开发人员指南

AWS 深度学习容器

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 深度学习容器: 开发人员指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

什么是 AWS 深度学习 Containers？ 1
主要功能 1
预安装的深度学习框架 1
硬件加速 1
AWS 服务集成 1
安全且定期更新 1
应用场景 2
模型训练 2
模型部署 2
实验和原型制作 2
持续集成和交付 2
Dee AWS p Learning Containers 入门 3
构建自定义镜像 3
..... 3
亚马逊 EC2 教程 4
亚马逊 EC2 设置 4
训练 5
推理 8
自定义入口点 11
亚马逊 ECS 教程 12
亚马逊 ECS 设置 12
训练 15
推理 21
自定义入口点 28
亚马逊 EKS 教程 28
亚马逊 EKS 设置 29
自定义入口点 64
对 EKS 上的 AWS 深度学习容器进行故障排除 65
发布说明 69
基础深度学习 Containers 69
Base DLC CUDA 12.8 X86 已开启 EC2 69
vlLM 深度学习容器 Deep Learning 73
适用于 X86 的 vlLM DLC 开启 EC2 73
PyTorch 深度学习 Containers 76

PyTorch 2.7 正在进行训练 SageMaker 82
PyTorch 2.7 在 ECS 和 EKS 上 EC2进行训练 86
PyTorch 2.7 ARM64 正在进行训练 EC2 90
PyTorch 2.6 在 ECS 和 EKS 上 EC2进行训练 94
PyTorch 2.6 正在进行训练 SageMaker 97
PyTorch 2.6 在 ECS 和 EKS 上进行 EC2推断 101
PyTorch 2.6 推断开启 SageMaker 105
PyTorch 2.6 ARM64 在 EC2、ECS 和 EKS 上进行推断 108
PyTorch 2.6 ARM64 推断开启 SageMaker 112
PyTorch 2.5 在 ECS 和 EKS 上 EC2进行训练 116
PyTorch 2.5 正在进行训练 SageMaker 120
PyTorch 2.5 在 ECS 和 EKS 上进行 EC2推断 124
PyTorch 2.5 推断开启 SageMaker 127
PyTorch 2.5 ARM64 在 EC2、ECS 和 EKS 上进行推断 131
PyTorch 2.5 在 S ARM64 agemaker 上进行推断 134
PyTorch 2.4 在 ECS 和 EKS 上 EC2进行训练 138
PyTorch 2.4 正在进行训练 SageMaker 142
PyTorch 2.4 在 ECS 和 EKS 上进行 EC2推断 146
PyTorch 2.4 推断开启 SageMaker 149
PyTorch 2.4 Graviton on EC2、ECS 和 EKS 153
PyTorch 2.4 Graviton 开启 SageMaker 157
TensorFlow 深度学习 Containers 160
TensorFlow 2.18 在 EC2、ECS 和 EKS 上进行训练 162
TensorFlow 2.18 正在训练 SageMaker 165
TensorFlow 2.18 在 EC2、ECS 和 EKS 上进行推断 169
TensorFlow 2.18 推断开启 SageMaker 172
TensorFlow 2.18 在 ECS 和 EKS 上进行 ARM64 EC2推断 175
TensorFlow 2.18 ARM64 推断开启 SageMaker 178
发布通知 182
支持策略 183
支持的框架 183
常见问题 183
哪些框架版本会获得安全补丁？ 184
AWS 发布新框架版本时会发布哪些镜像？ 184
哪些图像获得了新的 SageMaker AI/AWS 功能？ 184
“支持的框架”表中是如何定义当前版本的？ 184

如果我运行的版本不在“支持的框架”表中，该怎么办？ 184
是否 DLCs 支持以前版本的 TensorFlow？ 184
如何找到支持的框架版本的最新补丁映像？ 185
多长时间发布一次新映像？ 185
运行工作负载时，能在我的实例上以替代方式安装补丁吗？ 185
如果有新的补丁或更新的框架版本可用，会发生什么呢？ 185
是否可在不更改框架版本的情况下更新依赖项？ 185
对我的框架版本的主动支持何时结束？ 185
对于框架版本不再主动维护的映像，会为其安装补丁吗？ 187
如何使用旧框架版本？ 187
如何保持框架及其版本 up-to-date的支持变更？ 187
是否需要商业许可证才能使用 Anaconda 存储库？ 187
框架 Support 政策表 188
支持的框架版本 188
不支持的框架版本 188
更新 190
不支持的框架 190
安全性 191
数据保护 191
身份和访问管理 192
使用身份进行身份验证 193
使用策略管理访问 195
IAM 与 Amazon EMR 结合使用 197
监控和使用情况跟踪 197
使用情况跟踪 197
故障率跟踪 198
以下框架版本中的使用情况跟踪 198
合规性验证 198
恢复能力 199
基础设施安全性 199
文档历史记录 200
AWS 词汇表 201
..... ccii

什么是 AWS 深度学习 Containers？

AWS Deep Learning Containers 是预先构建的 Docker 镜像，可以更轻松地运行流行的深度学习框架和工具。 AWS它们为 AWS 基础架构上托管的深度学习应用程序提供一致 up-to-date、安全和优化的运行时环境。要开始使用，请参阅 Dee AWS p Learning Containers 入门。

主要功能

预安装的深度学习框架

AWS Deep Learning Containers 包括领先的深度学习框架的预安装和配置版本，例如 TensorFlow 和。 PyTorch这样就无需从头开始构建和维护自己的 Docker 镜像。

硬件加速

AWS Deep Learning Containers 针对基于 CPU、GPU 加速和基于硅的深度学习进行了优化。 AWS 它们支持 CUDA、cuDNN 和其他必要的库，以利用基于 GPU 的 EC2 亚马逊实例以及由 G CPUs raviton 和 Tra GPUs inium 以及英特尔的 Habana-Gaudi 处理器等设计的芯片 AWS以及英特尔的 Habana-Gaudi 处理器的强大功能。 AWS

AWS 服务集成

AWS Deep Learning Containers 可与各种 AWS 服务无缝集成，包括 SageMaker 人工智能、亚马逊弹性容器服务 (ECS)、亚马逊 Elastic Kubernetes Service (EKS)、亚马逊和 EC2 AWS ParallelCluster 这样可以轻松地在 AWS 基础架构上部署和运行深度学习模型和应用程序。

安全且定期更新

AWS 定期维护和更新 Dee AWS p Learning Containers，以确保您可以访问最新版本的深度学习框架和依赖关系。这有助于保持 AWS基于您的深度学习环境的安全 up-to-date，并且无需自己管理安全补丁和更新的开销。使用最新的安全补丁更新深度学习容器可能是一项资源密集型任务，但是 Dee AWS p Learning Containers 通过提供定期的自动更新来消除这种负担。这样可以确保您的深度学习环境保持安全和最新，而无需您进行大量手动操作。通过自动化更新过程，Dee AWS p Learning Containers 使您可以专注于开发深度学习模型和应用程序，而不必担心底层基础设施和安全维护，这可以提高团队的工作效率，并使您能够在托管的项目中更有效地利用最新的深度学习功能。 AWS

应用场景

AWS Deep Learning Containers 在以下 AWS基于深度学习的场景中特别有用：

模型训练

使用 Dee AWS p Learning Containers 在基于 CPU、GPU 加速或硅驱动的 AWS Amazon EC2 实例上训练您的深度学习模型，或者在 Hyperpod 上利用多节点训练。 AWS ParallelCluster SageMaker

