AWS re:Invent

AIM402-R

Deep Learning with PyTorch

Kris Skrinak
Sr. AI/ML Specialist PSA
Amazon Web Services

Chaitanya Hazarey
AI/ML Specialist PSA
Amazon Web Services

Agenda

- State of the art in NLP using SageMaker
- SageMaker PyTorch Support
- Lab [1] – Word Level Language Modeling Using PyTorch
- Train On Premise – Deploy in the Cloud
- Lab [2] – Toxic Comments Classification using Hugging Face Transformers and SageMaker
- Bonus Lab – CAPTUM – PyTorch Deep Learning Models Interpretability

Related sessions

AIM402-R1 [REPEAT] Deep learning with PyTorch

AIM412-R Deep learning applications using PyTorch, featuring Autodesk

AIM412-R1 Deep learning applications with PyTorch, featuring Freshworks

AIM407-R Amazon SageMaker and PyTorch: Tips & tricks

AIM407-R1 - [REPEAT] Amazon SageMaker and PyTorch: Tips & tricks

State of the art in NLP using SageMaker

AWS re:Invent

aws

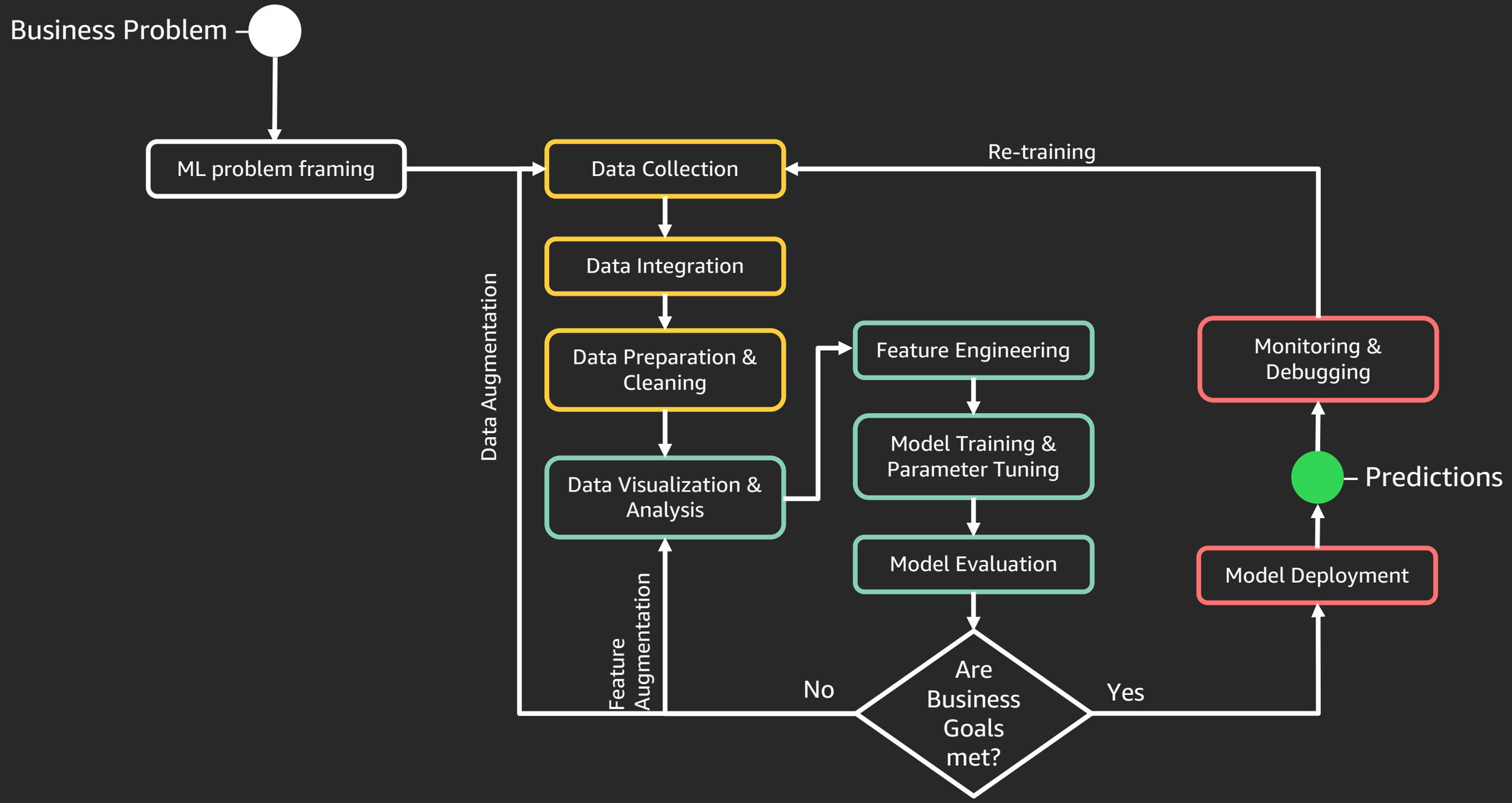
State of the art in NLP

- Essential components of NLP
  - Tokenize
  - Remove Stop Words
  - Lemmatize
  - Replace Rare Words
  - Embeddings
- Evolution of NLP from RNNs to Transformers
  - Seq2Seq
  - Attention
  - Transformer

Source:

- A Deep Dive into NLP with PyTorch – PyData London 2019
- https://www.youtube.com/watch?v=4jROlXH9Nvc

