Event-Driven Batch Processing on AWS

Guido Schmutz – 1.3.2018

@gschmutz

guidoschmutz.wordpress.com

AWS Compute - Overview

trivadis
makes IT easier.

AWS Compute – What is available

Elastic Compute (EC2) => Virtual servers in the cloud

Elastic Container Service (ECS) => Run and manage docker containers

AWS Fargate => run containers without managing servers or clusters, based on ECS, EKS in the future

Elastic Container Service for Kubernetes (EKS) => run managed and scalable Kubernetes (in preview)

AWS Lambda => Run code in response to events

AWS Batch => Run batch jobs at any scale

AWS Elastic Beanstalk => run and manage web application

trivadis
makes IT easier.

AWS Batch

trivadis
makes IT easier.

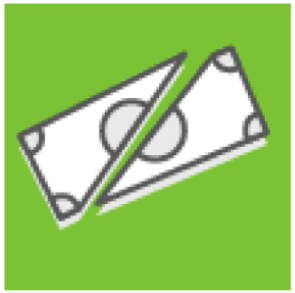
AWS Batch Introduction

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

trivadis
makes IT easier.

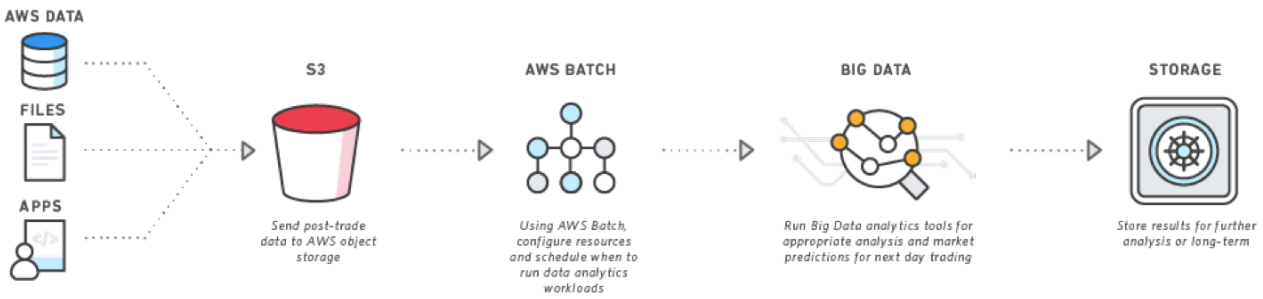
AWS Batch Introduction

Fully managed batch primitives

Focus on your applications (shell scripts, Linux executables, Docker images) and their resource requirements

AWS Batch takes care of the rest!

Cost-optimized resource provisioning



trivadis
makes IT easier.

AWS Batch Concepts

- Jobs
- Job Definitions
- Job Queues
- Compute Environments
- Scheduler

trivadis
makes IT easier.

Jobs

Jobs are the unit of work executed by AWS Batch as containerized applications running on Amazon EC2 container instances in an ECS cluster

Containerized jobs can reference a container image, command, and parameters.

A large number of independent, simple jobs can be submitted.

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

Job Definitions

Similar to Elastic Container Services (ECS) task definitions

AWS Job Definitions specify how jobs are to be run (think of it as a blueprint for the resources of a job

While each job must reference a job definition, many parameters can be overridden upon submitting the job.

Some of the attributes specified in a job definition:

- Which Docker image to use with the container in the job
- IAM role associated with the job
- vCPU and memory requirements
- Mount points
- Container properties
- Environment variables
- Retry strategy

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

trivadis
makes IT easier.

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

Job queues support priorities and multiple queues can schedule work to the same compute environment

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

trivadis
makes IT easier.

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 Spot Instance bid as a % of the On-Demand price). AWS Batch launches and scales resources on your behalf.

You can choose specific instance types (e.g. c3.2xlarge, c3.4xlarge) / instance families (e.g. C4, M4, R4) or chose "optimal" to allow AWS Batch to launch appropriately sized instances from their more modern instance families.

```
aws batch create-compute-environment --compute-environment-name
managedce --type MANAGED ...
```

trivadis
makes IT easier.

Scheduler

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

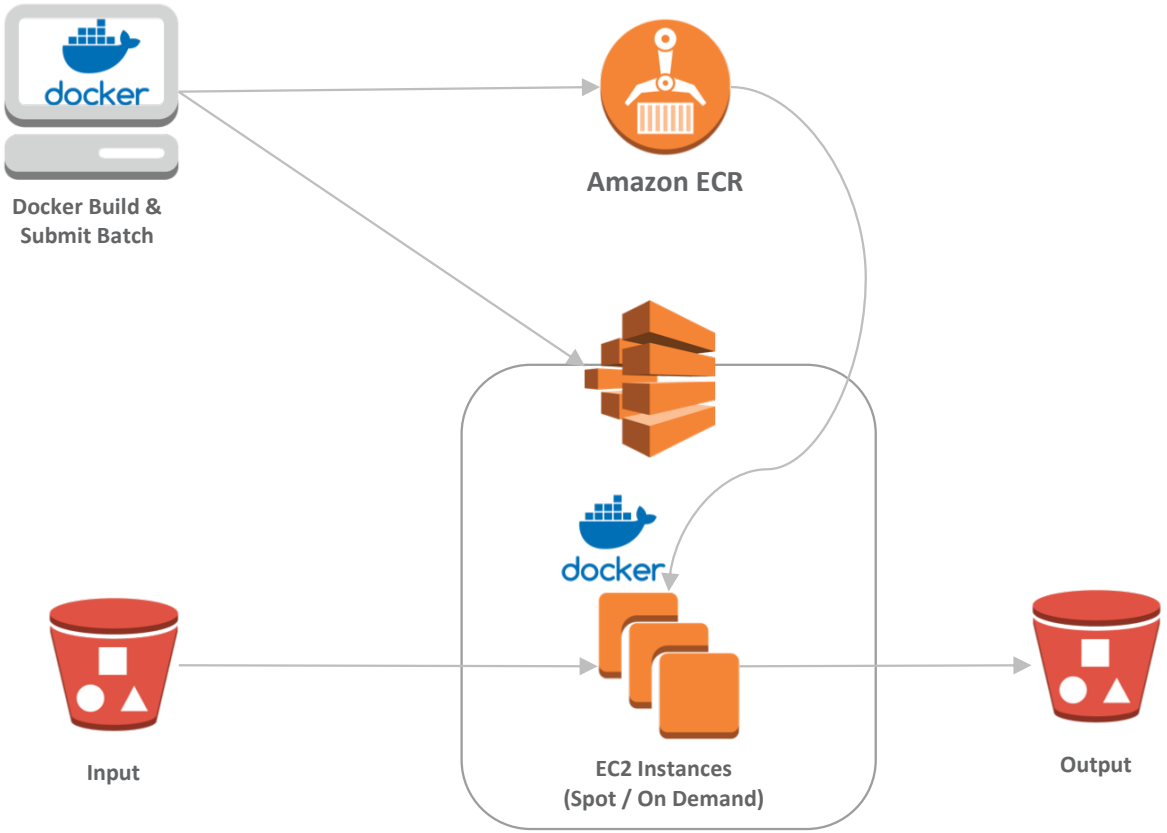
Scheduler

trivadis
makes IT easier.