模型部署

使用 Dee AWS p Learning Containers 部署经过训练的模型，以便通过人工智能等方式进行可扩展 AWS、可用于生产的推理。 SageMaker

实验和原型制作

AWS 使用预先配置的容器快速启动深度学习开发环境。 AWS Deep Learning Containers 是 SageMaker AI Studio 中笔记本的默认选项，可以轻松开始实验和原型设计。

持续集成和交付

将容器集成到 AWS基于您的 CI/CD 管道中，例如使用 Amazon ECS 或 Amazon EKS 的容器，以实现一致、自动化的深度学习工作负载。

Dee AWS p Learning Containers 入门

以下各节介绍如何使用 Deep Learning Containers 在 AWS 基础架构上运行每个框架中的示例代码。

使用案例

- 有关在 SageMaker AI 中使用 Deep Learning Containers 的信息，请参阅《将自己的算法或模型与 SageMaker AI 结合使用文档》。
- 要了解如何在 EKS HyperPod 上使用带有 SageMaker AI 的 Deep Learning Containers，请参阅使用 Amazon E SageMaker KS AI 编排 SageMaker HyperPod 集群。

主题

- 自定义深度学习 Containers
- 亚马逊 EC2 教程
- 亚马逊 ECS 教程
- 亚马逊 EKS 教程

自定义深度学习 Containers

我们可以使用 Deep Learning Containers 自定义训练和推理，使用 Docker 文件添加自定义框架、库和包。

示例

在下面的示例 Dockerfile 中，我们将包含许多深度学习模型示例的示例 GitHub 存储库添加到 PyTorch Inference 深度学习容器中。 AWS

```
# Take base container
FROM 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference:2.4-cpu-py311-ec2

# Add custom stack of code
RUN git clone https://github.com/aws-samples/deep-learning-models
```

使用 Docker 映像的自定义名称和自定义标签构建映像，同时指向您的个人 Docker 注册表（通常为您的用户名）。

```
docker build -f Dockerfile -t <registry>/<any name>:<any tag>
```

推送至您的个人 Docker 注册表：

```
docker push <registry>/<any name>:<any tag>
```

您可以使用以下命令运行容器：

```
docker run -it < name or tag>
```

> **Important**
>
> 你可能需要登录才能访问 Deep Learning Containers 图像存储库。在以下命令中指定您的区域：
>
> ```
> aws ecr get-login-password --region us-east-1 | docker login --username AWS --
> password-stdin 763104351884.dkr.ecr.us-east-1.amazonaws.com
> ```

亚马逊 EC2 教程

本节介绍如何在 Deep Learning Containers 上运行训练和推理以 EC2 使用 PyTorch、和。TensorFlow

在开始以下教程之前，请完成中的步骤亚马逊 EC2 设置。

内容

- 亚马逊 EC2 设置
- 训练
- 推理
- 自定义入口点

亚马逊 EC2 设置

在本节中，您将学习如何使用亚马逊弹性计算云设置 Dee AWS p Learning Containers。

完成以下步骤来配置您的实例：

- 使用以下策略创建 AWS Identity and Access Management 用户或修改现有用户。您可以在 IAM 控制台的策略选项卡中按名称搜索它们。
  - 亚马逊 ECS_ 政策 FullAccess
  - AmazonEC2ContainerRegistryFullAccess

  有关创建或编辑 IAM 用户的更多信息，请参阅 IA M 用户指南中的添加和删除 IAM 身份权限。
- 启动亚马逊弹性计算云实例（CPU 或 GPU），最好是深度学习基础 AMI。其他 AMIs 工作，但需要相关的 GPU 驱动程序。
- 通过使用 SSH 连接到您的 实例。有关连接的更多信息，请参阅 Amazon EC2 用户指南中的实例连接疑难解答。 。
- 使用安装 AWS CLI 当前 AWS CLI 版本中的步骤确保您的版本是最新的。
- 在您的实例中，运行 `aws configure` 并提供已创建用户的凭证。
- 在您的实例中，运行以下命令登录托管 Deep Learning Containers 图像的 Amazon ECR 存储库。

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin 763104351884.dkr.ecr.us-east-1.amazonaws.com
```

后续步骤

要了解有关使用 Deep Learning Containers 在亚马逊上 EC2 进行训练和推理的信息，请参阅亚马逊 EC2 教程。

训练

本节介绍如何 EC2 使用 PyTorch 和在 Dee AWS p Learning Containers for Amazon 上运行训练 TensorFlow。

内容

- PyTorch训练
- TensorFlow训练
- 后续步骤

PyTorch训练

要 PyTorch 从您的 Amazon EC2 实例开始训练，请使用以下命令运行容器。必须用**`nvidia-docker`**于 GPU 映像。

- 适用于 CPU

```
$ docker run -it <CPU training container>
```

- 对于 GPU

```
$ nvidia-docker run -it <GPU training container>
```

- 如果你有 docker-ce 版本 19.03 或更高版本，你可以在 docker 中使用--gpus 标志：

```
$ docker run -it --gpus <GPU training container>
```

运行以下命令开始训练。

- 适用于 CPU

```
$ git clone https://github.com/pytorch/examples.git
$ python examples/mnist/main.py --no-cuda
```

- 对于 GPU

```
$ git clone https://github.com/pytorch/examples.git
$ python examples/mnist/main.py
```

PyTorch 使用 NVIDIA Apex 进行分布式 GP

NVIDIA Apex 是一款具有用于混合精度和分布式训练的实用程序的 PyTorch 扩展。有关Apex提供的实用程序的更多信息，请访问 NVIDIA Apex网站。Apex 目前由以下系列的亚马逊 EC2 实例支持：

- 亚马逊 EC2 P3 实例
- 亚马逊 EC2 P4 实例
- 亚马逊 EC2 P5 实例
- 亚马逊 EC2 G5 实例

要开始使用 NVIDIA Apex 进行分布式训练，请在 GPU 训练容器的终端中运行以下命令。此示例要求您的 Amazon EC2 实例 GPUs 上至少有两个，才能运行并行分布式训练。

```
$ git clone https://github.com/NVIDIA/apex.git && cd apex
$ python -m torch.distributed.launch --nproc_per_node=2 examples/simple/distributed/
distributed_data_parallel.py
```

TensorFlow训练

登录 Amazon EC2 实例后，您可以使用以下命令运行 TensorFlow TensorFlow 2 个容器。必须用`nvidia-docker`于 GPU 映像。

- 对于基于 CPU 的训练，请运行以下命令。

```
$ docker run -it <CPU training container>
```

- 对于基于 GPU 的训练，请运行以下命令。

```
$ nvidia-docker run -it <GPU training container>
```

上一命令以交互模式运行容器并在容器内提供一个 shell 提示符。然后，您可以运行以下命令进行导入 TensorFlow。

```
$ python
```

```
>> import tensorflow
```

按 Ctrl+D 返回到 bash 提示符。运行以下命令以开始训练：

```
git clone https://github.com/fchollet/keras.git
```

```
$ cd keras
```

```
$ python examples/mnist_cnn.py
```

后续步骤

要在亚马逊上 EC2 使用 Deep Learning Cont PyTorch ainers 学习推理，请参阅PyTorch推断 。

推理

本节介绍如何使用 PyTorch和在适用于亚马逊弹性计算云的 Dee AWS p Learning Containers 上运行推理。 TensorFlow

内容

- PyTorch推断
- TensorFlow推断

PyTorch推断

1.6 及更高 PyTorch 版本的 Deep Learning Container TorchServe s 用于推理调用。1.5 及更早 PyTorch 版本的 Deep Learning Container `multi-model-server` s 用于推理调用。

PyTorch 1.6 及更高版本

为了运行推理 PyTorch，此示例使用了公共 S3 存储桶在 Imagenet 上预训练的模型。推理是使用 TorchServe提供的。有关更多信息，请参阅这篇关于使用部署 PyTorch 推理的 TorchServe博客。

对于 CPU 实例：