The Machine Learning Process



THE AWS ML STACK

Broadest and deepest set of capabilities

AI Services

VISION			SPEECH		LANGUAGE		CHATBOTS	FORECASTING	RECOMMENDATIONS
REKOGNITION IMAGE	REKOGNITION VIDEO	TEXTRACT	POLLY	TRANSCRIBE	TRANSLATE	COMPREHEND & COMPREHEND MEDICAL	LEX	FORECAST	PERSONALIZE

ML Services

Amazon SageMaker

Ground Truth	Notebooks	Algorithms + Marketplace	Reinforcement Learning	Training	Optimization	Deployment	Hosting

ML Frameworks + Infrastructure

FRAMEWORKS	INTERFACES
TensorFlow, mxnet, PYTORCH	GLUON, Keras

INFRASTRUCTURE							
EC2 P3 & P3DN	EC2 G4	EC2 C5	FPGAS	DL CONTAINERS & AMIs	GREENGRASS	ELASTIC INFERENCE	INFERENTIA

AMAZON SAGEMAKER

Bringing machine learning to all developers

Pre-built notebooks for common problems

Collect and prepare training data

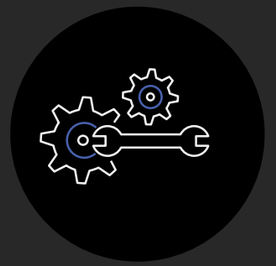
Built-in, high performance algorithms

Choose and optimize your ML algorithm

One-click training

Set up and manage environments for training

Optimization

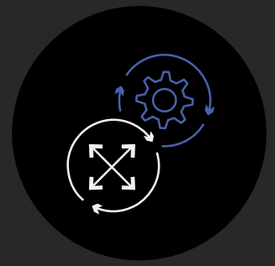


Train and tune model (trial and error)

One-click deployment

Deploy model in production

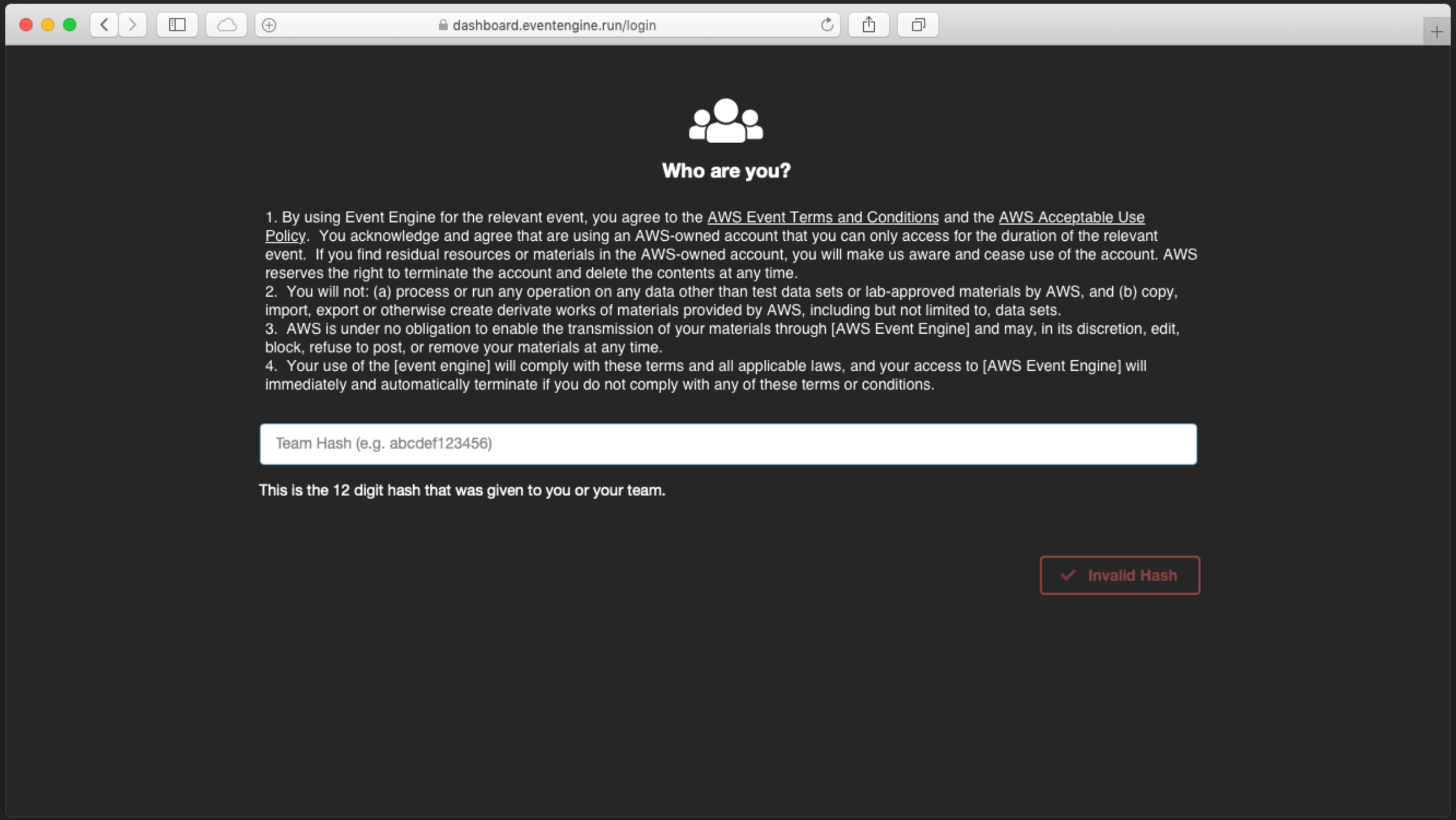
Fully managed with auto-scaling, health checks, automatic handling of node failures, and security checks

