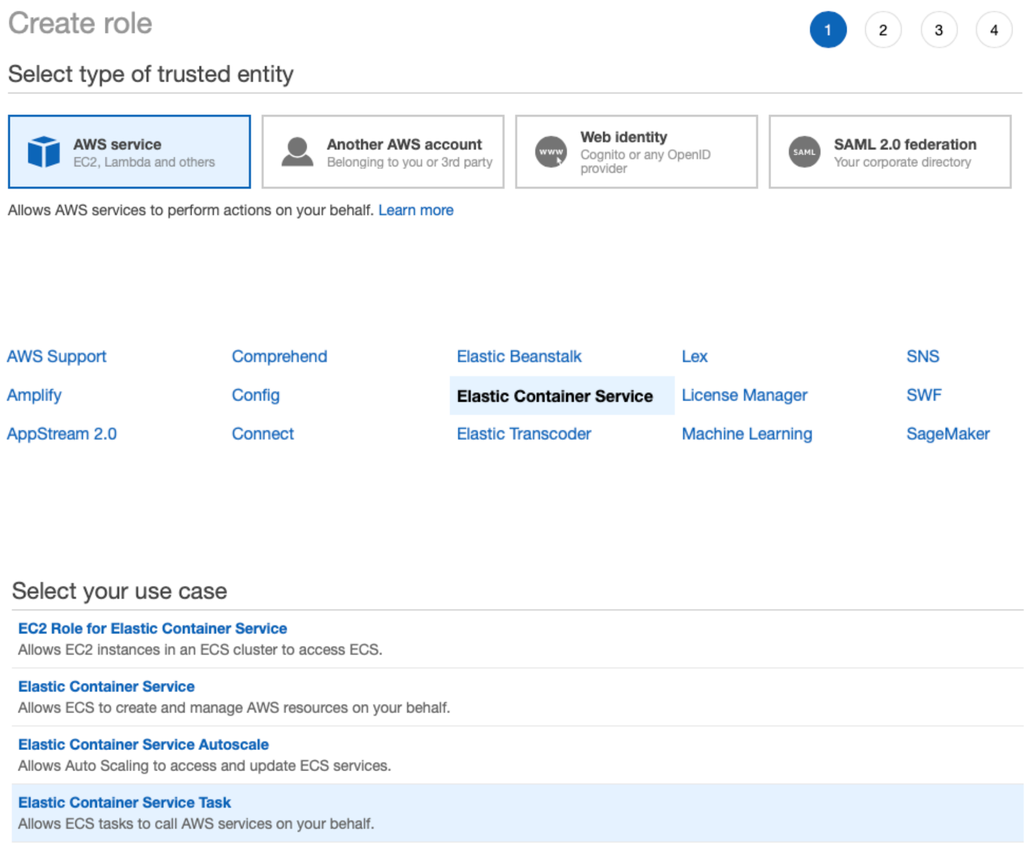
Policies

- A Policy defines permissions
- Principle of least privilege
- Rules:
  - If any policy denies -> AccessDenied
  - If some policy allows -> Allow
  - Otherwise AccessDenied

```
Policy: batch-talk-bucket-s3-policy
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::${AWSBUCKETNAME}/*"
      ]
    }
  ]
}
```

Roles

- A Role is an identity with a set of permissions
- Same trusted entity -> AWS service: ecs-tasks.amazonaws.com
- Assign policies to roles:
  - batch-talk-bucket-s3-role: batch-talk-bucket-s3-policy
  - batch-talk-post-bucket-s3-role: AmazonS3ReadOnlyAccess



Amazon ECR repositories

- Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images

```
aws ecr create-repository \
--repository-name \
batch-talk/shortest-path-s3
```

Compute Environment

- Name it: batch-talk-ce
- Requires a VPC and subnets
- Service role: AWSBatchServiceRole
- Instance role: ecsInstanceRole
- …
- Add a Name tag with the value batch-talk

- VPC
  - Elastic IP
  - VPC Wizard: VPC with Public and Private Subnets
- https://docs.aws.amazon.com/batch/latest/userguide/create-public-private-vpc.html
- https://aws.amazon.com/premiumsupport/knowledge-center/batch-job-stuck-runnable-status/

Job Queue

- Name
- Priority (same Compute Environment)
- Compute Environment

```
aws batch create-job-queue \
--job-queue-name batch-talk-queue \
--state ENABLED \
--priority 1 \
--compute-environment-order \
order=1,computeEnvironment=batch-
talk-ce
```

Job Definition

- Specify how jobs are to be run:
  - Container image
  - How many vCPUs and how much memory to use with the container
  - What IAM role your job should use for AWS permissions
  - …

aws batch register-job-definition --cli-input-json file://aws-batch/s3/shortest-path-s3-def.json

```
{
    "jobDefinitionName": "shortest-path-s3",
    "type": "container",
    "containerProperties": {
        "image": "${AWSID}.dkr.ecr.$
{AWSREGION}.amazonaws.com/batch-talk/shortest-path-
s3:latest",
        "vcpus": 1,
        "memory": 250,
        "jobRoleArn" : "arn:aws:iam::${AWSID}:role/batch-
talk-bucket-s3-role",
        "environment": [
            {
                "name": "BUCKET_NAME",
                "value": "${AWSBUCKETNAME}"
            }
        ]
    }
}
```

Job Definition (Post)

- Specify how jobs are to be run:
  - Container image
  - How many vCPUs and how much memory to use with the container
  - What IAM role your job should use for AWS permissions
  - …

aws batch register-job-definition --cli-input-json file://aws-batch/s3-post/shortest-path-post-s3-def.json

```
{
    "jobDefinitionName": "shortest-path-post-s3",
    "type": "container",
    "containerProperties": {
        "image": "${AWSID}.dkr.ecr.$
{AWSREGION}.amazonaws.com/batch-talk/shortest-path-post-
s3:latest",
        "vcpus": 1,
        "memory": 250,
        "jobRoleArn" : "arn:aws:iam::${AWSID}:role/batch-
talk-post-bucket-s3-role",
        "environment": [
            {
                "name": "BUCKET_NAME",
                "value": "${AWSBUCKETNAME}"
            }
        ]
    }
}
```

Job

- Jobs are the unit of work executed by AWS Batch
  - Array size (if required)
  - Dependencies (if any)
  - …

aws batch submit-job --cli-input-json file://aws-batch/s3/shortest-path-s3-job.json

```
{
    "jobName": "shortest-path-s3",
    "jobQueue": "batch-talk-queue",
    "arrayProperties": {
        "size": 5
    },
    "jobDefinition": "shortest-path-s3"
}
```

Job with Dependencies

- Take the jobId from previous command
- Modify the job according and run it

aws batch submit-job --cli-input-json file://aws-batch/s3-post/shortest-path-post-s3-job.json

```
{
    "jobName": "shortest-path-post-s3",
    "jobQueue": "batch-talk-queue",
    "jobDefinition": "shortest-path-post-s3",
    "dependsOn": [
        {
            "jobId": “4a151cd4-…”
        }
    ]
}
```

CloudWatch

- Look for results in Log groups > /aws/batch/job

Demo

Now, what?

- Use CDK (Cloud Development Kit) to recreate all the resources at once:
  - Variables: AWSID, AWSREGION, AWSBUCKETNAME
  - Dependencies (Queue -> CE -> VPC -> Subnets, Security groups)
- AWS Lambda (https://engineering.door2door.io/orchestrating-scheduled-data-batch-jobs-on-aws-ee45f940696f) and SQS (Simple Queue Service)
- Run the presentation exercise at home :)

Questions?