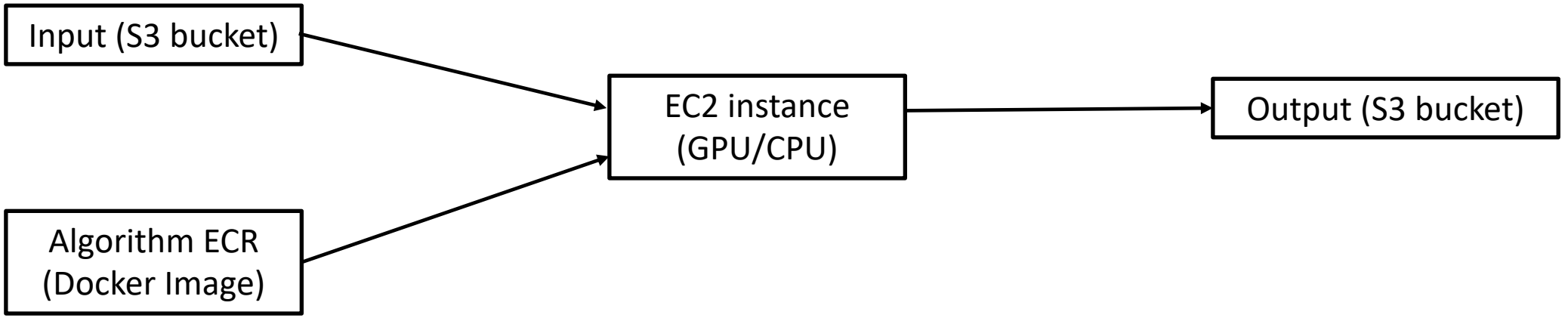
SageMaker Training Job in AWS

Abinash Pun

AISQUAD

Outline



- Amazon **S3** or **Amazon Simple Storage Service** is a service offered by Amazon Web Services that provides object storage through a web service interface.
- Amazon **Elastic Container Registry** (**Amazon ECR**) is an AWS managed container image registry service that is secure, scalable, and reliable.
- Amazon **Elastic Compute Cloud** (**EC2**) is the Amazon Web Service you use to create and run virtual machines in the cloud (we call these virtual machines 'instances').

Docker

- Software level virtualization
- creating image containing all the software dependencies to run the code
- **Build Docker**
  - Dockerfile: all the configuration
  - $ docker build –t image_name .
- **Push to ECR**
- ***build-and-push.sh****:* script to build and push docker image
- ***ecr_image_fullname.txt:*** contains image name with path (Do we always need to do this?)

Input and Output

- Initial Input: s3://../input
- E2:
  - Input: /opt/ml/input
    - Copy to /opt/ inside container (?)
  - Output: copy to /opt/ml/output
- Final output: s3://../output

https://us-east-1.console.aws.amazon.com/sagemaker/home?region=us-east-1#/jobs/create

Manual Setup: Amazon SageMaker > Training jobs > Create training job

- Job Setting
  - Job name:
  - IAM role: *SageMakerExecutionRole*
  - Algorithm Option
    - Algorithm source: *Your own algorithm container in ECR (?)*
    - Choose an Algorithm: *choose the image file* (?)
- Resource configuration
  - Instance type, Instance count, Additional storage volume per stance
  - Keep alive period (*waiting period?)*
  - Encryption key (for data)
- Stopping condition
  - Maximum runtime
- Network
  - Enable network isolation
  - VPC (Virtual Private Cloud)
- Input Data Configuration
  - Channels: train
  - Data source (*S3*), S3 Data type, S3 Data location
- Checkpoint configuration
- Output Data Configuration
- Managed Spot Training
- Tags