AWS Batch - Demo

trivadis
makes IT easier.

Demo – 1st step



trivadis
makes IT easier.

Dockerfile implementing batch job

```
FROM jrottenberg/ffmpeg:latest

RUN apt-get update -q
RUN DEBIAN_FRONTEND=noninteractive apt-get install -qy python-pip groff-base
RUN pip install awscli

ADD run.sh /usr/local/bin/run.sh

WORKDIR /tmp

CMD         ["--help"]
ENTRYPOINT ["/usr/local/bin/run.sh"]
```

```
TMP_BASEDIR=/tmp/ffmpeg
TMP_OUTPUT_DIR=${TMP_BASEDIR}/output
TMP_INPUT_DIR=${TMP_BASEDIR}/input

S3_INPUT_FILE="${1}"
S3_OUTPUT_PATH="s3://gus-audi-test/videos/output"

IFS=',' read -r -a S3_ARRAY_INPUT_FILE <<< "${2}"

#if [ -z "${INPUT_FILE_S3_URL}" ]; then
#  usage "INPUT_FILE_S3_URL not set. No object to download."
#fi

# Standard function to print an error and exit with a failing return code
error_exit () {
  echo "${BASENAME} - ${1}" >&2
  exit 1
}

# Run the ffmpeg
run_ffmpgeg () {
  # Create a temporary file and download the script
  # aws s3 cp s3://gus-audi-test/videos/input/"${INPUT_FILE_S3_URL}" - > /tmp/etl.mp4 || error_exit "Failed to download S3 script."

  echo "Starting job .... Job-ID: ${AWS_BATCH_JOB_ID} / Attempt=${AWS_BATCH_JOB_ATTEMPT} / ArrayIndex=${AWS_BATCH_JOB_ARRAY_INDEX}"
  mkdir -p ${TMP_INPUT_DIR}
  mkdir -p ${TMP_OUTPUT_DIR}

  echo "File 1 ${S3_ARRAY_INPUT_FILE}[0]"


  echo "copying ${S3_INPUT_FILE} to local folder ...."
  aws s3 cp "${S3_INPUT_FILE}" - > ${TMP_INPUT_DIR}/inputfile.mp4 || error_exit "Failed to download S3 script."
  echo "... copied successfully !"

  echo "running ffmpeg ...."
  ffmpeg -i ${TMP_INPUT_DIR}/inputfile.mp4 -vf "select=gt(scene\,0.5)" -frames:v 100 -vsync vfr ${TMP_OUTPUT_DIR}/image%02d.jpg
  echo "running ffmpeg finished successfully!"

  echo "list of local result files after run of FFMPEG"
  ls -lsa ${TMP_OUTPUT_DIR}

  echo "copying result files to S3 ...."
  touch xxxx
  #aws s3 cp AWS_BATCH_JOB_ID s3://gus-audi-test/videos/output/
  aws s3 cp ${TMP_OUTPUT_DIR} ${S3_OUTPUT_PATH}/${AWS_BATCH_JOB_ID}/ --recursive

  echo "finsihing ...."
  touch ${TMP_OUTPUT_DIR}/SUCCESS
  aws s3 cp ${TMP_OUTPUT_DIR}/SUCCESS ${S3_OUTPUT_PATH}/${AWS_BATCH_JOB_ID}/SUCCESS
}

run_ffmpgeg "${@}"
```

trivadis
makes IT easier.

Build and Deploy the image

```
aws ecr get-login --no-include-email --region eu-central-1
```

```
docker login -u AWS -p
eyJwYXlsb2FkIjoiYkVSNnVDRktaQ3VsUytCNnZHSXh5Nm5ZNXBoamNmRWZiTndBUlQwY09hWG0rYnFsa2hzQlpDRVRZUVMvRmxOaW4
0QW4zYmtEZTNveWR0RXFRdXZtUEJHZjd4QitvcURFanRtM28rTWg3OWtLeTBSZkE5NEJnWmN4M2RMdko0ZG56d3ovbHNpTmRwUVErL2
xGRE5jR25yWjdSY3B2bEZ3dE12dFBNM3JhbmdZZHYyTy9ybnVvcUw2RU5XUTY2eDV3aWl0eWtzNTZOYXRLbXpQajZta3lpMmdVWGNxM
zZRQjBiT2JkN3dkRWtDKy8rUGRZVE5ibVJjTWhOdlNwbGpPcUcxbmRlMDZ4VlloNTNiMU5UQm83ZWJoSGZQRnV6Q1o4T0phOUJPV0dw
L3dKOEZuOHFUZ2cwaDFiVlkvMjZzU3lzZ3VraGhaZ0lYcjArck9CUGJydUd3cEJST3ZrOFVaU2tOdG00bytSNGxEcXprT250aGxpSVh
aOFg2NVdJK25GWnVZSitEWHJuTTkyL3hjdXFoL0xYMnJLRU40VUp2b0QvYjVFTEtUeUlZR3J6QjB6aWthWDkyR1d4ejN6V1FrNU1EUF
V6UU1uRGxzZTBGaHRZY21sdlQvcnVDRjQxNGhtVjVDTVpwRWIxYUZ2YVdnQmNyWS9DN0lGbHlINzdyWEZkTC9XeEd4THUzaFJndnFjb
kttN0xGd2Y0alFsS3IyeVk1VlpnWDc4aG1HUVRlcXdYQklNQXczdCtHSVF0NDVzdUxhVTFNZGRHSk8xbVQvTUJoajk2OHdycnZNSStt
WFc2QnhLQksyTUdmZm9pWHliRXc5emlsOXU5dklTeU54UW5RMjVzckFKRTEyMDhYSHJCaFhNd0EveW5kdWxuSTVDZmtWZVZjRUU2L0V
4OFY5cUkwUG43ZnRNWkZtbVhROFRmeDlsVTM1TWlpWVpQelhxVHl1TDYxaGdEcDdBRmpFMkpXT2MwUytMbmZQZTk4MzFWbjdVZnVFOU
ZqdEZ5bExIS2I2K29yWmV5MlJ5a0VmMHJHa1A3Qi91UHFWVmgyWUhWWVhmTTlYZDM1alE2TzVlckQzcGNMaGk5MkgwRnUrQWdJWHQ3e
FNFSU1NdUt6NmVqUWkzNjhZRytod0Z1dDFqNWN6RHF3SWMyTXV3aWV4WTZtTFdxLzl6SEhVN1pNS0ZpR3BiY1B5azZyRkJvYmxsVEpN
dDVkUlJGUEg0TUlCUDJYZGFReGVDb2V4a2RHNktEYlgyMktRSjhtMzRsWlEwNjU5V3NGMlV4eXV2d1h0bWRJZ1k2RzVTQjNqYkdWK3B
adTllWXdyRHR2UDFuTm5VS29sR01BeWlWQU1idW45MUJGdUt0MnJvekh5VGJYd0hRMW5hTWJESUxNeFJCbzRGRlg1alpBanFVMk9mZG
9qY0xRQWgwbzMzZ1NFYVFDd0tQTzlIS0M1ajA1UC9waEpGRnQ4TXpZVno5Ym1sLzlBTE92cEJqc3FUa0d1dCIsImRhdGFrZXkiOiJBU
UVCQUhoM3pZT2R0cEJCU1Z3L1kyYW44QlhJRDRrd0xRZm5vVGo1endadEdKWm9aUUFBQUg0d2ZBWUpLb1pJaHZjTkFRY0dvRzh3YlFJ
QkFEQm9CZ2txaGtpRzl3MEJCd0V3SGdZSllJWklBV1VEQkFFdU1CRUVESnh3MnBpR0dHWGFKYVJpUHdJQkVJQTdiSWFwWGlFVCsvVFN
3S0M0QWdwakhUZTNHdEZCMXhQZDVHU2p1dExZQlN0Ym1kL0R0aG5VUmJja1FrVTdlVFhZc3dueGkxT0RmdzV6WWlVPSIsInZlcnNpb2
4iOiIyIiwidHlwZSI6IkRBVEFfS0VZIiwiZXhwaXJhdGlvbiI6MTUxOTc4MzU0NH0= https://663559919114.dkr.ecr.eu-
central-1.amazonaws.com
```