```
$ docker run -itd --name torchserve -p 80:8080  -p 8081:8081 <your container image id>
 \
torchserve --start --ts-config /home/model-server/config.properties \
--models pytorch-densenet=https://torchserve.s3.amazonaws.com/mar_files/densenet161.mar
```

对于 GPU 实例

```
$ nvidia-docker run -itd --name torchserve -p 80:8080  -p 8081:8081 <your container
 image id> \
torchserve --start --ts-config /home/model-server/config.properties \
--models pytorch-densenet=https://torchserve.s3.amazonaws.com/mar_files/densenet161.mar
```

如果你有 docker-ce 版本 19.03 或更高版本，则可以在启动 Docker 时使用该`--gpus`标志。

配置文件包含在容器中。

服务器启动后，您现在可以使用以下方法从不同的窗口运行推理。

```
$ curl -O https://s3.amazonaws.com/model-server/inputs/flower.jpg
```

```
curl -X POST http://127.0.0.1:80/predictions/pytorch-densenet -T flower.jpg
```

使用完容器后，您可以使用以下方法将其移除。

```
$ docker rm -f torchserve
```

PyTorch 1.5 及更早版本

为了运行推理 PyTorch，此示例使用了公共 S3 存储桶在 Imagenet 上预训练的模型。推理是使用的 multi-model-server，它可以支持任何框架作为后端。有关更多信息，请参阅 multi-model-server。

对于 CPU 实例：

```
$ docker run -itd --name mms -p 80:8080  -p 8081:8081 <your container image id> \
multi-model-server --start --mms-config /home/model-server/config.properties \
--models densenet=https://dlc-samples.s3.amazonaws.com/pytorch/multi-model-server/
densenet/densenet.mar
```

对于 GPU 实例

```
$ nvidia-docker run -itd --name mms -p 80:8080  -p 8081:8081 <your container image id>
 \
multi-model-server --start --mms-config /home/model-server/config.properties \
--models densenet=https://dlc-samples.s3.amazonaws.com/pytorch/multi-model-server/
densenet/densenet.mar
```

如果你有 docker-ce 版本 19.03 或更高版本，则可以在启动 Docker 时使用该`--gpus`标志。

配置文件包含在容器中。

服务器启动后，您现在可以使用以下方法从不同的窗口运行推理。

```
$ curl -O https://s3.amazonaws.com/model-server/inputs/flower.jpg
curl -X POST http://127.0.0.1/predictions/densenet -T flower.jpg
```

使用完容器后，您可以使用以下方法将其移除。

```
$ docker rm -f mms
```

TensorFlow推断

为了演示如何使用 Deep Learning Containers 进行推理，此示例使用了一个带有 TensorFlow 2 Serving 的简单半加二模型。我们建议使用深度学习基础 AMI for TensorFlow 2。登录实例后，运行以下命令。

```
$ git clone -b r2.0 https://github.com/tensorflow/serving.git
$ cd serving
```

使用此处的命令开始使用此模型的 Deep Learning Containers 进行 TensorFlow 服务。与用于训练的 Deep Learning Containers 不同，模型服务在运行容器后立即开始，并作为后台进程运行。

- 对于 CPU 实例：

  ```
  $ docker run -p 8500:8500 -p 8501:8501 --name tensorflow-inference --mount
   type=bind,source=$(pwd)/tensorflow_serving/servables/tensorflow/testdata/
  saved_model_half_plus_two_cpu,target=/models/saved_model_half_plus_two -e
   MODEL_NAME=saved_model_half_plus_two -d <cpu inference container>
  ```

  例如：

  ```
  $ docker run -p 8500:8500 -p 8501:8501 --name tensorflow-inference --mount
   type=bind,source=$(pwd)/tensorflow_serving/servables/tensorflow/testdata/
  saved_model_half_plus_two_cpu,target=/models/saved_model_half_plus_two
   -e MODEL_NAME=saved_model_half_plus_two -d 763104351884.dkr.ecr.us-
  east-1.amazonaws.com/tensorflow-inference:2.0.0-cpu-py36-ubuntu18.04
  ```

- 对于 GPU 实例：

  ```
  $ nvidia-docker run -p 8500:8500 -p 8501:8501 --name tensorflow-inference --
  mount type=bind,source=$(pwd)/tensorflow_serving/servables/tensorflow/testdata/
  saved_model_half_plus_two_gpu,target=/models/saved_model_half_plus_two -e
   MODEL_NAME=saved_model_half_plus_two -d <gpu inference container>
  ```

  例如：

  ```
  $ nvidia-docker run -p 8500:8500 -p 8501:8501 --name tensorflow-inference
   --mount type=bind,source=$(pwd)/tensorflow_serving/servables/tensorflow/
  testdata/saved_model_half_plus_two_gpu,target=/models/saved_model_half_plus_two
  ```

```
 -e MODEL_NAME=saved_model_half_plus_two -d 763104351884.dkr.ecr.us-
east-1.amazonaws.com/tensorflow-inference:2.0.0-gpu-py36-cu100-ubuntu18.04
```

> **Note**
>
> 加载 GPU 模型服务器可能需要一些时间。

接下来，使用 Deep Learning Containers 运行推理。

```
$ curl -d '{"instances": [1.0, 2.0, 5.0]}' -X POST http://127.0.0.1:8501/v1/models/
saved_model_half_plus_two:predict
```

输出类似于以下内容。

```
{
    "predictions": [2.5, 3.0, 4.5
    ]
}
```

> **Note**
>
> 要调试容器的输出，您可以使用名称附加到容器的输出，如以下命令所示：
>
> ```
> $ docker attach <your docker container name>
> ```
>
> 使用了这个例子`tensorflow-inference`。

后续步骤

要了解如何在 Amazon ECS 上将自定义入口点与 Deep Learning Containers 配合使用，请参阅。自定义入口点

自定义入口点

对于某些图像，Deep Learning Containers 使用自定义入口点脚本。如果您要使用自己的入口点，可以按如下方式覆盖入口点。

- 要指定要运行的自定义入口点脚本，请使用此命令。

```
docker run --entrypoint=/path/to/custom_entrypoint_script -it <image> /bin/bash
```

- 要将入口点设置为空，请使用此命令。