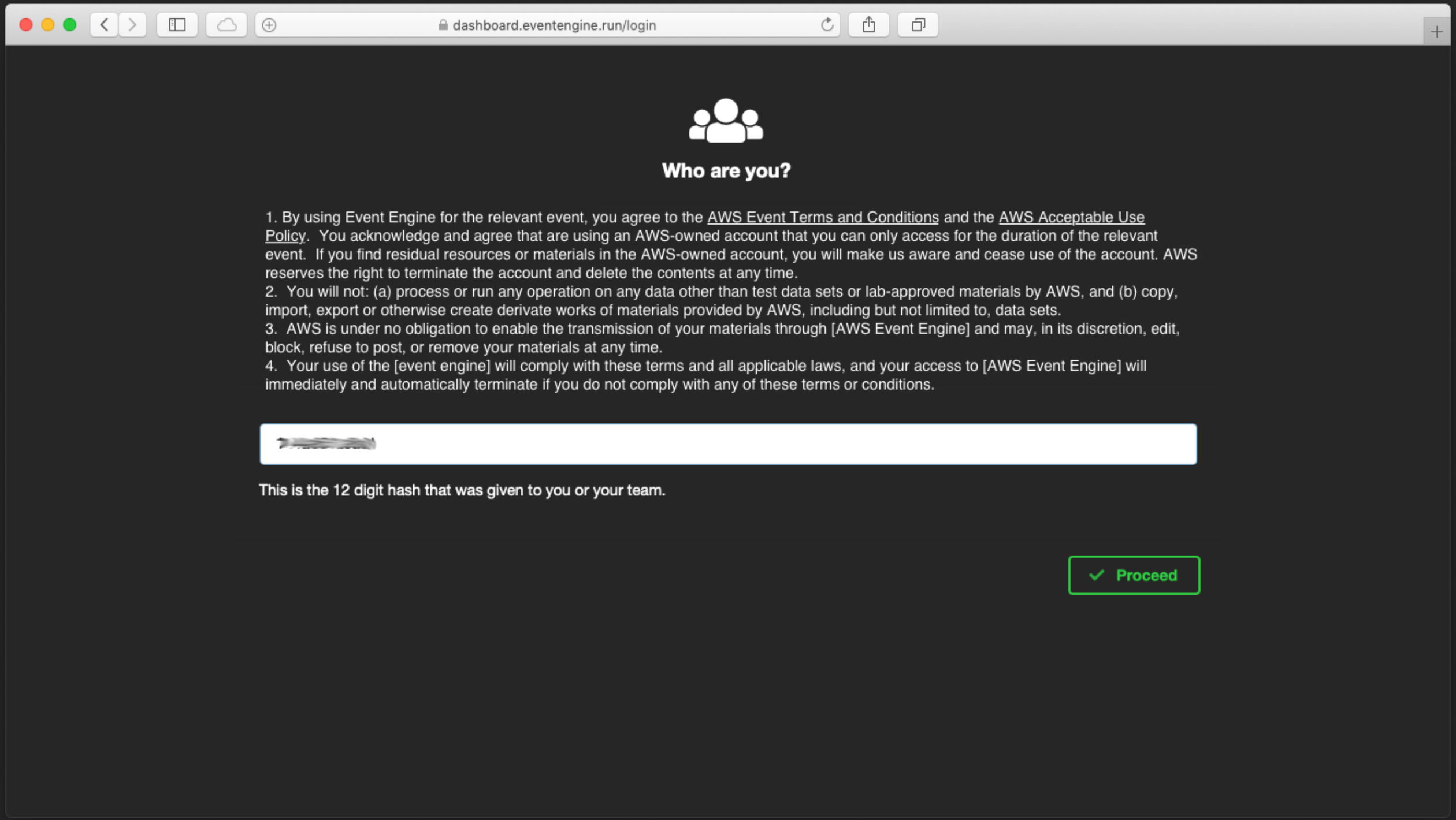


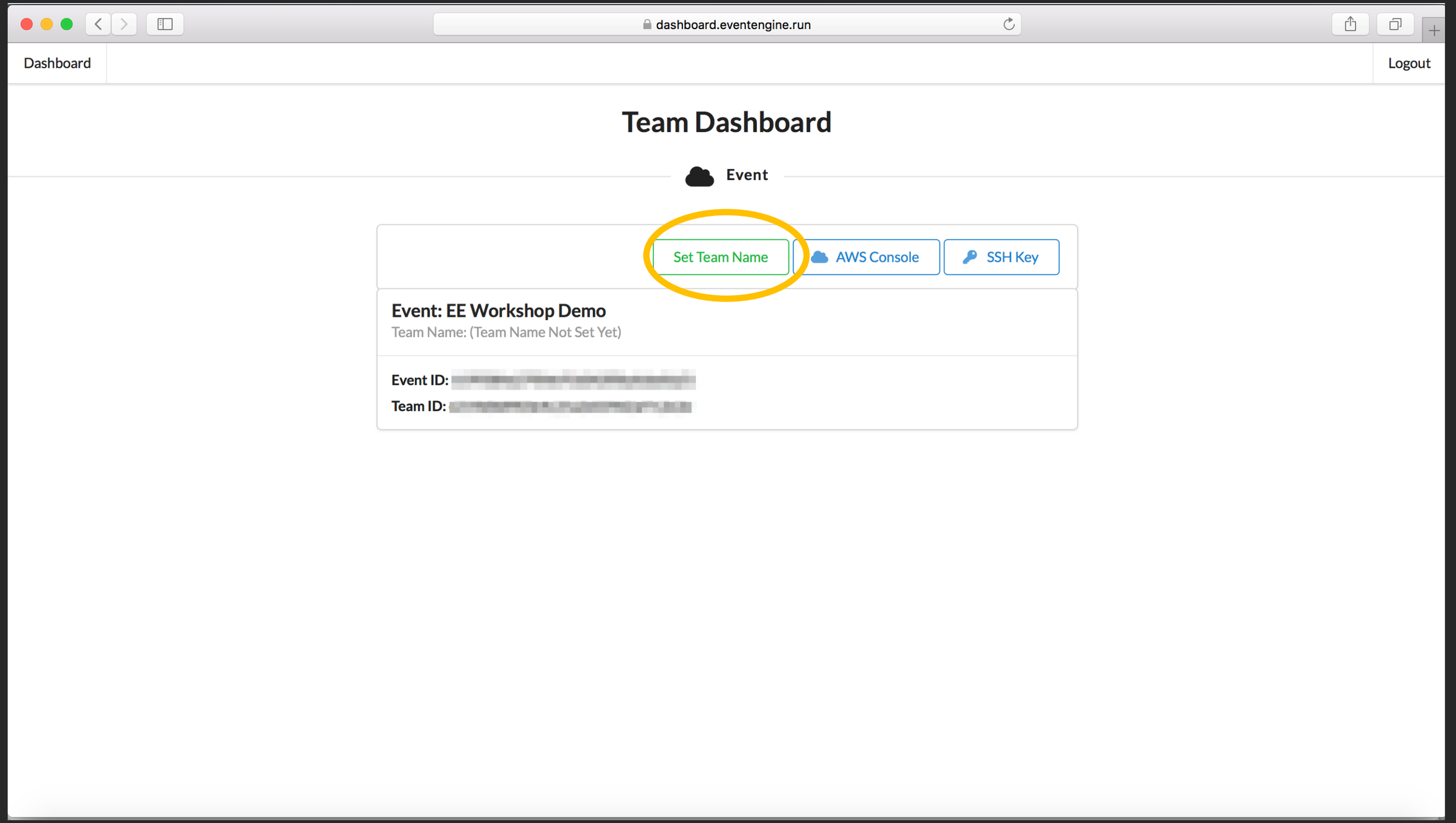
Scale and manage the production environment