trivadis
makes IT easier.

Build and Deploy the image

```
docker build -t trivadis/ffmpeg .
```

```
docker tag trivadis/ffmpeg:latest 663559919114.dkr.ecr.eu-central-1.amazonaws.com/trivadis/ffmpeg:latest
```

```
docker push 663559919114.dkr.ecr.eu-central-1.amazonaws.com/trivadis/ffmpeg:latest
```

https://eu-central-1.console.aws.amazon.com/ecs/home?region=eu-central-1#/repositories

trivadis
makes IT easier.

Create a AWS Batch Environment – Job Definition

Job definitions > Create job definition

Create a job definition

Job definitions allow you to save the values for a job so it can be used as a template later.

Job definition name ffmpeg_job

Job role audi-ecs-role

You can optionally specify an IAM role that provides the container in your job with permissions to use the AWS APIs. This feature uses Amazon ECS IAM roles for tasks functionality. Learn more.

Container image 59919114.dkr.ecr.eu-central-1.amazonaws.com/trivadis/ffmpeg

Environment

Command Space delimited | JSON

space delimited: echo hello world

Text is delimited by spaces and converted to a JSON array to be submitted with the job.

JSON result:

```
[ ]
```

For more details on how the command field is used in your job definition, view our documentation.

vCPUs 1

Memory (MiB) 1000

Job attempts 1

trivadis makes IT easier.

Create a AWS Batch Environment – Job Definition

```
aws batch register-job-definition
    --cli-input-json file://<path>/job_definition.json
```

```
{ "jobDefinitionName": "ffmpeg_job",
  "type": "container",
  "parameters": {"inputFileName": "s3://inputFileName.mp4"},
  "containerProperties": {
     "image": "663559919114.dkr.ecr.eu-central-1.amazonaws.com/trivadis/ffmpeg",
     "vcpus": 1,
     "memory": 1000,
     "command": [],
     "jobRoleArn": "arn:aws:iam::<account-id>:role/openfoam-docker-job-role",
     "readonlyRootFilesystem": false,
     "privileged": true,
     "user": "nobody"
  },
  "retryStrategy": { "attempts": 3 }
}
```

trivadis
makes IT easier.

Create a AWS Batch Environment – Computing Environment

aws Services Resource Groups Guido Schmutz Frankfurt Support

AWS Batch
- Dashboard
- Jobs
- Job definitions
- Job queues
- **Compute environments**

Compute environments > Create a compute environment

Create a compute environment

A compute environment is a collection of AWS Batch compute resources that can be associated with a job queue. In a managed compute environment, AWS Batch handles the scaling and provisioning of your instances for you; in an unmanaged compute environment, you control the instances.

Compute environment type
- (●) Managed — AWS scales and configures your instances for you.
- ( ) Unmanaged — You control and manage the instance configuration, provisioning, and scaling.

Compute environment name: on-demand

Service role: AWSBatchServiceRole
You can optionally specify an IAM role that provides the container in your job with permissions to use the AWS APIs. This feature uses Amazon ECS IAM roles for tasks functionality.

Instance role: ecsInstanceRole
Your compute resources use the ecsInstanceRole IAM instance profile to make calls to the AWS APIs on your behalf. If you do not already have the ecsInstanceRole, we can create it for you.

EC2 key pair: SparkCluster

Enable compute environment [✓]

Configure your compute resources

Compute resources are the Amazon EC2 instances that jobs are placed on. There are multiple instance types that are available to use in AWS Batch as either ON-Demand or SpoT instances.

Provisioning model
- (●) On-demand — EC2 instances that you pay by the hour for.
- ( ) Spot

trivadis makes IT easier.

Create a AWS Batch Environment – Computing Environment

```
aws batch create-compute-environment
    --cli-input-json file://<path>/computing_env.json
```

```
{ "computeEnvironmentName": "audi-mdm-on-demand",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": { "type": "EC2",
    "minvCpus": 0,
    "maxvCpus": 10,
    "desiredvCpus": 1,
    "instanceTypes": [ "m3" ],
    "subnets": [ "subnet-31d3f758", "subnet-93b386d9", "subnet-ae473cd5"],
    "securityGroupIds": [ "sg-10809379" ],
    "ec2KeyPair": "SparkCluster",
    "instanceRole": "arn:aws:iam::663559919114:instance-profile/ecsInstanceRole"
  },
  "serviceRole": "arn:aws:iam::663559919114:role/service-role/AWSBatchServiceRole"
}
```

trivadis
makes IT easier.

Create a AWS Batch Environment – Job Queue

Job queues > Create job queue

Create a job queue

A job queue accepts job submissions and places them on an appropriate compute environment.

Queue name audi-mds-job-queue

Priority 1

Job queues with a higher integer value for priority are given preference for compute resources.

Enable Job queue ✔

When a queue is disabled it cannot accept new job submissions.

