
AWS
re:Invent

AIM317

Build consistent and portable ML environments with containers

Antje Barth

Sr. Developer Advocate, AI and Machine Learning

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Use case – let’s build a BERT-based NLP classifier model

Why machine learning (ML) with containers?

Using ML containers with Amazon Elastic Kubernetes Service (EKS)

Using ML containers with Amazon SageMaker

Demo

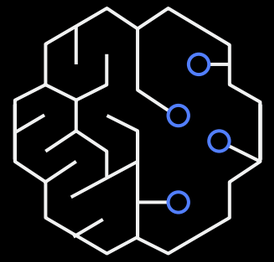


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Meet Sarah, a data scientist



- Recently started as a data scientist at Amazon
- Joined Customer Reviews team
- First project – build BERT-based NLP model to classify customer reviews from amazon.com



“I simply love this product.”



Sarah's workflow

Let's look for training data

I heard about pre-trained BERT models; I'm going to use one, and fine-tune it to my data

Start prototyping
…

OK results – I need more training resources

```
US REVIEWS DATASET:
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Wireless_v1_00.tsv.gz
https://s3.amazonaws.com/amazon_reviews-pds/tsv/amazon_reviews_us_Watches_v1_00.tsv.gz
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Video_Games_v1_00.tsv.gz
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Video_DVD_v1_00.tsv.gz
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Video_v1_00.tsv.gz
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Toys_v1_00.tsv.gz
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Tools_v1_00.tsv.gz
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Sports_v1_00.tsv.gz
https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Software_v1_00.tsv.gz
```



Jupyter notebook



Training cluster

But…it worked on my machine. Why not here?

Why ML with containers?



AWS re:Invent

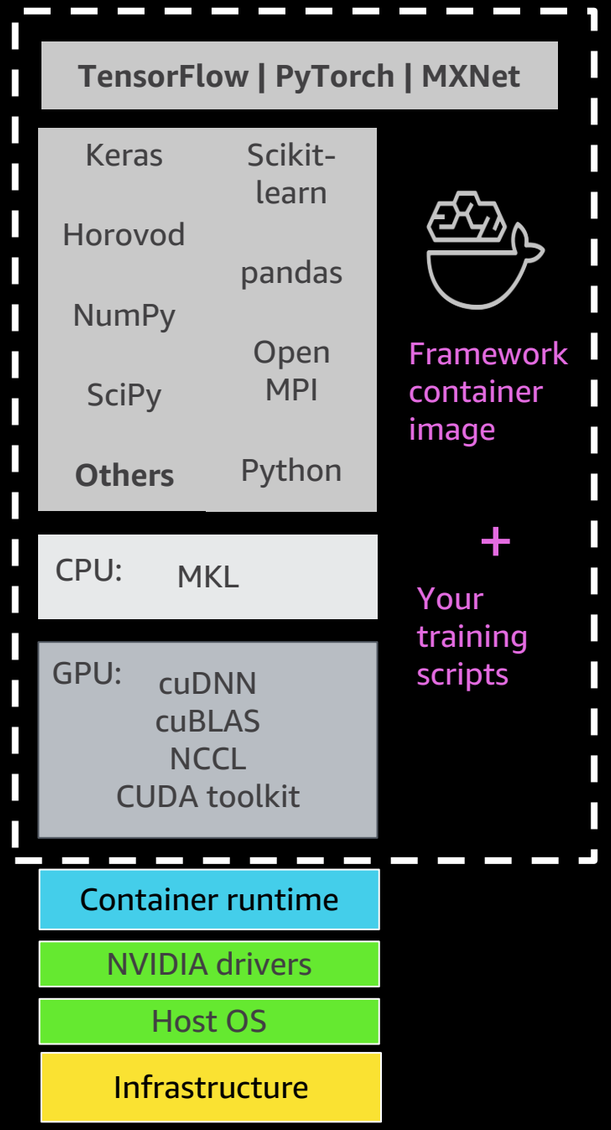
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why ML with containers?



Why ML with containers?