```
docker run --entrypoint="" <image> /bin/bash
```

亚马逊 ECS 教程

本节介绍如何使用 PyTorch和在适用于 Amazon ECS 的 Dee AWS p Learning Containers 上运行训练和 TensorFlow推理。

在开始以下教程之前，请完成中的步骤亚马逊 ECS 设置。

内容

- 亚马逊 ECS 设置
- 训练
- 推理
- 自定义入口点

亚马逊 ECS 设置

本主题介绍如何使用亚马逊弹性容器服务设置 D AWS eep Learning Containers。

内容

- 前提条件
- 为 Deep Learning Containers 设置亚马逊 ECS

前提条件

本安装指南假设您已完成以下先决条件：

- 安装并配置最新版本的 AWS CLI。有关安装或升级的更多信息 AWS CLI，请参阅安装 AWS Command Line Interface。
- 完成使用 Amazon ECS 进行设置中的步骤。

- 确认您拥有 Amazon ECS 容器实例角色。有关更多信息，请参阅《亚马逊弹性容器服务开发人员指南》中的 Amazon ECS 容器实例 IAM 角色。
- Amazon CloudWatch Logs IAM 策略已添加到 Amazon ECS 容器实例角色中，该角色允许 Amazon ECS 向亚马逊发送日志 CloudWatch。有关更多信息，请参阅 A mazon 弹性容器服务开发者指南中的CloudWatch 日志 IAM 策略。
- 创建新的安全组或更新现有的安全组，以打开所需的推理服务器的端口。
  - 为了进行 TensorFlow 推断，端口 8501 和 8500 向 TCP 流量开放。

  有关更多信息，请参阅 Amazon EC2 安全组。

为 Deep Learning Containers 设置亚马逊 ECS

本节介绍如何将 Amazon ECS 设置为使用 Deep Learning Containers。

> **Important**
>
> 如果您的账户已经创建了 Amazon ECS 服务相关角色，则除非您在此处指定角色，否则默认情况下，该角色将用于您的服务。如果您的任务定义使用 awsvpc 网络模式，或者服务配置为使用以下任一功能：服务发现、外部部署控制器、多个目标组或 Elastic Inference 加速器，则需要服务相关角色。如果是这种情况，则不应在此处指定角色。有关更多信息，请参阅 Amazon ECS 开发人员指南中的使用适用于 Amazon ECS 的服务相关角色。

从您的主机运行以下操作。

1. 在包含您之前创建的密钥对和安全组的区域中创建 Amazon ECS 集群。

```
aws ecs create-cluster --cluster-name ecs-ec2-training-inference --region us-east-1
```

2. 在您的集群中启动一个或多个 Amazon EC2 实例。有关基于 GPU 的工作，请参阅 GPUs Amazon ECS 开发人员指南中的在 Amazon ECS 上使用，以告知您的实例类型选择。如果您选择 GPU 实例类型，请务必选择经过 GPU 优化的 Amazon ECS AMI。对于基于 CPU 的工作，你可以使用经过 ECS 优化的亚马逊 Linux 或 Amazon Linux 2。 AMIs有关兼容的实例类型和亚马逊 ECS 优化的 AMI 的更多信息 IDs，请参阅亚马逊 ECS 优化。 AMIs在本示例中，您将启动一个带有基于 GPU 的 AMI 的实例，其磁盘大小为 100 GB，采用 us-east-1。

   a. 使用以下内容创建名为 `my_script.txt` 的文件。引用您在上一步中创建的同一集群名称。

```
#!/bin/bash
echo ECS_CLUSTER=ecs-ec2-training-inference >> /etc/ecs/ecs.config
```

b. （可选）使用以下内容创建名为 `my_mapping.txt` 的文件，这将在创建实例后更改根卷的大小。

```
[
    {
        "DeviceName": "/dev/xvda",
        "Ebs": {
            "VolumeSize": 100
        }
    }
]
```

c. 使用亚马逊 ECS 优化的 AMI 启动亚马逊 EC2 实例，并将其连接到集群。使用您创建的安全组 ID 和 key pair 名称，并在以下命令中替换它们。要获取最新的亚马逊 ECS 优化版 AMI ID，请参阅亚马逊弹性容器服务开发 AMIs者指南中的亚马逊 ECS 优化。

```
aws ec2 run-instances --image-id ami-0dfdeb4b6d47a87a2 \
                      --count 1 \
                      --instance-type p2.8xlarge \
                      --key-name key-pair-1234 \
                      --security-group-ids sg-abcd1234 \
                      --iam-instance-profile Name="ecsInstanceRole" \
                      --user-data file://my_script.txt \
                      --block-device-mapping file://my_mapping.txt \
                      --region us-east-1
```

在 Amazon EC2 控制台中，您可以通过响应中的`instance-id`来验证此步骤是否成功。

现在，您已经拥有一个正在运行容器实例的 Amazon ECS 集群。通过以下步骤验证 Amazon EC2 实例是否已在集群中注册。

验证 Amazon EC2 实例是否已在集群中注册

1. 在 https://console.aws.amazon.com/ecs/v2 中打开控制台。
2. 选择包含已注册的 Amazon EC2 实例的集群。

3. 在集群页面上，选择基础设施。
4. 在 “容器实例” 下，验证是否显示`instance-id`了在上一步中创建的。另外，请注意可用 CPU 和可用内存的值，因为这些值在以下教程中可能很有用。上述值可能需要几分钟才能显示在控制台中。

后续步骤

要了解有关在 Amazon ECS 上使用 Deep Learning Containers 进行训练和推理的信息，请参阅亚马逊 ECS 教程。

训练

> **Note**
>
> 本节介绍如何使用和在 Dee AWS p Learning Containers for Amazon 弹性容器 PyTorch 服务上运行训练 TensorFlow。

> **Important**
>
> 如果您的账户已创建 Amazon ECS 服务相关角色，则默认情况下会为您的服务使用该角色，除非您在此处指定一个角色。如果您的任务定义使用 awsvpc 网络模式或将服务配置为使用服务发现，则需要服务相关角色。如果服务使用外部部署控制器、多个目标组或 Elastic Inference 加速器，则也需要该角色，在这种情况下，您不应在此处指定角色。有关更多信息，请参阅 Amazon ECS 开发人员指南中的使用适用于 Amazon ECS 的服务相关角色。

内容

- PyTorch 训练
- 后续步骤
- TensorFlow训练
- 后续步骤

PyTorch 训练

必须先注册任务定义，然后才能在 Amazon ECS 集群上运行任务。任务定义是分组在一起的一系列容器。以下示例使用一个示例 Docker 镜像，该镜像将训练脚本添加到 Deep Learning Containers 中。

1. 使用以下内容创建名为 `ecs-deep-learning-container-training-taskdef.json` 的文件。

   - 适用于 CPU

```
{
   "requiresCompatibilities":[
      "EC2"
   ],
   "containerDefinitions":[
      {
         "command":[
            "git clone https://github.com/pytorch/examples.git && python
 examples/mnist/main.py --no-cuda"
         ],
         "entryPoint":[
            "sh",
            "-c"
         ],
         "name":"pytorch-training-container",
         "image":"763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-
training:1.5.1-cpu-py36-ubuntu16.04",
         "memory":4000,
         "cpu":256,
         "essential":true,
         "portMappings":[
            {
               "containerPort":80,
               "protocol":"tcp"
            }
         ],
         "logConfiguration":{
            "logDriver":"awslogs",
            "options":{
               "awslogs-group":"/ecs/pytorch-training-cpu",
               "awslogs-region":"us-east-1",
               "awslogs-stream-prefix":"mnist",
               "awslogs-create-group":"true"
```

```
            }
         }
      }
   ],
   "volumes":[

   ],
   "networkMode":"bridge",
   "placementConstraints":[

   ],
   "family":"pytorch"
}
```

- 对于 GPU

```
{
      "requiresCompatibilities": [
        "EC2"
      ],
      "containerDefinitions": [
        {
    "command": [
                "git clone https://github.com/pytorch/examples.git && python
 examples/mnist/main.py"
            ],
            "entryPoint": [
               "sh",
               "-c"
            ],
          "name": "pytorch-training-container",
          "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-
training:1.5.1-gpu-py36-cu101-ubuntu16.04",
          "memory": 6111,
          "cpu": 256,
          "resourceRequirements" : [{
            "type" : "GPU",
            "value" : "1"
          }],
          "essential": true,
          "portMappings": [
            {
              "containerPort": 80,
              "protocol": "tcp"
```