Connected compute environments for this queue

Jobs are submitted to the connected compute environments based on the order they are listed and the available capacity of those environments.

Order	Compute environment
1	Name: audi-mdm-on-demand Provisioning model: On-demand Instance types: c3 Min vCPUs: 0 Max vCPUs: 8

Select a compute environment *Select a compute environment...*

Cancel Create

trivadis makes IT easier.

Create a AWS Batch Environment – Job Queue

```
aws batch create-job-queue
    --cli-input-json file://<path>/job_queue.json
```

```
{
   "jobQueueName": "audi-mds-job-queue",
   "state": "ENABLED",
   "priority": 1,
   "computeEnvironmentOrder": [
      {
        "order": 1,
        "computeEnvironment": "audi-mdm-on-demand"
      }
   ]
}
```

trivadis
makes IT easier.

Run the job – AWS Console

Jobs > Submit job

Submit an AWS Batch Job

Select a job definition to run and apply any command, parameter, or environment variable overrides.

Job run-time

Job name ffmpeg_13223

Job definition ffmpeg_job:6

Job queue audi-mds-queue

Environment

Job Type Single

Array

Job depends on *optional: enter a jobId this will depend on*

Environment

Command **Space delimited** **JSON**

space delimited: echo hello world

Text is delimited by spaces and converted to a JSON array to be submitted with the job.

JSON result:

[ ]

For more details on how the command field is used in your job definition, view our documentation.

trivadis makes IT easier.

Run the job – from CLI

```
aws batch submit-job
    --cli-input-json file://<path>/job.json
```

```
{
   "containerOverrides": {
       "command": ["Ref::inputFileName"],
       "memory": 1000,
       "vcpus": 1
   },
   "jobDefinition": "ffmpeg_job:6",
   "jobName": "ffmpeg",
   "jobQueue": "audi-mds-queue",
   "parameters": "inputFileName": "s3://gus-audi-test/videos/input/etl.mp4"},
   "retryStrategy": {
       "attempts": 1
}}
```

trivadis
makes IT easier.

Run the job – Python API

```
analyzer_job = client.submit_job(
                  jobName='ffmpeg' + context.aws_request_id,
                  jobQueue='audi-mds-queue’,
                  jobDefinition='ffmpeg_job:6’,
                  parameters={
            'inputFileName': 's3://' + 'gus-audi-test’ + '/videos/input/etl.mp4’
                  },
      containerOverrides={
            'command': ['Ref::inputFileName’]
      })
```

trivadis
makes IT easier.

AWS Batch Dashboard

AWS Batch

- Dashboard
- Jobs
- Job definitions
- Job queues
- Compute environments

Create job

Job queues

Last loaded: 01:33:49 pm 03/01/18

Name	Priority	SUBMITTED	PENDING	RUNNABLE	STARTING	RUNNING	FAILED	SUCCEEDED
audi-mds-queue	1	0	0	1	0	0	0	20
SampleJobQueue-9562d9d9b5a8bb9	1	0	0	0	0	0	0	0

Compute environments

Last loaded: 01:33:39 pm 03/01/18

Name	Type	Minimum vCPUs	Desired vCPUs	Maximum vCPUs
audi-mdm-on-demand-II	MANAGED	0	0	10
SampleComputeEnvironmen-25ea9fb5081fc20	MANAGED	0	0	64
audi-mdm-on-demand	MANAGED	0	0	8
audi-mdm-spot	MANAGED	0	0	10
audi-mdm-spot-II	MANAGED	0	0	10

trivadis
makes IT easier.

Job States

SUBMITTED	Accepted into the queue, but not yet evaluated for execution
PENDING	Job has dependencies on other jobs which have not yet completed
RUNNABLE	Job has been evaluated by the scheduler and is ready to run. Jobs in this state are started as soon as there are sufficient resources available
STARTING	Job is in the process of being scheduled to a compute resource
RUNNING	Job is currently running
SUCCEEDED	Job has finished with exit code 0
FAILED	Job finished with a non-zero exit code or was cancelled or terminated

Job Retries

Job retries enable jobs to be retried due to failures. Failures can be caused by errors in the AWS Batch job itself or by termination of the Spot instance

Defined as the retry strategy as either part of the job definition or during `submitJob` API call.

Supports up to 10 attempts, with an attempt being a job moving into RUNNABLE state.

For each job attempt, the `AWS_BATCH_JOB_ATTEMPT` environment variable is set to the container's corresponding job attempt number

trivadis
makes IT easier.

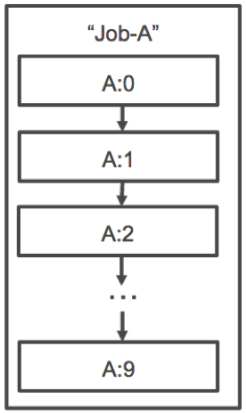
Array Jobs

An array job is a job that shares common parameters

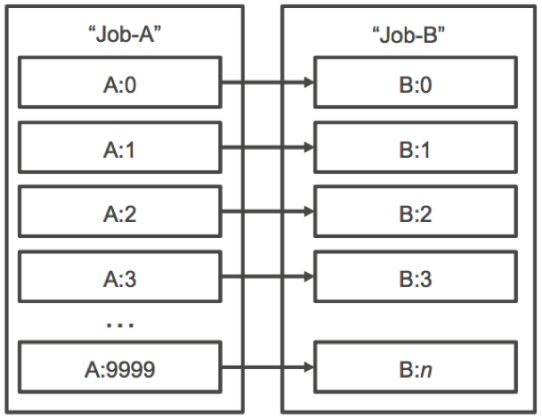
such as the job definition, vCPUs, and memory

runs as a collection of related, yet separate, basic jobs that may be distributed across multiple hosts and may run concurrently

SEQUENTAL



N_TO_N



trivadis
makes IT easier.

AWS Batch CLI

cancel-job
create-compute-environment
create-job-queue
delete-compute-environment
delete-job-queue
deregister-job-definition
describe-compute-environments
describe-job-definitions
describe-job-queues
describe-jobs
list-jobs
register-job-definition
submit-job
terminate-job
update-compute-environment
update-job-queue

```
aws batch list-jobs --job-queue audi-mds-queue --job-status RUNNABLE
```

trivadis
makes IT easier.

AWS Lambda

trivadis
makes IT easier.

AWS Lambda Introduction

Serverless compute platform for stateless code execution in response to events

No server to manage

Continuous scaling

No idle/cold servers

Pay per request

trivadis
makes IT easier.