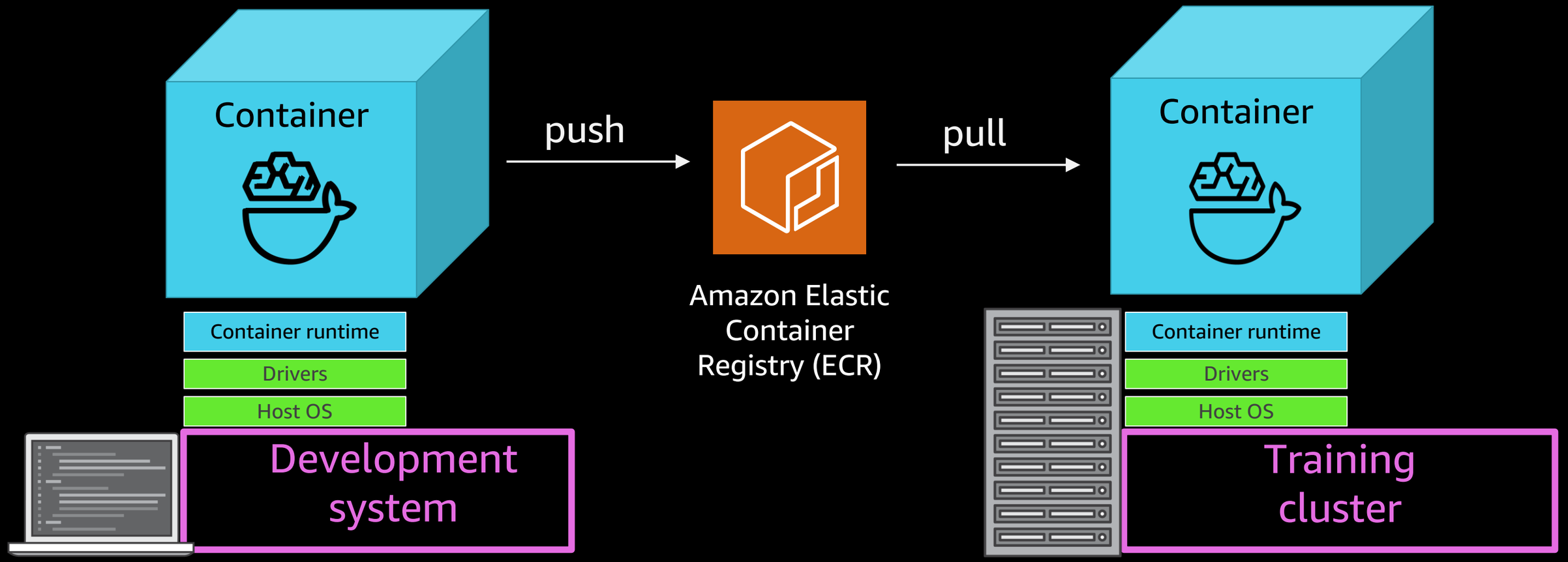
Packages

- Training code
- Dependencies
- Configurations

ML environments that are

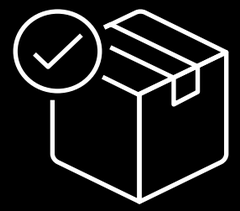
- Lightweight
- Portable
- Scalable
- Consistent

Why ML with containers?

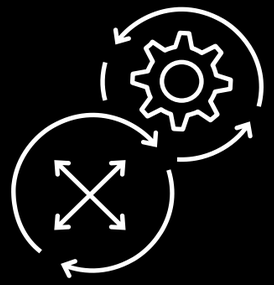


AWS Deep Learning Containers

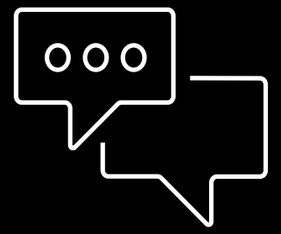
OPTIMIZED AND CUSTOMIZABLE CONTAINERS FOR DEEP LEARNING ENVIRONMENTS



Pre-packaged Docker container images, fully configured and validated



Best performance and scalability without tuning



Works with Amazon SageMaker, Amazon EKS, Amazon Elastic Container Service (ECS), and Amazon EC2

Key features

Customizable container images

Support for

TensorFlow PyTorch mxnet

Single and multi-node training and inference

TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.