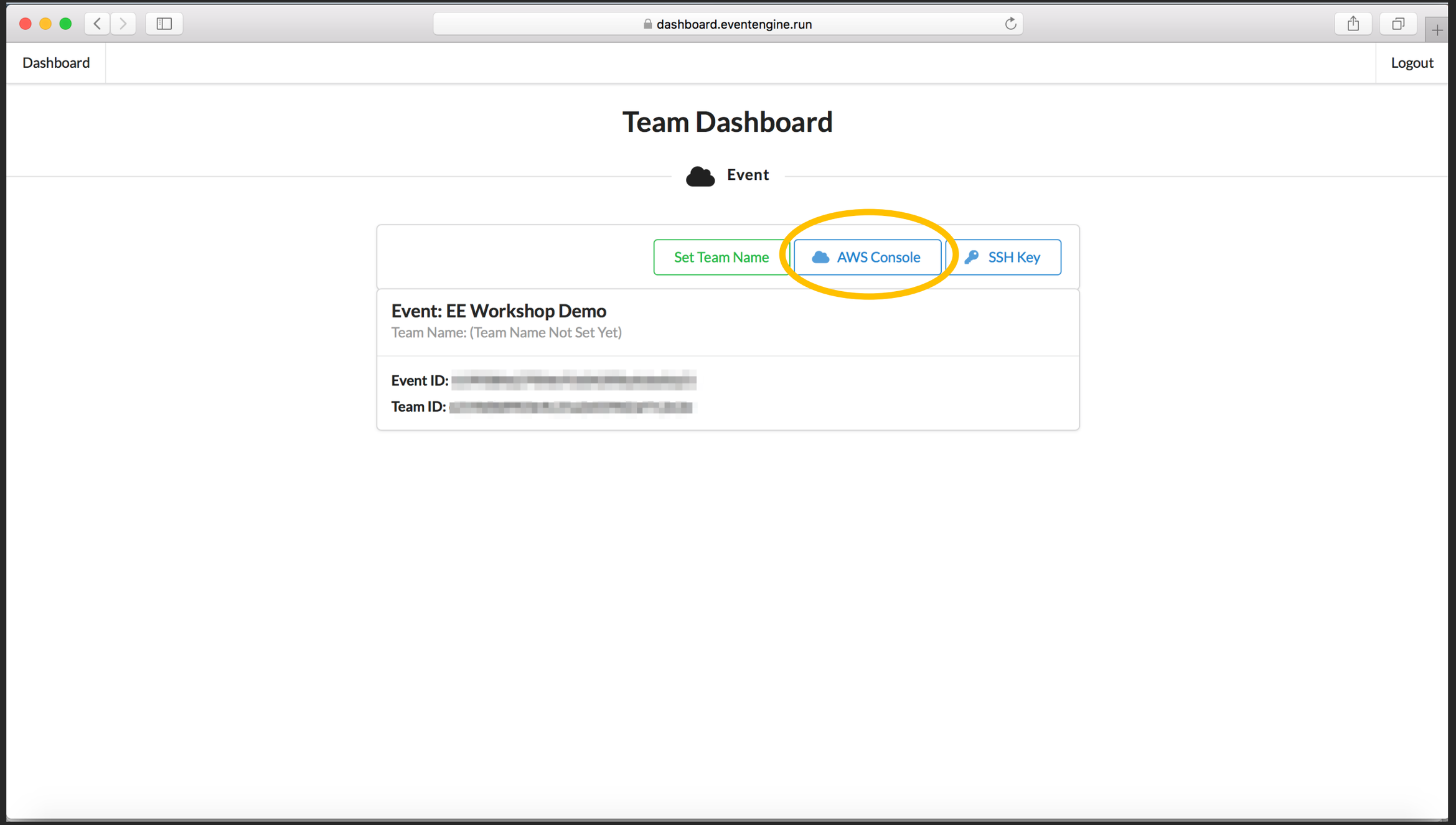
Launch SageMaker

https://dashboard.eventengine.run


dashboard.eventengine.run/login
Who are you?
1. By using Event Engine for the relevant event, you agree to the AWS Event Terms and Conditions and the AWS Acceptable Use Policy. You acknowledge and agree that are using an AWS-owned account that you can only access for the duration of the relevant event. If you find residual resources or materials in the AWS-owned account, you will make us aware and cease use of the account. AWS reserves the right to terminate the account and delete the contents at any time.
2. You will not: (a) process or run any operation on any data other than test data sets or lab-approved materials by AWS, and (b) copy, import, export or otherwise create derivate works of materials provided by AWS, including but not limited to, data sets.
3. AWS is under no obligation to enable the transmission of your materials through [AWS Event Engine] and may, in its discretion, edit, block, refuse to post, or remove your materials at any time.
4. Your use of the [event engine] will comply with these terms and all applicable laws, and your access to [AWS Event Engine] will immediately and automatically terminate if you do not comply with any of these terms or conditions.
Team Hash (e.g. abcdef123456)
This is the 12 digit hash that was given to you or your team.
Invalid Hash


dashboard.eventengine.run/login
Who are you?
1. By using Event Engine for the relevant event, you agree to the AWS Event Terms and Conditions and the AWS Acceptable Use Policy. You acknowledge and agree that are using an AWS-owned account that you can only access for the duration of the relevant event. If you find residual resources or materials in the AWS-owned account, you will make us aware and cease use of the account. AWS reserves the right to terminate the account and delete the contents at any time.
2. You will not: (a) process or run any operation on any data other than test data sets or lab-approved materials by AWS, and (b) copy, import, export or otherwise create derivate works of materials provided by AWS, including but not limited to, data sets.
3. AWS is under no obligation to enable the transmission of your materials through [AWS Event Engine] and may, in its discretion, edit, block, refuse to post, or remove your materials at any time.
4. Your use of the [event engine] will comply with these terms and all applicable laws, and your access to [AWS Event Engine] will immediately and automatically terminate if you do not comply with any of these terms or conditions.
This is the 12 digit hash that was given to you or your team.
Proceed


dashboard.eventengine.run
Dashboard
Logout
Team Dashboard
Event
Set Team Name
AWS Console
SSH Key
Event: EE Workshop Demo
Team Name: (Team Name Not Set Yet)
Event ID:
Team ID:


dashboard.eventengine.run
Dashboard
Logout
Team Dashboard
Event
Set Team Name
AWS Console
SSH Key
Event: EE Workshop Demo
Team Name: (Team Name Not Set Yet)
Event ID:
Team ID:

AWS Console Login

Remember to only use "[redacted]" as your region, unless otherwise directed by the event operator.