Working with AWS Lambda

Bring your own code
- Node.js, Java, Python, C#, Go
- Bring your own libraries

Simple resource model
- Select power rating from 128MB to 1.5GB
- CPU and network allocated proportionately
- Metrics show usage

Flexible use
- Call or send events
- Integrated with other AWS services
- Build serverless ecosystems

Stateless
- Persist data using S3, RDS, DynamoDB, ...
- No affinity to infrastructure

Flexible authorization
- Securely grant access to resources
- Fine-grained control over what can call your functions

Programming model
- Build-in AWS SDK
- Use processes, threads normally
- Package code as .zip and upload to Lambda or S3
- Plugins for Eclipse and Visual Studio
- Command line tools

Monitoring and logging
- Built-in metrics for requests, latency, errors
- Built in logging with CloudWatch

trivadis
makes IT easier.

Common Use Cases

Data Triggers

- Trigger functions on data updates in S3, SNS, etc.

Big Data

- Real time processing of streaming data updates using Kinesis

Control systems

- Customize responses and workflows to state changes within AWS

Serverless backends

- Execute server-side backend logic

Building Blocks of AWS Lambda

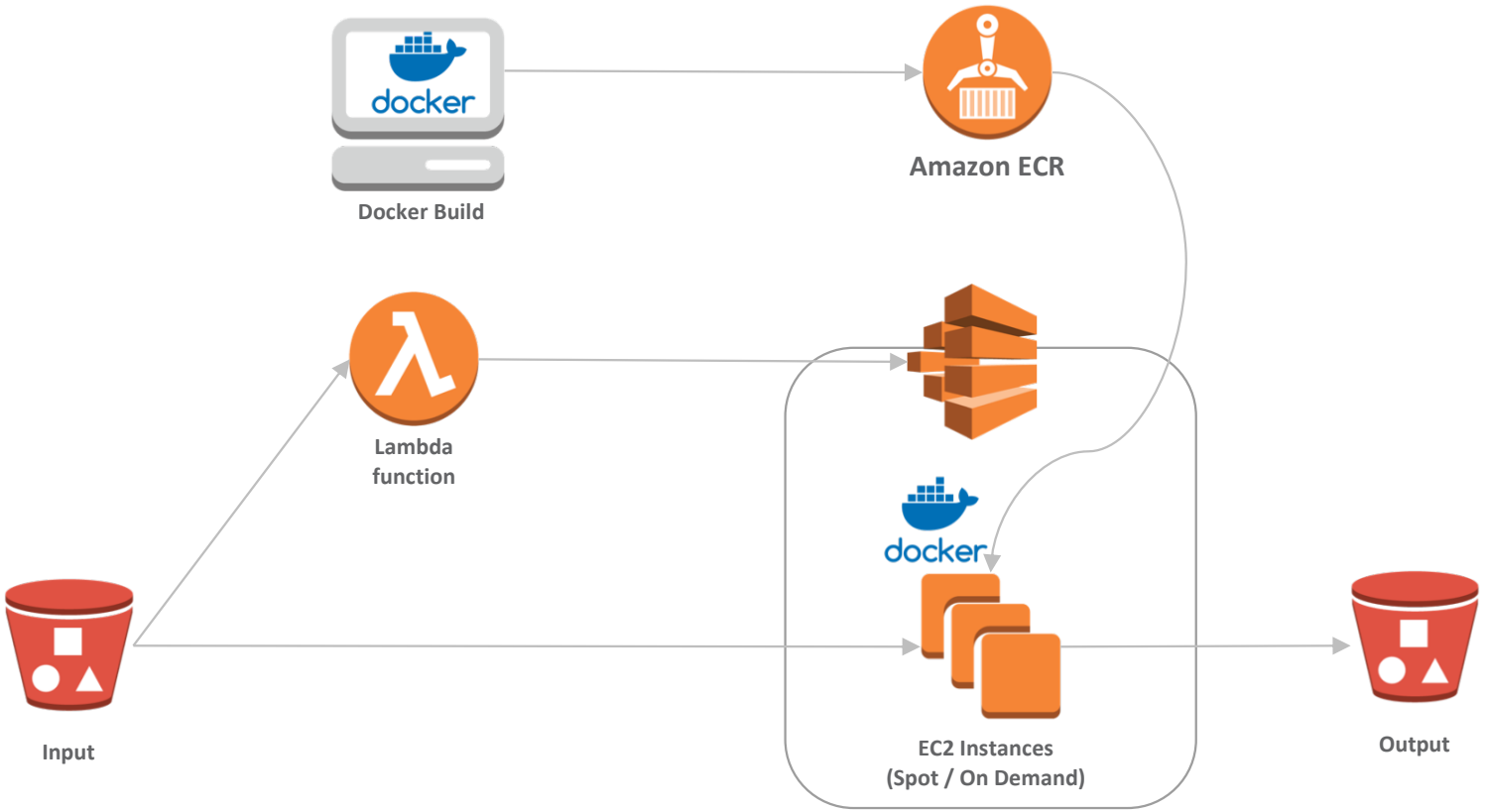
Lambda function: The foundation, it is comprised of your custom code and any dependent libraries. For more information, see Lambda Functions.
Event source: An AWS service, such as Amazon SNS, or a custom service, that triggers your function and executes its logic. For more information, see Event Source Mapping.
Downstream resources: An AWS service, such DynamoDB tables or Amazon S3 buckets, that your Lambda function calls once it is triggered.
Log streams: While Lambda automatically monitors your function invocations and reports metrics to CloudWatch, you can annotate your function code with custom logging statements that allow you to analyze the execution flow and performance of your Lambda function to ensure it's working properly.
AWS SAM: A model to define serverless applications. AWS SAM is natively supported by AWS CloudFormation and defines simplified syntax for expressing serverless resources. For more information, see Using the AWS Serverless Application Model (AWS SAM)

trivadis
makes IT easier.