```
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/pytorch-training-gpu",
                "awslogs-region": "us-east-1",
                "awslogs-stream-prefix": "mnist",
                    "awslogs-create-group": "true"
            }
        }
      }
    ],
    "volumes": [],
    "networkMode": "bridge",
    "placementConstraints": [],
    "family": "pytorch-training"
  }
```

2. 注册任务定义。记下输出中的修订版号，并在下一步中使用它。

```
aws ecs register-task-definition --cli-input-json file://ecs-deep-learning-
container-training-taskdef.json
```

3. 使用任务定义创建任务。您需要上一步中的修订标识符。

```
aws ecs run-task --cluster ecs-ec2-training-inference --task-definition pytorch:1
```

4. 在 https://console.aws.amazon.com/ecs/v2 中打开控制台。
5. 选择 `ecs-ec2-training-inference` 集群。
6. 在 Cluster 页面上，选择 Tasks。
7. 任务处于`RUNNING`状态后，选择任务标识符。
8. 在 “日志” 下，选择 “查看登录信息” CloudWatch。这会将您带到 CloudWatch 控制台以查看训练进度日志。

后续步骤

要使用 PyTorch Deep Learning Containers 在 Amazon ECS 上学习推理，请参阅PyTorch 推断。

TensorFlow训练

您必须先注册任务定义才能在 ECS 集群上运行任务。任务定义是分组在一起的一系列容器。以下示例使用一个示例 Docker 镜像，该镜像将训练脚本添加到 Deep Learning Containers 中。您可以将此脚本与 TensorFlow 或 TensorFlow 2 配合使用。要将其与 TensorFlow 2 一起使用，请将 Docker 镜像更改为 TensorFlow 2 镜像。

1. 使用以下内容创建名为 `ecs-deep-learning-container-training-taskdef.json` 的文件。

   - 适用于 CPU

```
{
        "requiresCompatibilities": [
            "EC2"
        ],
        "containerDefinitions": [{
            "command": [
                "mkdir -p /test && cd /test && git clone https://github.com/
fchollet/keras.git && chmod +x -R /test/ && python keras/examples/mnist_cnn.py"
            ],
            "entryPoint": [
                "sh",
                "-c"
            ],
            "name": "tensorflow-training-container",
            "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference:1.15.2-cpu-py36-ubuntu18.04",
            "memory": 4000,
            "cpu": 256,
            "essential": true,
            "portMappings": [{
                "containerPort": 80,
                "protocol": "tcp"
            }],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "awslogs-tf-ecs",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "tf",
                    "awslogs-create-group": "true"
                }
```

```
            }
        }],
        "volumes": [],
        "networkMode": "bridge",
        "placementConstraints": [],
        "family": "TensorFlow"
    }
```

- 对于 GPU

```
{
          "requiresCompatibilities": [
            "EC2"
          ],
          "containerDefinitions": [
            {
        "command": [
                    "mkdir -p /test && cd /test && git clone https://github.com/
fchollet/keras.git && chmod +x -R /test/ && python keras/examples/mnist_cnn.py"
                 ],
                 "entryPoint": [
                    "sh",
                    "-c"
                 ],
              "name": "tensorflow-training-container",
              "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
training:1.15.2-gpu-py37-cu100-ubuntu18.04",
              "memory": 6111,
              "cpu": 256,
              "resourceRequirements" : [{
                "type" : "GPU",
                "value" : "1"
              }],
              "essential": true,
              "portMappings": [
                {
                  "containerPort": 80,
                  "protocol": "tcp"
                }
              ],
              "logConfiguration": {
                  "logDriver": "awslogs",
                  "options": {
                      "awslogs-group": "awslogs-tf-ecs",
```

```
                        "awslogs-region": "us-east-1",
                        "awslogs-stream-prefix": "tf",
                            "awslogs-create-group": "true"
                    }
                }
              }
            ],
            "volumes": [],
            "networkMode": "bridge",
            "placementConstraints": [],
            "family": "tensorflow-training"
          }
```

2. 注册任务定义。记下输出中的修订版号，并在下一步中使用它。

```
aws ecs register-task-definition --cli-input-json file://ecs-deep-learning-
container-training-taskdef.json
```

3. 使用任务定义创建任务。您需要上一步中的修订版号和在安装过程中创建的集群的名称

```
aws ecs run-task --cluster ecs-ec2-training-inference --task-definition tf:1
```

4. 打开 Amazon ECS 经典控制台，网址为https://console.aws.amazon.com/ecs/。
5. 选择 `ecs-ec2-training-inference` 集群。
6. 在 Cluster 页面上，选择 Tasks。
7. 任务处于`RUNNING`状态后，选择任务标识符。
8. 在 “日志” 下，选择 “查看登录信息” CloudWatch。这会将您带到 CloudWatch 控制台以查看训练进度日志。

后续步骤

要使用 TensorFlow Deep Learning Containers 在 Amazon ECS 上学习推理，请参阅TensorFlow推断。

推理

本节介绍如何使用、和在适用于亚马逊弹性容器服务 (Amazon ECS) 的 Dee AWS p Learning Containers PyTorch 上运行推理。 TensorFlow

> **Important**
>
> 如果您的账户已经创建了 Amazon ECS 服务相关角色，则除非您在此处指定角色，否则默认情况下，该角色将用于您的服务。如果您的任务定义使用 awsvpc 网络模式，则需要服务相关角色。如果服务配置为使用服务发现、外部部署控制器、多个目标组或 Elastic Inference 加速器，则也需要该角色，在这种情况下，您不应在此处指定角色。有关更多信息，请参阅 Amazon ECS 开发人员指南中的使用适用于 Amazon ECS 的服务相关角色。

内容

- PyTorch 推断
- TensorFlow推断

PyTorch 推断

必须先注册任务定义，然后才能在 Amazon ECS 集群上运行任务。任务定义是分组在一起的一系列容器。以下示例使用一个示例 Docker 镜像，该镜像将 CPU 或 GPU 推理脚本添加到 Deep Learning Containers 中。

后续步骤

要了解如何在 Amazon ECS 上使用带有 Deep Learning Containers 的自定义入口点，请参阅。自定义入口点

TensorFlow推断

以下示例使用一个示例 Docker 镜像，该镜像通过主机的命令行将 CPU 或 GPU 推理脚本添加到 Deep Learning Containers 中。

基于 CPU 的推理

使用以下示例运行基于 CPU 的推理。

1. 使用以下内容创建名为 `ecs-dlc-cpu-inference-taskdef.json` 的文件。你可以将其与 TensorFlow 或 TensorFlow 2 一起使用。要将其与 TensorFlow 2 一起使用，请将 Docker 镜像更改为 TensorFlow 2 镜像，然后克隆 r2.0 服务存储库分支而不是 r1.15。