Setup via Script: *training_job.py*

```python
import os
from urllib.parse import urlparse
import io
import boto3
import json
import jsonlines
import sagemaker
import pandas as pd
import numpy as np
from PIL import Image
from itertools import cycle, islice
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from sagemaker.estimator import Framework, Estimator
from sagemaker.processing import Processor, ProcessingInput, ProcessingOutput

sagemaker_session = sagemaker.Session()

# we are using the notebook instance role for training in this example
#role = 'AmazonSageMaker-ExecutionRole-833537904510'
role = sagemaker.get_execution_role(sagemaker_session=sagemaker_session)
# you can specify a bucket name here, we're using the default bucket of SageMaker
bucket = "sagemaker-studio-833537904510-3lvvbxobayc"

print(role)
print(bucket)

with open (os.path.join('ecr_image_fullname.txt'), 'r') as f:
    container = f.readlines()[0][:-1]

print(container)
s3_input = 's3://{}/auto_synth'.format(bucket)
s3_autobox_input = '{}/auto_box'.format(s3_input)
s3_synth_data_input = '{}/synth_data'.format(s3_input)

print(s3_autobox_input)
print(s3_synth_data_input)

cfg='{}/cfg/'.format(s3_input)
outpath='{}/results/'.format(s3_input)
images='{}/images/'.format(s3_autobox_input)
labels='{}/labels/'.format(s3_autobox_input)
background='{}/background/'.format(s3_synth_data_input)
bg_noise='{}/bg_noise/'.format(s3_synth_data_input)
bg_masks='{}/bg_masks/'.format(s3_synth_data_input)

print(cfg)
print(outpath)
print(images)
print(labels)
print(background)
print(bg_noise)
print(bg_masks)

# Job configuration
from sagemaker.session import TrainingInput

inputs = {
    "cfg": TrainingInput(cfg),
    "images": TrainingInput(images),
    "labels": TrainingInput(labels),
    "background": TrainingInput(background),
    "bg_noise": TrainingInput(bg_noise),
    "bg_masks": TrainingInput(bg_masks),
}

estimator = Estimator(
    image_uri=container,
    role=role,
    instance_count=1,
    instance_type='ml.m5.large',
    # instance_type='local',
    input_mode='File',
    output_path=outpath,
    base_job_name='auto-synth-test1'
)

estimator.fit(inputs)
```

Some Details

auto_box / Dockerfile Info

```
1  ARG BASE_IMG=${BASE_IMG}
2  #ARG BASE_IMG='pytorch/pytorch:latest'
3  FROM ${BASE_IMG}
4  RUN mkdir -p /opt
5  ENV PATH="/opt:${PATH}"
6  RUN apt-get update \
7   && apt-get install build-essential -y --no-install-recommends --allow-unauthenticated \
8      jq git gcc libgl1-mesa-dev wget gsutil libglib2.0-0 cmake
9  ## fix /usr/local/cuda-10.0/compat/libcuda.so
10 ## RUN bash -c 'echo "/usr/local/cuda-10.0/compat" > /etc/ld.so.conf.d/cuda.conf'
11 RUN ldconfig -v
12 #RUN pip install tensorboard torch torchvision --upgrade
13
14 COPY requirements.txt /opt/requirements.txt
15 COPY . /opt
16 RUN pip install -r /opt/requirements.txt
17 RUN cd /opt
18 WORKDIR /opt
19 RUN chmod +x train
20 ## https://github.com/aws/sagemaker-pytorch-training-toolkit/issues/143#issuecomment-566776288
21 ## https://github.com/aws/sagemaker-pytorch-training-toolkit/blob/upgrade-training-
   toolkit/docker/build_artifacts/start_with_right_hostname.sh
22 ## https://github.com/aws/deep-learning-containers/blob/v2.0-pt-1.5.1-
   py36/pytorch/training/docker/1.5.1/py3/Dockerfile.gpu#L181
23 COPY start_with_right_hostname.sh /usr/local/bin/start_with_right_hostname.sh
```

auto_box / start_with_right_hostname.sh Info

```
1  #!/usr/bin/env bash
2  if [[ "$1" = "train" ]]; then
3       CURRENT_HOST=$(jq .current_host  /opt/ml/input/config/resourceconfig.json)
4       sed -ie "s/PLACEHOLDER_HOSTNAME/$CURRENT_HOST/g" changehostname.c
5       gcc -o changehostname.o -c -fPIC -Wall changehostname.c
6       gcc -o libchangehostname.so -shared -export-dynamic changehostname.o -ldl
7       LD_PRELOAD=/opt/code/libchangehostname.so train
8  else
9       eval "$@"
10 fi
```

Some Details

auto_box / build-and-push.sh Info

```
#!/bin/bash
image=$1
if [ "$image" == "" ]
then
    echo "Usage: $0 <image-name>"
    exit 1
fi
account=$(aws sts get-caller-identity --query Account --output text)
region=$(aws configure get region)
fullname="${account}.dkr.ecr.${region}.amazonaws.com/${image}:latest"
aws ecr describe-repositories --repository-names "${image}" > /dev/null 2>&1
if [ $? -ne 0 ]
then
    aws ecr create-repository --repository-name "${image}" > /dev/null
fi
# Get the login command from ECR and execute it directly (for our own registry and for the base image registry)
aws ecr get-login-password --region ${region} | docker login --username AWS --password-stdin
${account}.dkr.ecr.${region}.amazonaws.com
base_img='pytorch/pytorch:latest'
echo 'base_img:'$base_img
# Build the docker image locally and then push it to ECR with the full name.
cd container
echo "Building image with name ${image}"
docker build --no-cache -t ${image} -f Dockerfile --build-arg BASE_IMG=$base_img .
docker tag ${image} ${fullname}
echo "Pushing image to ECR ${fullname}"
docker push ${fullname}
# Writing the image name to let the calling process extract it without manual intervention:
echo "${fullname}" > ecr_image_fullname.txt
```