AWS Lambda - Demo

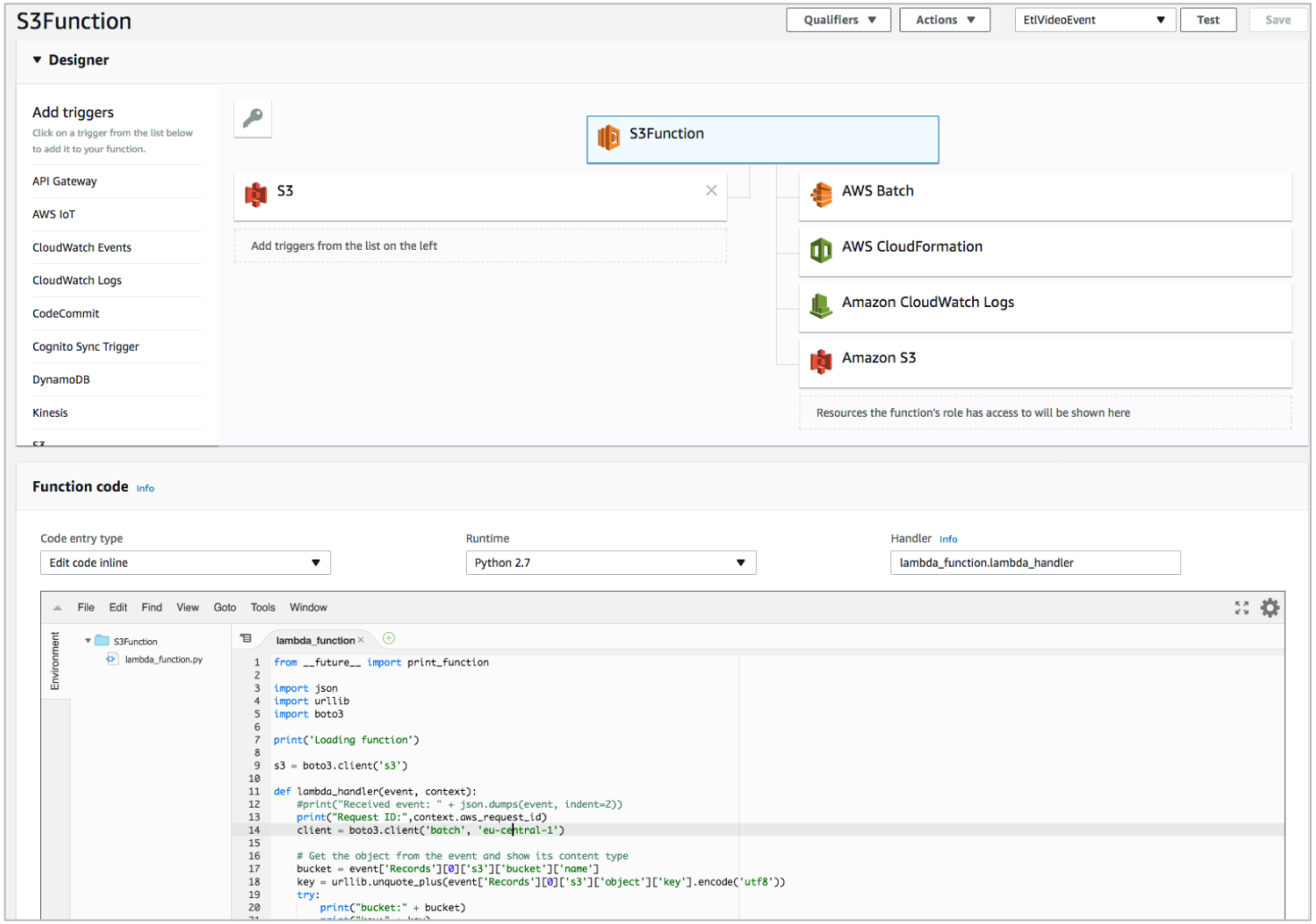
trivadis
makes IT easier.

Demo – 2nd step



trivadis
makes IT easier.

Lambda Implementation



trivadis
makes IT easier.

Lambda Implementation – Using Python

File Edit Find View Goto Tools Window Save Test

Environment

S3Function

lambda_function.py

lambda_function

```python
from __future__ import print_function

import json
import urllib
import boto3

print('Loading function')

s3 = boto3.client('s3')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    print("Request ID:",context.aws_request_id)
    client = boto3.client('batch', 'eu-central-1')

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.unquote_plus(event['Records'][0]['s3']['object']['key'].encode('utf8'))
    try:
        print("bucket:" + bucket)
        print("key:" + key)

        analyzer_job = client.submit_job(
                        jobName='ffmpeg' + context.aws_request_id,
                        jobQueue='audi-mds-queue',
                        jobDefinition='ffmpeg_job:6',
                        parameters={'inputFileName': 's3://' + bucket + '/' + key},
                        containerOverrides={
                                'command': ['Ref::inputFileName']
                        })
        theobjects = s3.list_objects_v2(Bucket="gus-audi-test", Prefix="videos/input")
        for object in theobjects["Contents"]:
            print(object["Key"])
        print ("analyzer jobName:" + analyzer_job['jobName'])
        print ("analyzer jobId:" + analyzer_job['jobId'])

    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

14:42 Python Spaces: 4

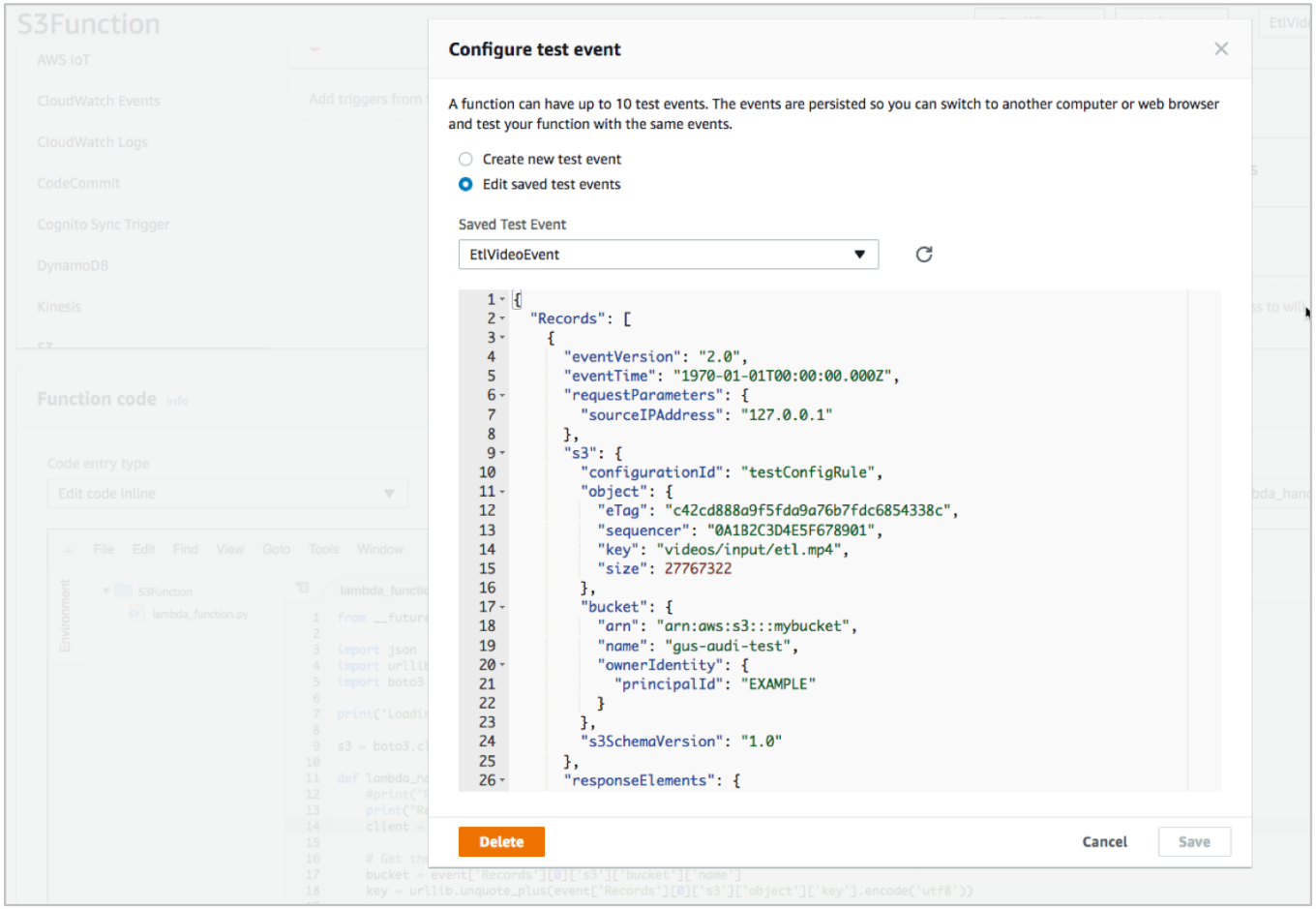
Execution Result

Execution results

No execution results yet

trivadis makes IT easier.

Lambda Implementation – Configure test event



trivadis
makes IT easier.

AWS Step Functions

trivadis
makes IT easier.

AWS Step Functions

I want to sequence functions

I want to run functions in parallel

I want to select functions based on data

I want to retry functions

I want to try/catch/finally

I have code that runs for hours

trivadis
makes IT easier.

AWS Step Functions - Concepts

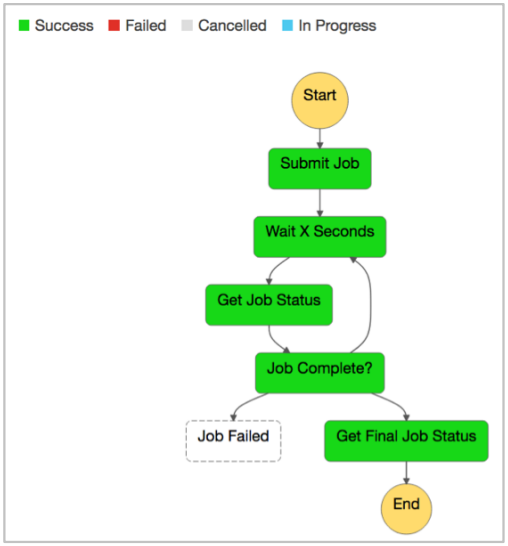
State Machine is defined using the [Amazon States Language]

A diagram is generated based on that structure

States

- Task
- Choice
- Succeed/Fail
- Pass
- Wait
- Parallel