```
{
 "requiresCompatibilities": [
  "EC2"
```

```
 ],
 "containerDefinitions": [{
  "command": [
   "mkdir -p /test && cd /test && git clone -b r1.15 https://github.com/
tensorflow/serving.git && tensorflow_model_server --port=8500 --rest_api_port=8501
 --model_name=saved_model_half_plus_two --model_base_path=/test/serving/
tensorflow_serving/servables/tensorflow/testdata/saved_model_half_plus_two_cpu"
  ],
  "entryPoint": [
   "sh",
   "-c"
  ],
  "name": "tensorflow-inference-container",
  "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference:1.15.0-cpu-py36-ubuntu18.04",
  "memory": 8111,
  "cpu": 256,
  "essential": true,
  "portMappings": [{
    "hostPort": 8500,
    "protocol": "tcp",
    "containerPort": 8500
   },
   {
    "hostPort": 8501,
    "protocol": "tcp",
    "containerPort": 8501
   },
   {
    "containerPort": 80,
    "protocol": "tcp"
   }
  ],
  "logConfiguration": {
   "logDriver": "awslogs",
   "options": {
    "awslogs-group": "/ecs/tensorflow-inference-gpu",
    "awslogs-region": "us-east-1",
    "awslogs-stream-prefix": "half-plus-two",
    "awslogs-create-group": "true"
   }
  }
 }],
 "volumes": [],
```

```
 "networkMode": "bridge",
 "placementConstraints": [],
 "family": "tensorflow-inference"
}
```

2. 注册任务定义。记下输出中的修订版号，并在下一步中使用它。

```
aws ecs register-task-definition --cli-input-json file://ecs-dlc-cpu-inference-taskdef.json
```

3. 创建 Amazon ECS 服务 指定任务定义时，请`revision_id`用上一步输出中任务定义的修订版号替换。

```
aws ecs create-service --cluster ecs-ec2-training-inference \
                       --service-name cli-ec2-inference-cpu \
                       --task-definition Ec2TFInference:revision_id \
                       --desired-count 1 \
                       --launch-type EC2 \
                       --scheduling-strategy="REPLICA" \
                       --region us-east-1
```

4. 通过完成以下步骤来验证服务并获取网络终端节点。

   a. 在 https://console.aws.amazon.com/ecs/v2 中打开控制台。

   b. 选择 `ecs-ec2-training-inference` 集群。

   c. 在 Cluster (集群) 页面上，选择 Services (服务)，然后选择 cli-ec2-inference-cpu。

   d. 任务处于`RUNNING`状态后，选择任务标识符。

   e. 在 “日志” 下，选择 “查看登录信息” CloudWatch。这会将您带到 CloudWatch 控制台以查看训练进度日志。

   f. 在 Containers (容器) 下，展开容器详细信息。

   g. 在 “名称” 和 “网络绑定” 下，在 “外部链接” 下记下端口 8501 的 IP 地址，并在下一步中使用该地址。

5. 要运行推理，请使用以下命令。将外部 IP 地址替换为上一步中的外部链接 IP 地址。

```
curl -d '{"instances": [1.0, 2.0, 5.0]}' -X POST http://<External ip>:8501/v1/models/saved_model_half_plus_two:predict
```

下面是示例输出。

```
{
    "predictions": [2.5, 3.0, 4.5
    ]
}
```

> **Important**
>
> 如果您无法连接到外部 IP 地址，请确保您的公司防火墙没有阻塞非标准端口，例如 8501。您可以尝试切换至来宾网络来验证。

基于 GPU 的推理

使用以下示例运行基于 GPU 的推理。

1. 使用以下内容创建名为 `ecs-dlc-gpu-inference-taskdef.json` 的文件。你可以将其与 TensorFlow 或 TensorFlow 2 一起使用。要将其与 TensorFlow 2 一起使用，请将 Docker 镜像更改为 TensorFlow 2 镜像，然后克隆 r2.0 服务存储库分支而不是 r1.15。

```
{
 "requiresCompatibilities": [
  "EC2"
 ],
 "containerDefinitions": [{
  "command": [
   "mkdir -p /test && cd /test && git clone -b r1.15 https://github.com/
tensorflow/serving.git && tensorflow_model_server --port=8500 --rest_api_port=8501
 --model_name=saved_model_half_plus_two --model_base_path=/test/serving/
tensorflow_serving/servables/tensorflow/testdata/saved_model_half_plus_two_gpu"
  ],
  "entryPoint": [
   "sh",
   "-c"
  ],
  "name": "tensorflow-inference-container",
  "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference:1.15.0-gpu-py36-cu100-ubuntu18.04",
  "memory": 8111,
  "cpu": 256,
  "resourceRequirements": [{
```

```
    "type": "GPU",
    "value": "1"
   }],
   "essential": true,
   "portMappings": [{
     "hostPort": 8500,
     "protocol": "tcp",
     "containerPort": 8500
    },
    {
     "hostPort": 8501,
     "protocol": "tcp",
     "containerPort": 8501
    },
    {
     "containerPort": 80,
     "protocol": "tcp"
    }
   ],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-group": "/ecs/TFInference",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs",
     "awslogs-create-group": "true"
    }
   }
  }],
  "volumes": [],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "TensorFlowInference"
}
```

2. 注册任务定义。记下输出中的修订版号，并在下一步中使用它。

```
aws ecs register-task-definition --cli-input-json file://ecs-dlc-gpu-inference-taskdef.json
```

3. 创建 Amazon ECS 服务 指定任务定义时，请`revision_id`用上一步输出中任务定义的修订版号替换。

```
aws ecs create-service --cluster ecs-ec2-training-inference \
                       --service-name cli-ec2-inference-gpu \
                       --task-definition Ec2TFInference:revision_id \
                       --desired-count 1 \
                       --launch-type EC2 \
                       --scheduling-strategy="REPLICA" \
                       --region us-east-1
```

4. 通过完成以下步骤来验证服务并获取网络终端节点。

   a. 在 https://console.aws.amazon.com/ecs/v2 中打开控制台。

   b. 选择 `ecs-ec2-training-inference` 集群。

   c. 在 Cluster (集群) 页面上，选择 Services (服务)，然后选择 cli-ec2-inference-cpu。

   d. 任务处于`RUNNING`状态后，选择任务标识符。

   e. 在 “日志” 下，选择 “查看登录信息” CloudWatch。这会将您带到 CloudWatch 控制台以查看训练进度日志。

   f. 在 Containers (容器) 下，展开容器详细信息。

   g. 在 “名称” 和 “网络绑定” 下，在 “外部链接” 下记下端口 8501 的 IP 地址，并在下一步中使用该地址。

5. 要运行推理，请使用以下命令。将外部 IP 地址替换为上一步中的外部链接 IP 地址。

```
curl -d '{"instances": [1.0, 2.0, 5.0]}' -X POST http://<External ip>:8501/v1/
models/saved_model_half_plus_two:predict
```

下面是示例输出。

```
{
    "predictions": [2.5, 3.0, 4.5
    ]
}
```

> **Important**
>
> 如果您无法连接到外部 IP 地址，请确保您的公司防火墙没有阻塞非标准端口，例如 8501。您可以尝试切换至来宾网络来验证。

自定义入口点

对于某些图像，Deep Learning Containers 使用自定义入口点脚本。如果您要使用自己的入口点，可以按如下方式覆盖入口点。

修改包含任务定义的 JSON 文件中的`entryPoint`参数。包括自定义入口点脚本的文件路径。此处显示了一个示例 。