ML container options

Image registry
Container image repository

Amazon ECR

Customer-provided containers

AWS Deep Learning Containers

1

Amazon ECS

Amazon EKS

2

Amazon SageMaker

Compute
Where the containers run

G4 ··· P3 ··· C5 ··· C5n



1

Using containers with Amazon EKS



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using customer-provided containers with Amazon EKS



Local IDE
or hosted
notebook

train.py
Dockerfile

Docker build



Docker push

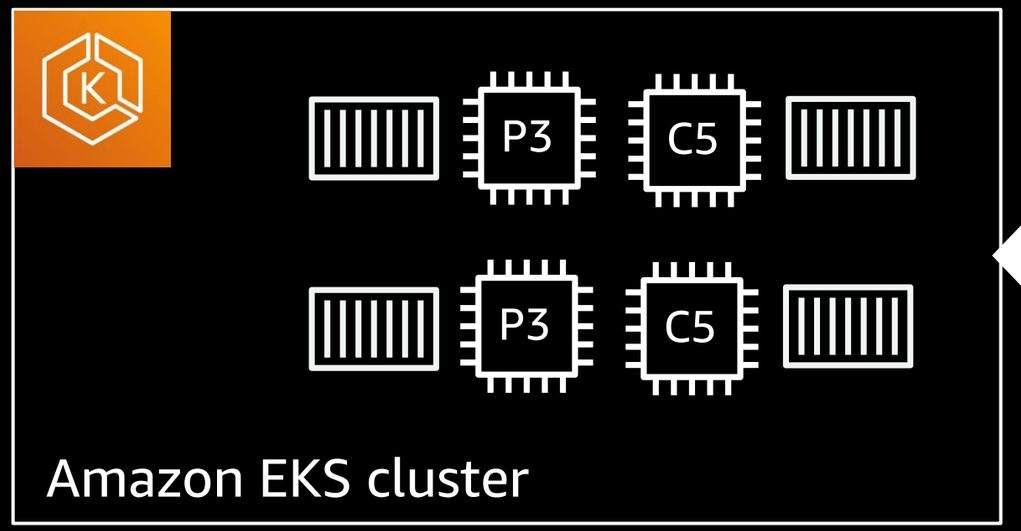


Amazon ECR

Container
registry

Approach

1. Provision EKS cluster
2. Develop your code
3. Build custom container image
4. Create K8s job YAML file
5. Apply training job YAML file