```
1  {
2    "Comment": "A state machine that submits a Job to AWS Batch and
     monitors the Job until it completes.",
3    "StartAt": "Submit Job",
4    "States": {
5      "Submit Job": {
6        "Type": "Task",
7        "Resource": "arn:aws:lambda:eu-central-
     1:663559919114:function:StepFunctionsSample-JobStatusPol-
     SubmitJobFunction-I53U8EH0NBJX",
8        "ResultPath": "$.guid",
9        "Next": "Wait X Seconds"
10     },
11     "Wait X Seconds": {
12       "Type": "Wait",
13       "SecondsPath": "$.wait_time",
14       "Next": "Get Job Status"
```



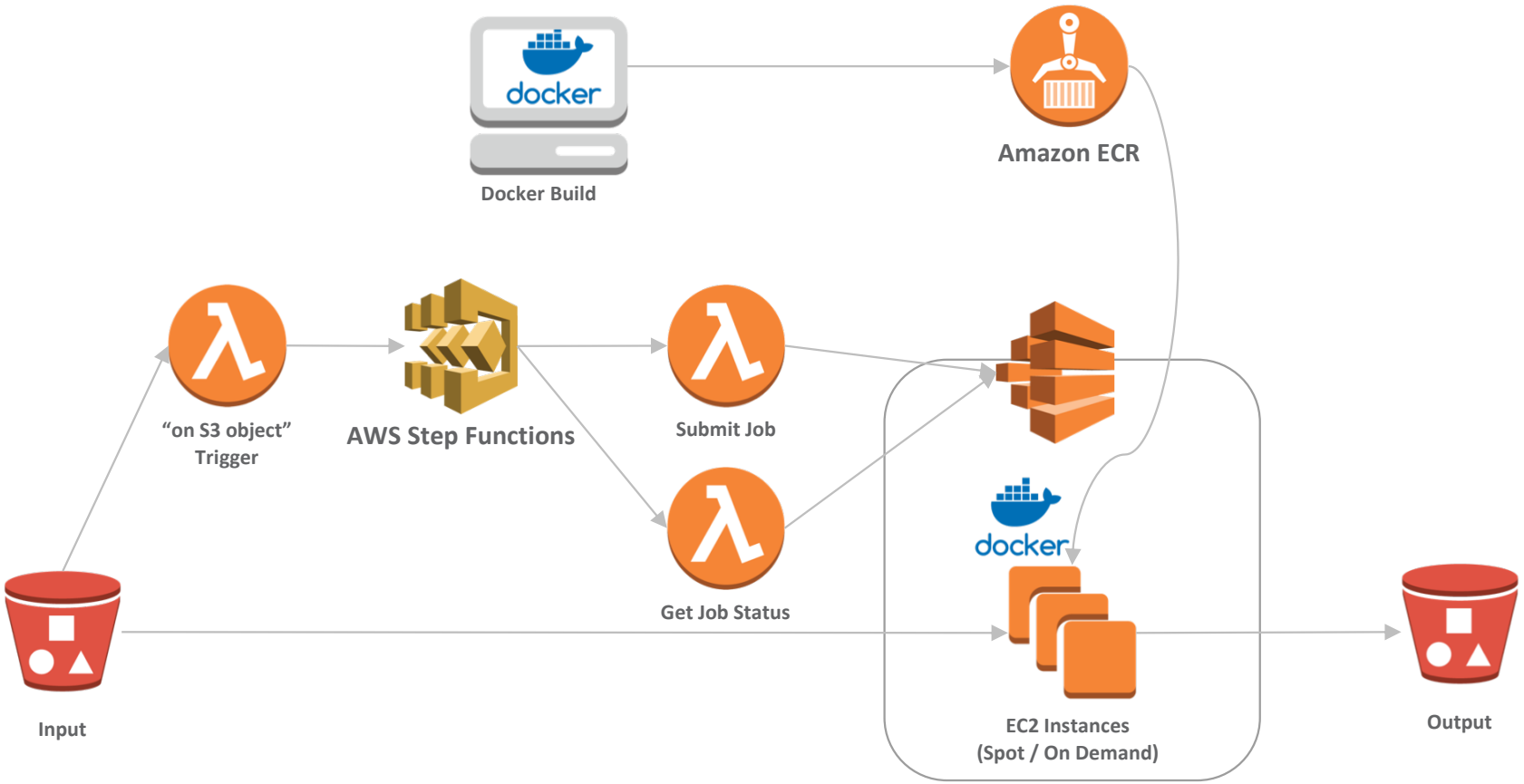
2.5 cents per thousand state transitions / 4000 free each month

trivadis
makes IT easier.

AWS Step Functions - Demo

trivadis
makes IT easier.

Demo – 3rd step



trivadis
makes IT easier.

AWS Batch Job Poller

Dashboard › JobStatusPollerStateMachine-9FerBhD59R1C

State Machine Arn: arn:aws:states:eu-central-1:663559919114:stateMachine:JobStatusPollerStateMachine-9FerBhD59R1C

JobStatusPollerStateMachine-9FerBhD59R1C

Copy to New | Edit state machine

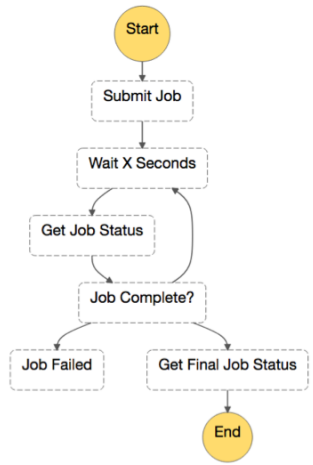
On this page you can add one or more executions for this state machine

Executions | State machine details

Code