```
"entryPoint":[
           "sh",
           "-c",
           "/usr/local/bin/multi-model-server --start --foreground --mms-config /home/
model-server/config.properties --models densenet=https://dlc-samples.s3.amazonaws.com/
pytorch/multi-model-server/densenet/densenet.mar"],
```

亚马逊 EKS 教程

Amazon EKS 教程提供了训练和推理示例，并展示了如何在以下设备上设置和使用 Dee AWS p Learning Containers：

- Amazon Elastic Kubernetes Service(Amazon EKS)
- Kubeflow 开启 AWS

Kubeflow on AWS 是适用于亚马逊 Elastic Kubernetes Service（亚马逊 EKS）的 Kubeflow 经过优化的开源发行版。有关更多信息，请参阅 Kubeflow 的AWS 功能。

> **Note**
>
> 本节中的所有训练和推理示例都在单节点集群上运行。

上的 Kubeflow 安装说明 AWS 提供了在部署 Kubeflow AWS 发行版之前创建 Amazon EKS 集群的步骤。

内容

- 亚马逊 EKS 设置
- 自定义入口点
- 对 EKS 上的 AWS 深度学习容器进行故障排除

亚马逊 EKS 设置

本节提供安装说明，用于在亚马逊 Elastic Kubernetes Service（亚马逊 EKS）上设置运行 AWS 深度学习容器的深度学习环境。

许可

要使用 GPU 硬件，请使用具有必要 GPU 驱动程序的 Amazon 系统映像。我们建议使用支持 GPU 的 Amazon EKS 优化版 AMI，本指南的后续步骤将使用该功能。此 AMI 包括不包含的软件 AWS，因此需要最终用户许可协议 (EULA)。您必须在中订阅 EKS 优化的 AMI AWS Marketplace 并接受 EULA，然后才能在工作节点组中使用 AMI。

> **Important**
>
> 要订阅 AMI，请访问 AWS Marketplace。

配置安全设置

要使用 Amazon EKS，您必须拥有一个有权访问多项安全权限的用户账户。这些是使用 AWS Identity and Access Management (IAM) 工具设置的。

1. 按照在您的 AWS 账户中创建 IAM 用户中的步骤创建 IAM 用户或更新现有的 IAM 用户。
2. 获取此用户的凭证。
   a. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 在下方`Users`，选择您的用户。
   c. 选择 `Security Credentials`。
   d. 选择 `Create access key`。
   e. 下载 key pair 或复制信息以备日后使用。
3. 将以下策略添加到您的 IAM 用户。这些策略为亚马逊 EKS、IAM 和亚马逊弹性计算云 (Amazon EC2) 提供了所需的访问权限。
   a. 选择 `Permissions`。
   b. 选择 `Add permissions`。
   c. 选择 `Create policy`。
   d. 从`Create policy`窗口中选择`JSON`选项卡。

e. 粘贴以下内容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

f. 命名策略EKSFullAccess并创建策略。

g. 导航回Grant permissions窗口。

h. 选择 Attach existing policies directly。

i. 搜索EKSFullAccess并选中该复选框。

j. 搜索AWSCloudFormationFullAccess并选中该复选框。

k. 搜索AmazonEC2FullAccess并选中该复选框。

l. 搜索IAMFullAccess并选中该复选框。

m. 搜索AmazonEC2ContainerRegistryReadOnly并选中复选框。

n. 搜索AmazonEKS_CNI_Policy并选中复选框。

o. 搜索AmazonS3FullAccess并选中复选框。

p. 接受更改。

网关节点

要设置 Amazon EKS 集群，请使用开源工具eksctl。我们建议您使用带有深度学习基础 AMI (Ubuntu) 的 Amazon EC2 实例来分配和控制您的集群。您可以在计算机上本地运行这些工具，也可以在已经运行的 Amazon EC2 实例上运行这些工具。但是，为了简化本指南，我们假设您在Ubuntu 16.04中使用的是深度学习基础AMI（DLAMI）。我们称之为您的网关节点。

在开始之前，请考虑您的训练数据的位置或您要运行集群以响应推理请求的位置。通常情况下，您的用于训练或推理的数据和集群应位于同一区域。此外，您还可以在同一区域启动网关节点。您可以按照这个 10 分钟的快速教程进行操作，该教程将指导您启动 DLAMI 以用作网关节点。

1. 登录到您的网关节点。
2. 安装或升级 AWS CLI。要访问所需的新 Kubernetes 功能，您必须具有最新版本。

```
$ sudo pip install --upgrade awscli
```

3. `eksctl`通过运行 Amazon EKS 用户指南安装说明中与您的操作系统对应的命令进行安装。有关的更多信息`eksctl`，另请参阅 eksctl 文档。
4. `kubectl`按照安装 kubectl 指南中的步骤进行安装。

> **Note**
>
> 您使用的`kubectl`版本必须与 Amazon EKS 集群控制平面版本相差不到一个小版本。例如，1.18 `kubectl` 客户端可与 Kubernetes 1.17、1.18 和 1.19 集群配合使用。

5. 通过运行以下命令安装 `aws-iam-authenticator`。有关的更多信息 aws-iam-authenticator，请参阅安装`aws-iam-authenticator`。

```
$ curl -o aws-iam-authenticator https://amazon-eks.s3.us-
west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/aws-iam-authenticator
$ chmod +x aws-iam-authenticator
$ cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/
bin:$PATH
```

6. 从“Security Configuration (安全配置)”部分中运行 IAM 用户的 `aws configure`。您正在复制 IAM 用户的 AWS 访问 AWS 密钥，然后复制您在 IAM 控制台中访问的私有访问密钥，然后将其粘贴到的提示中。`aws configure`

GPU 集群

1. 检查以下使用 p3.8xlarge 实例类型创建集群的命令。在运行它之前，必须进行以下修改。
   - `name`是您用来管理集群的工具。您可以将 `cluster-name` 更改为希望的任何名称，只要其中没有空格或特殊字符。
   - `eks-version`是 Amazon E KS kubernetes 版本。有关支持的亚马逊 EKS 版本，请参阅可用的亚马逊 EKS Kubernetes 版本。
   - `nodes`是您想要在集群中安装的实例数量。在本示例中，我们将从三个节点开始。
   - `node-type`指的是一个实例类。

- `timeout`而且`*ssh-access *`可以不管。
- `ssh-public-key`是您要用来登录工作节点的密钥的名称。要么使用你已经使用的安全密钥，要么创建一个新的安全密钥，但一定要 ssh-public-key用为你使用的区域分配的密钥替换。注意：您只需要提供在 Ama EC2 zon 控制台的 “密钥对” 部分中显示的密钥名称。
- `region`是启动集群的 Amazon EC2 区域。如果您计划使用位于特定区域（除外*<us-east-1>*）的训练数据，我们建议您使用同一区域。 ssh-public-key必须有权在该地区启动实例。

> **Note**
>
> 本指南的其余部分假设*<us-east-1>*为区域。

2. 对命令进行更改后，运行该命令并等待。单节点群集可能需要几分钟，如果您选择创建大型集群，则可能需要更长的时间。

```
$ eksctl create cluster <cluster-name> \
                      --version <eks-version> \
                      --nodes 3 \
                      --node-type=<p3.8xlarge> \
                      --timeout=40m \
                      --ssh-access \
                      --ssh-public-key <key_pair_name> \
                      --region <us-east-1> \
                      --zones=us-east-1a,us-east-1b,us-east-1d \
                      --auto-kubeconfig
```

您应该可以看到类似于如下输出的内容：

```
EKS cluster "training-1" in "us-east-1" region is ready
```

3. 理想情况下，auto-kubeconfig 应已配置您的集群。但是，如果您遇到问题，则可以运行以下命令来设置您的 kubeconfig。如果您要从其他位置更改网关节点和管理集群，也可使用此命令。

```
$ aws eks --region <region> update-kubeconfig --name <cluster-name>
```

您应该可以看到类似于如下输出的内容：