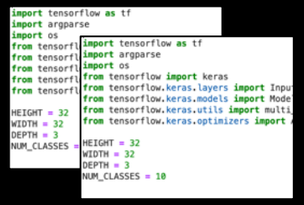


kubectl create
–f train.yaml

Using AWS-provided containers with Amazon EKS



Local IDE or hosted notebook



Code files

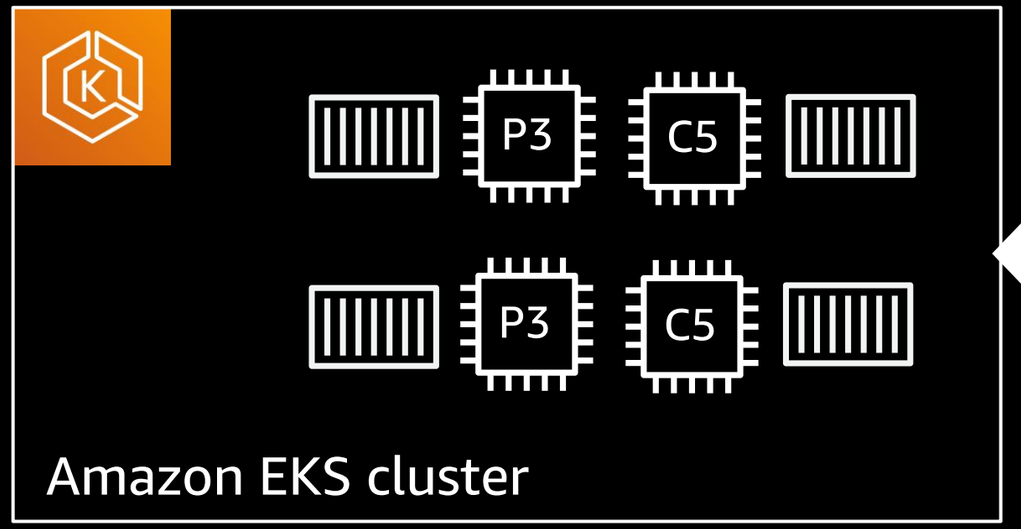
AWS Deep Learning Container

Amazon ECR

Container registry

Approach

1. Provision EKS cluster
2. Develop your code
3. Create K8s job YAML file referencing pre-built container
4. Apply training job YAML file



kubectl create –f train.yaml

Sample train.yaml

train.yaml

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: bert-model-training
spec:
  volumes:
  - name: fsx-opt-ml
    persistentVolumeClaim:
      claimName: fsx-claim
  containers:
    - name: bert
      command:
        - python
        - /opt/ml/code/train.py
        - --train_steps_per_epoch=100
        - --epochs=10
        - --learning_rate=0.00001
        - --epsilon=0.00000001
        - --train_batch_size=128
        - --validation_batch_size=64
        - --test_batch_size=64
        - --validation_steps=10
        - --test_steps=10
        - --use_xla=True
        - --use_amp=False
        - --max_seq_length=64
        - --freeze_bert_layer=True
        - --run_validation=True
        - --run_test=True
        - --run_sample_predictions=True
      image: 763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training:2.1.0-cpu-py36-ubuntu18.04
      imagePullPolicy: Always
      securityContext:
        privileged: true
      volumeMounts:
      - mountPath: /opt/ml/
        name: fsx-opt-ml
  restartPolicy: Never
```

Training code

Command line arguments

Container image

Amazon FSx mount

used

used

train.py

```python
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, epsilon=epsilon)
print('** use_amp {}'.format(use_amp))
if use_amp:
    # loss scaling is currently required when using mixed precision
    optimizer = tf.keras.mixed_precision.experimental.LossScaleOptimizer(optimizer, 'dynamic')

print('*** OPTIMIZER {} ***'.format(optimizer))

loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')

model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
print('Compiled model {}'.format(model))
model.layers[0].trainable = not freeze_bert_layer
print(model.summary())
```

```python
# Model Output
transformer_fine_tuned_model_path = os.path.join(local_model_dir, 'transformers/fine-tuned/')
os.makedirs(transformer_fine_tuned_model_path, exist_ok=True)

# SavedModel Output
tensorflow_saved_model_path = os.path.join(local_model_dir, 'tensorflow/saved_model/0')
os.makedirs(tensorflow_saved_model_path, exist_ok=True)
```

2

Using containers with Amazon SageMaker



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon SageMaker model training options

Increasing level of customization

Use built-in algorithms (container and script)

Amazon SageMaker comes with dozens of built-in algorithms for

- Classification
- Working with text
- Sequence translation
- Computer vision
- Recommendation
- Regression
- Anomaly detection
- Topic modeling
- Forecasting
- Clustering
- Feature reduction

1 Use pre-built framework containers

Bring Your Own Script

TensorFlow

PyTorch

mxnet

Keras

GLUON

scikit learn

HOROVOD

dmlc XGBoost

2 Use customer-provided containers

Bring Your Own Script
Bring Your Own Container



TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.

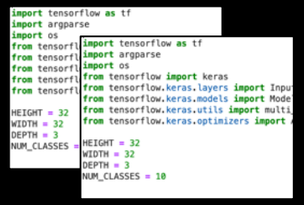
1 Using AWS-provided containers



Bring Your Own Script

Local IDE
or hosted
notebook

SageMaker SDK



Code files

AWS
framework
container

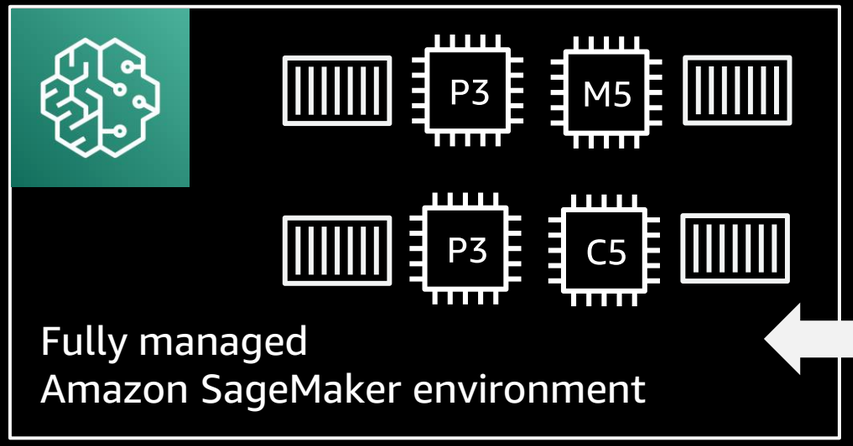


Amazon ECR