```
1  {
2    "Comment": "A state machine that submits a Job to AWS Batch and monitors the Job
     until it completes.",
3    "StartAt": "Submit Job",
4    "States": {
5      "Submit Job": {
6        "Type": "Task",
7        "Resource": "arn:aws:lambda:eu-central-
     1:663559919114:function:StepFunctionsSample-JobStatusPol-SubmitJobFunction-
     I53U8EH0NBJX",
8        "ResultPath": "$.guid",
9        "Next": "Wait X Seconds"
10     },
11     "Wait X Seconds": {
12       "Type": "Wait",
13       "SecondsPath": "$.wait_time",
14       "Next": "Get Job Status"
15     },
16     "Get Job Status": {
17       "Type": "Task",
18       "Resource": "arn:aws:lambda:eu-central-
     1:663559919114:function:StepFunctionsSample-JobStatusPoll-CheckJobFunction-
     5X2P2552S97Z",
19       "Next": "Job Complete?",
20       "InputPath": "$.guid",
21       "ResultPath": "$.status"
22     },
```

Visual Workflow

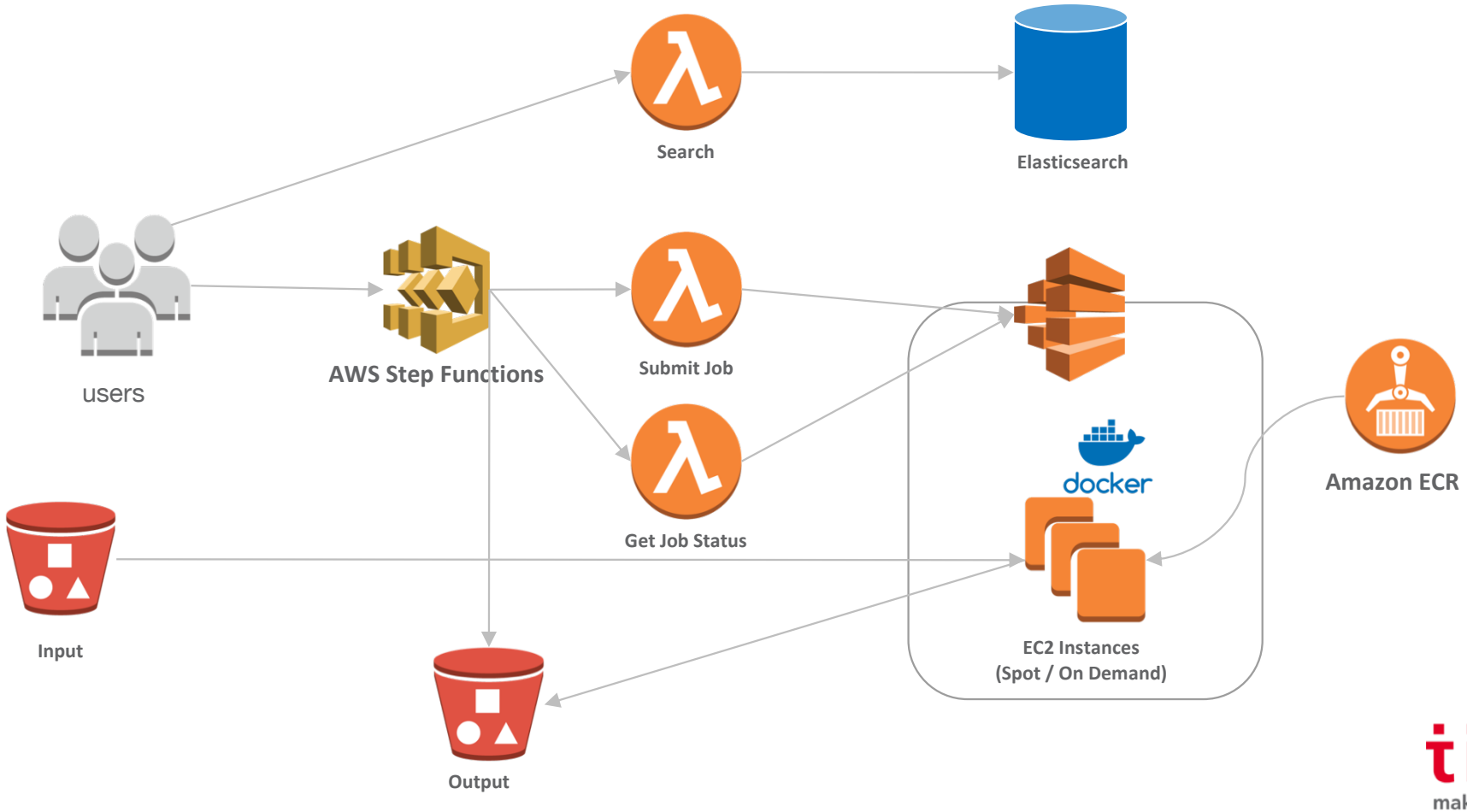


trivadis
makes IT easier.

Idea – Implementing Search

trivadis
makes IT easier.

Idea for implementing search with Lambda, Step Functions and AWS Batch



trivadis
makes IT easier.

Technology on its own won't help you.

You need to know how to use it properly.

trivadis
makes IT easier.