Login Link

Open AWS Console | Copy Login Link

Credentials / CLI Snippets

Mac / Linux | Windows

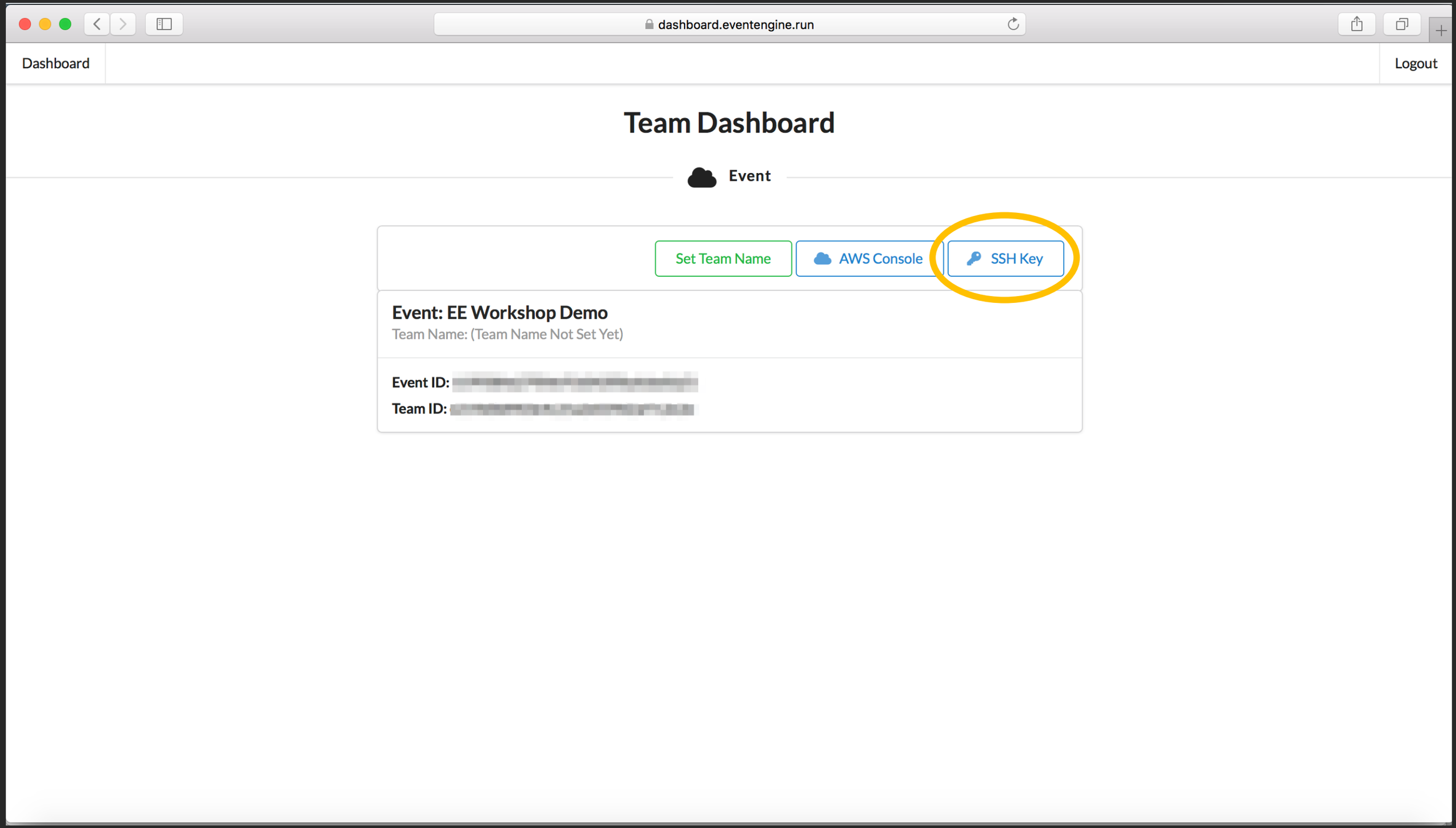
Mac or Linux

```
export AWS_DEFAULT_REGION=[redacted]
export AWS_ACCESS_KEY_ID=[redacted]
export AWS_SECRET_ACCESS_KEY=[redacted]
export AWS_SESSION_TOKEN=[redacted]
```

How do I use the AWS CLI?

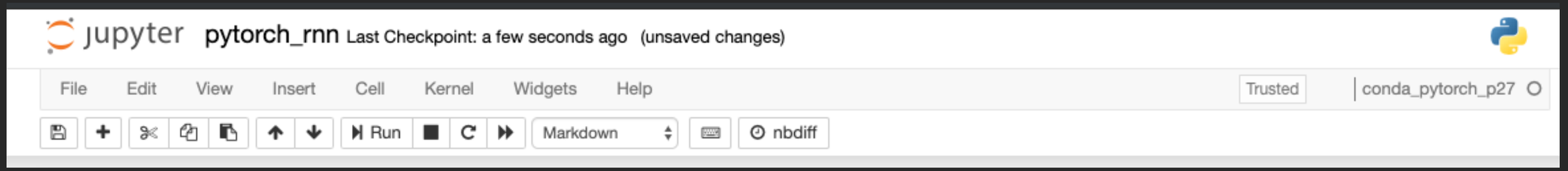
Checkout the AWS CLI documentation here: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

OK


dashboard.eventengine.run
Dashboard
Logout
Team Dashboard
Event
Set Team Name
AWS Console
SSH Key
Event: EE Workshop Demo
Team Name: (Team Name Not Set Yet)
Event ID:
Team ID:

SageMaker PyTorch Support

SageMaker PyTorch Support

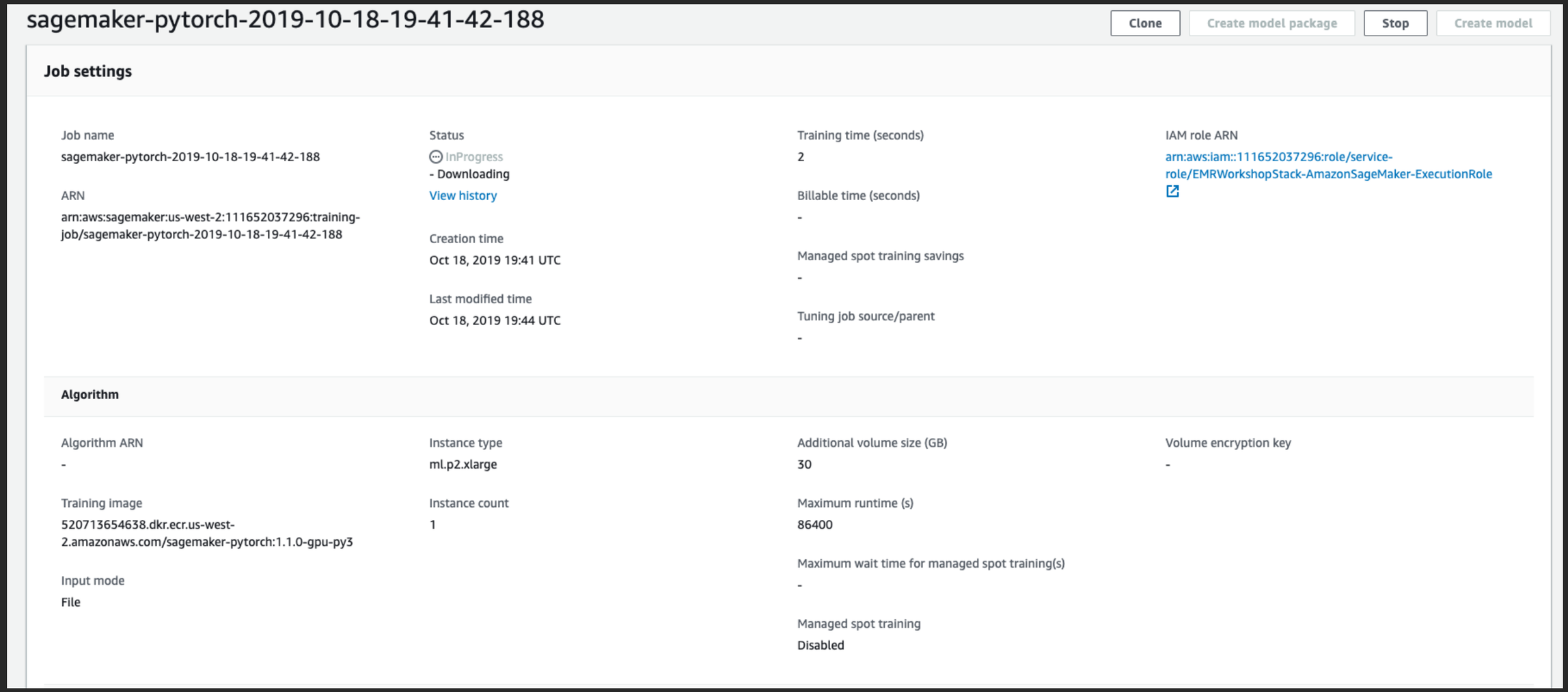


Estimator

```
from sagemaker.pytorch import PyTorch

estimator = PyTorch(entry_point='train.py',
                    role=role,
                    framework_version='1.1.0',
                    train_instance_count=1,
                    train_instance_type='ml.p2.xlarge',
                    # available hyperparameters: emsize, nhid, nlayers, lr, clip, epochs, batch_size,
                    #                            bptt, dropout, tied, seed, log_interval
                    hyperparameters={
                        'epochs': 6,
                        'tied': True
                    })
```

Training Job



Hyperparameters

Metrics

empty

Output data configuration

S3 output path
s3://sagemaker-us-west-2-111652037296/

Output encryption key
-

Hyperparameters

Key	Value
epochs	6
sagemaker_container_log_level	20
sagemaker_enable_cloudwatch_metrics	false
sagemaker_job_name	"sagemaker-pytorch-2019-10-18-19-41-42-188"
sagemaker_program	"train.py"
sagemaker_region	"us-west-2"
sagemaker_submit_directory	"s3://sagemaker-us-west-2-111652037296/sagemaker-pytorch-2019-10-18-19-41-42-188/source/sourcedir.tar.gz"
tied	true

Training Script

jupyter train.py ✔ an hour ago

File Edit View Language Python

```python
# Based on github.com/pytorch/examples/blob/master/word_language_model
import argparse
import math
import os
from shutil import copy
import time
import torch
import torch.nn as nn

import data
from rnn import RNNModel

parser = argparse.ArgumentParser(description='PyTorch Wikitext-2 RNN/LSTM Language Model')

# Hyperparameters sent by the client are passed as command-line arguments to the script.
parser.add_argument('--emsize', type=int, default=200,
                    help='size of word embeddings')
parser.add_argument('--nhid', type=int, default=200,
                    help='number of hidden units per layer')
parser.add_argument('--nlayers', type=int, default=2,
                    help='number of layers')
parser.add_argument('--lr', type=float, default=20,
                    help='initial learning rate')
parser.add_argument('--clip', type=float, default=0.25,
                    help='gradient clipping')
parser.add_argument('--epochs', type=int, default=40,
                    help='upper epoch limit')
parser.add_argument('--batch_size', type=int, default=20, metavar='N',
                    help='batch size')
parser.add_argument('--bptt', type=int, default=35,
                    help='sequence length')
parser.add_argument('--dropout', type=float, default=0.2,
                    help='dropout applied to layers (0 = no dropout)')
parser.add_argument('--tied', type=bool, default=False,
                    help='tie the word embedding and softmax weights')
parser.add_argument('--seed', type=int, default=1111,
                    help='random seed')
parser.add_argument('--log-interval', type=int, default=200, metavar='N',
                    help='report interval')

# Data and model checkpoints/otput directories from the container environment
parser.add_argument('--model-dir', type=str, default=os.environ['SM_MODEL_DIR'])
parser.add_argument('--output-data-dir', type=str, default=os.environ['SM_OUTPUT_DATA_DIR'])
parser.add_argument('--data-dir', type=str, default=os.environ['SM_CHANNEL_TRAINING'])

args = parser.parse_args()
```