```
from sagemaker.tensorflow import TensorFlow

estimator = TensorFlow(entry_point='train.py',
                       source_dir='src',
                       role=sagemaker.get_execution_role(),
                       instance_count=1,
                       instance_type='p3.2xlarge',
                       volume_size=1024,
                       py_version='py3',
                       framework_version='2.1.0',
                       hyperparameters=hyperparameters,
                       input_mode='Pipe')

estimator.fit(inputs={'train': s3_input_train_data,
                      'validation': s3_input_validation_data,
                      'test': s3_input_test_data})
```





Amazon S3

2 Using customer-provided containers



Bring Your Own Container

Local IDE
or hosted
notebook

SageMaker SDK

Docker build



AWS CodeBuild



Docker push



Amazon ECR

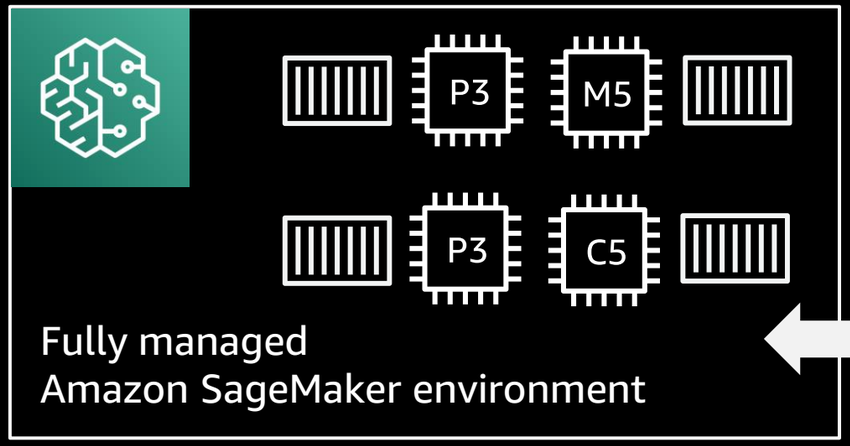
```
!$(aws ecr get-login --region ${region} --no-include-email)

!docker build -t ${algorithm_name} .
!docker tag ${algorithm_name} ${fullname}
!docker push ${fullname}

from sagemaker.estimator import Estimator

estimator = Estimator(image_uri=${fullname},
                      role=sagemaker.get_execution_role(),
                      base_job_name='custom-container'
                      instance_count=1,
                      instance_type='p3.2xlarge')

estimator.fit(inputs={'train': s3_input_train_data,
                      'validation': s3_input_validation_data,
                      'test': s3_input_test_data})
```





Amazon S3

/ aim317 /
Name
00_setup_eks
code
img
input
model
00_Prepare_Dataset_BERT.ipynb
01_Develop_Code_Notebook.ip...
02_Run_ML_Training_EKS.ipynb
03_Run_ML_Training_SageMak...
README.md
train.yaml

Amazon Customer Reviews Dataset

https://s3.amazonaws.com/amazon-reviews-pds/readme.html

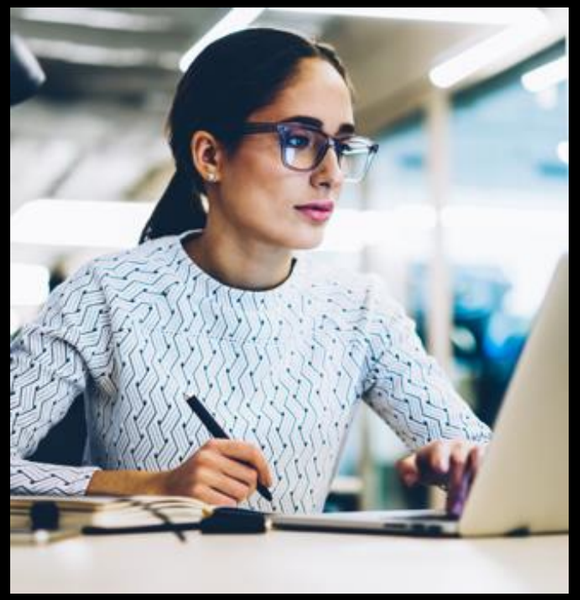
Dataset Columns:

- `marketplace` 2-letter country code (in this case all "US").
- `customer_id` Random identifier that can be used to aggregate reviews written by a single author.
- `review_id` A unique ID for the review.
- `product_id` The Amazon Standard Identification Number (ASIN). http://www.amazon.com/dp/ links to the product's detail page.
- `product_parent` The parent of that ASIN. Multiple ASINs (color or format variations of the same product) can roll up into a single parent.
- `product_title` Title description of the product.
- `product_category` Broad product category that can be used to group reviews (in this case digital videos).
- `star_rating` The review's rating (1 to 5 stars).
- `helpful_votes` Number of helpful votes for the review.
- `total_votes` Number of total votes the review received.
- `vine` Was the review written as part of the Vine program?
- `verified_purchase` Was the review from a verified purchase?
- `review_headline` The title of the review itself.
- `review_body` The text of the review.
- `review_date` The date the review was written.

Feature Transformation

Presentation Deep Dive on BERT:

Sarah’s new workflow



- ✓ Look for training data
- ✓ Use a pre-trained NLP model
- ✓ Develop custom model code
- ✓ Leverage pre-built deep learning framework containers to have consistent environment across platforms
- ✓ Run model training at scale using Amazon SageMaker
- Leverage Managed Spot Training to save up to 90% over on-demand instances

Summary

- Containers let you build lightweight, portable, and consistent ML environments
- AWS Deep Learning Containers include frameworks optimized by experts to deliver the best performance on CPUs and GPUs
- Build self-managed ML platforms with Amazon EKS and compatible ML toolkits
- Leverage Amazon SageMaker as a fully managed service to build, train, and deploy ML models quickly

Get started

AWS Deep Learning Containers

https://github.com/aws/deep-learning-containers

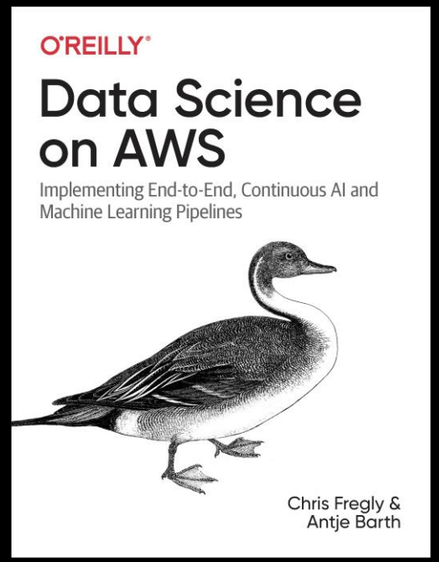
https://docs.aws.amazon.com/deep-learning-containers/latest/devguide/deep-learning-containers-images.html

https://docs.aws.amazon.com/sagemaker/latest/dg/docker-containers.html

Amazon SageMaker
https://aws.amazon.com/sagemaker/

Demo code
https://github.com/data-science-on-aws/workshop

Book
https://www.datascienceonaws.com/



Thank you!

Antje Barth

Twitter: @anbarth

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.