Training Script

```
169
170 def train():
171     # Turn on training mode which enables dropout.
172     model.train()
173     total_loss = 0.
174     start_time = time.time()
175     hidden = model.init_hidden(args.batch_size)
176     for batch, i in enumerate(range(0, train_data.size(0) - 1, args.bptt)):
177         data, targets = get_batch(train_data, i)
178         # Starting each batch, we detach the hidden state from how it was previously produced.
179         # If we didn't, the model would try backpropagating all the way to start of the dataset.
180         hidden = repackage_hidden(hidden)
181         model.zero_grad()
182         output, hidden = model(data, hidden)
183         loss = criterion(output.view(-1, ntokens), targets)
184         loss.backward()
185
186         # `clip_grad_norm` helps prevent the exploding gradient problem in RNNs / LSTMs.
187         torch.nn.utils.clip_grad_norm(model.parameters(), args.clip)
188         for p in model.parameters():
189             p.data.add_(-lr, p.grad.data)
190
191         total_loss += loss.item()
192
193         if batch % args.log_interval == 0 and batch > 0:
194             cur_loss = total_loss / args.log_interval
195             elapsed = time.time() - start_time
196             print('| epoch {:3d} | {:5d}/{:5d} batches | lr {:02.2f} | ms/batch {:5.2f} | '
197                   'loss {:5.2f} | ppl {:8.2f}'.format(
198                       epoch, batch, len(train_data) // args.bptt, lr,
199                       elapsed * 1000 / args.log_interval, cur_loss, math.exp(cur_loss)))
200             total_loss = 0
201             start_time = time.time()
202
203
```

Model

```
95
96  ###############################################################################
97  # Build the model
98  ###############################################################################
99
100 print('Build the model')
101 ntokens = len(corpus.dictionary)
102 rnn_type = 'LSTM'
103 model = RNNModel(rnn_type, ntokens, args.emsize, args.nhid, args.nlayers, args.dropout, args.tied).to(device)
104
105 criterion = nn.CrossEntropyLoss()
106
107 # Save the data into model dir to be used with the model later
108 for file_name in os.listdir(args.data_dir):
109     full_file_name = os.path.join(args.data_dir, file_name)
110     if os.path.isfile(full_file_name):
111         copy(full_file_name, args.model_dir)
112
113 # Save arguments used to create model for restoring the model later
114 with open(model_info_path, 'wb') as f:
115     model_info = {
116         'rnn_type': rnn_type,
117         'ntoken': ntokens,
118         'ninp': args.emsize,
119         'nhid': args.nhid,
120         'nlayers': args.nlayers,
121         'dropout': args.dropout,
122         'tie_weights': args.tied
123     }
124     torch.save(model_info, f)
125
126
```

Save the model

```
208 print('Starting training.')
209 for epoch in range(1, args.epochs+1):
210     epoch_start_time = time.time()
211     train()
212     val_loss = evaluate(val_data)
213     print('-' * 89)
214     print('| end of epoch {:3d} | time: {:5.2f}s | valid loss {:5.2f} | '
215           'valid ppl {:8.2f}'.format(epoch, (time.time() - epoch_start_time),
216                                      val_loss, math.exp(val_loss)))
217     print('-' * 89)
218     # Save the model if the validation loss is the best we've seen so far.
219     if not best_state or val_loss < best_state['val_loss']:
220         best_state = {
221             'epoch': epoch,
222             'lr': lr,
223             'val_loss': val_loss,
224             'val_ppl': math.exp(val_loss),
225         }
226         print('Saving the best model: {}'.format(best_state))
227         with open(checkpoint_path, 'wb') as f:
228             torch.save(model.state_dict(), f)
229         with open(checkpoint_state_path, 'w') as f:
230             f.write('epoch {:3d} | lr: {:5.2f} | valid loss {:5.2f} | '
231                     'valid ppl {:8.2f}'.format(epoch, lr, val_loss, math.exp(val_loss)))
232     else:
233         # Anneal the learning rate if no improvement has been seen in the validation dataset.
234         lr /= 4.0
235
```

Model Deployment - PyTorchPredictor

- model_fn() – load model
- input_fn() – convert input payload into a PyTorch Tensor
- predict_fn() – predict_fn() generates predictions from the model based on the return value of input_fn()
- output_fn() – serializes the output from predict_fn() so that it can be returned by the SageMaker Endpoint

Hosting the model

In [ ]:
```
from sagemaker.pytorch import PyTorchModel

training_job_name = estimator.latest_training_job.name
desc = sagemaker_session.sagemaker_client.describe_training_job(TrainingJobName=training_job_name)
trained_model_location = desc['ModelArtifacts']['S3ModelArtifacts']
model = PyTorchModel(model_data=trained_model_location,
                     role=role,
                     framework_version='1.0.0',
                     entry_point='generate.py',
                     source_dir='pytorch-rnn-scripts',
                     git_config=git_config,
                     predictor_cls=JSONPredictor)
```

In [ ]:
```
predictor = model.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

Lab 1 – Word-level Language modeling using PyTorch

Interpreting Results

Train on Premise – Deploy in Cloud using SageMaker

AWS re:Invent

aws

Train on premise – Deploy in the cloud

- Start the workflow from your laptop
- Use your favorite IDE
- Deploy to the cloud using SageMaker

Deploy models in production

Scale and manage the production environment

One-click Deployment

Fully managed with auto-scaling for 75% less

Lab 2 – Toxic Comments Classification using Hugging Face Transformers and SageMaker

AWS re:Invent

aws

Toxic Comment Classification Challenge

- Comments from Wikipedia – human labeled
- 6 Levels – 6 Labels – Multiclass Classification
- Use Fastbert as a wrapper to use HuggingFace's Models
- Debug the lab notebook and run inference on your own text

Source:

- https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
- https://github.com/kaushaltrivedi/fast-bert/tree/master/sample_notebooks

Interpreting Results

Bonus Lab – CAPTUM – Model Interpretability

AWS re:Invent

aws

Thank you!

Kris Skrinak
skrinak@amazon.com

Chaitanya Hazarey
chazarey@amazon.com

AWS re:Invent

aws

Please complete the session survey in the mobile app.