```
Added new context arn:aws:eks:us-east-1:999999999999:cluster/training-1 to /home/
ubuntu/.kube/config
```

4. 如果您计划使用 GPU 实例类型，请务必使用以下命令在集群上运行适用于 Kubernetes 的 NVIDIA 设备插件：

```
$ kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/
v1.12/nvidia-device-plugin.yml
$ kubectl create -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/
v0.9.0/nvidia-device-plugin.yml
```

5. 验证集群中每个节点上都有 GPUs 可用的

```
$ kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

CPU 集群

请参阅前一节关于使用eksctl命令启动 GPU 集群并修改`node-type`为使用 CPU 实例类型的讨论。

哈瓦那集群

请参阅前面关于使用eksctl命令启动 GPU 集群的讨论，并修改`node-type`为使用带有 Habana Gaudi 加速器的实例（例如实例类型）。DL1

测试您的集群

1. 您可以在集群上运行kubectl命令以检查其状态。试用该命令以确保其选择的是您要管理的当前集群。

```
$ kubectl get nodes -o wide
```

2. 简单了解 ~/.kube。此目录具有用于从您的网关节点配置的各个集群的 kubeconfig 文件。如果您进一步浏览该文件夹，则可以找到 ~/。 kube/eksctl/clusters-它保存了使用 eksctl 创建的集群的 kubeconfig 文件。此文件包含一些您理想情况下不必修改的细节，因为这些工具会为您生成和更新配置，但是在故障排除时最好参考一下。
3. 验证集群是否处于活动状态。

```
$ aws eks --region <region> describe-cluster --name <cluster-name> --query
 cluster.status
```

您应看到以下输出：

```
"ACTIVE"
```

4. 如果您在同一主机实例中具有多个集群设置，请验证 kubectl 上下文。有时，确保找到的默认上下文设置kubectl正确会有所帮助。使用以下命令检查此内容：

```
$ kubectl config get-contexts
```

5. 如果未按预期设置该上下文，请使用以下命令修复此问题：

```
$ aws eks --region <region> update-kubeconfig --name <cluster-name>
```

管理您的集群

当你想控制或查询集群时，你可以使用 kubeconfig 参数通过配置文件对其进行寻址。这在您有多个集群时很有用。例如，如果您有一个名为 “training-gpu-1” 的单独集群，则可以通过将配置文件作为参数传递来对其调用get pods命令，如下所示：

```
$ kubectl --kubeconfig=/home/ubuntu/.kube/eksctl/clusters/training-gpu-1 get pods
```

值得注意的是，你可以在不使用 kubeconfig 参数的情况下运行同样的命令。在这种情况下，该命令将使用当前主动控制的集群 (`current-context`)。

```
$ kubectl get pods
```

如果您设置了多个集群，而这些集群尚未安装 NVIDIA 插件，则可以采用以下方式安装该插件：

```
$ kubectl --kubeconfig=/home/ubuntu/.kube/eksctl/clusters/training-gpu-1 create -f
 https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/nvidia-device-
plugin.yml
```

您还可以通过更新 kubeconfig、传递要管理的集群的名称来更改活动集群。以下命令更新 kubeconfig 且无需使用 kubeconfig 参数。

```
$ aws eks —region us-east-1 update-kubeconfig —name training-gpu-1
```

如果您遵循本指南中的所有示例，则可能会经常在活动集群之间切换。这样您就可以编排训练或推理，或者使用在不同集群上运行的不同框架。

清理

使用完集群后，请将其删除，以免产生额外费用。

```
$ eksctl delete cluster --name=<cluster-name>
```

要仅删除 pod，请运行以下命令：

```
$ kubectl delete pods <name>
```

要重置访问集群的密钥，请运行以下命令：

```
$ kubectl delete secret ${SECRET} -n ${NAMESPACE} || true
```

要删除nodegroup连接到集群的连接，请运行以下命令：

```
$ eksctl delete nodegroup --name <cluster_name>
```

要将nodegroup连接到集群，请运行以下命令：

```
$ eksctl create nodegroup
                      --cluster <cluster-name> \
                      --node-ami <ami_id> \
                      --nodes <num_nodes> \
                      --node-type=<instance_type> \
                      --timeout=40m \
                      --ssh-access \
                      --ssh-public-key <key_pair_name> \
                      --region <us-east-1> \
                      --auto-kubeconfig
```

后续步骤

要了解有关在 Amazon EKS 上使用 Deep Learning Containers 进行训练和推理的信息，请访问训练或推理。

内容

- 训练
- 推理

训练

使用中的步骤创建集群后亚马逊 EKS 设置，即可使用它来运行训练作业。对于训练，您可以使用 CPU、GPU 或分布式 GPU 示例，具体取决于集群中的节点。以下各节中的主题展示了如何使用 PyTorch 训练示例。

内容

- CPU 训练
- GPU 训练
- 分布式 GPU 训练

CPU 训练

本节用于有关基于 CPU 的容器的培训。

内容

- PyTorch CPU 训练
- TensorFlow CPU 训练
- 后续步骤

PyTorch CPU 训练

本教程将指导您在单节点 CPU Pod 上训练 PyTorch 模型。

1. 为您的集群创建 pod 文件。pod 文件将提供有关集群应运行什么的说明。此 pod 文件将下载 PyTorch存储库并运行 MNIST 示例。打开vi或vim，然后复制并粘贴以下内容。将此文件另存为 `pytorch.yaml`。

```
apiVersion: v1
kind: Pod
metadata:
  name: pytorch-training
spec:
```

```
  restartPolicy: OnFailure
  containers:
  - name: pytorch-training
    image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.5.1-cpu-
py36-ubuntu16.04
    command:
      - "/bin/sh"
      - "-c"
    args:
    - "git clone https://github.com/pytorch/examples.git && python examples/mnist/
main.py --no-cuda"
    env:
    - name: OMP_NUM_THREADS
      value: "36"
    - name: KMP_AFFINITY
      value: "granularity=fine,verbose,compact,1,0"
    - name: KMP_BLOCKTIME
      value: "1"
```

2. 使用将 pod 文件分配给集群kubectl。

```
$ kubectl create -f pytorch.yaml
```

3. 您应看到以下输出：

```
pod/pytorch-training created
```

4. 检查状态。作业“pytorch-training”的名称位于 pytorch.yaml 文件中。它现在将显示在状态中。如果您正在运行任何其他测试或以前运行过某些测试，则它会出现在此列表中。多次运行此项，直到您看到状态更改为“Running (正在运行)”。

```
$ kubectl get pods
```

您应看到以下输出：

```
NAME READY STATUS RESTARTS AGE
pytorch-training 0/1 Running 8 19m
```

5. 检查日志以查看训练输出。

```
$ kubectl logs pytorch-training
```

您应该可以看到类似于如下输出的内容：

```
Cloning into 'examples'...
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/
MNIST/raw/train-images-idx3-ubyte.gz
9920512it [00:00, 40133996.38it/s]
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/
MNIST/raw/train-labels-idx1-ubyte.gz
Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw
32768it [00:00, 831315.84it/s]
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/
MNIST/raw/t10k-images-idx3-ubyte.gz
1654784it [00:00, 13019129.43it/s]
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/
MNIST/raw/t10k-labels-idx1-ubyte.gz
8192it [00:00, 337197.38it/s]
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw
Processing...
Done!
Train Epoch: 1 [0/60000 (0%)]    Loss: 2.300039
Train Epoch: 1 [640/60000 (1%)]    Loss: 2.213470
Train Epoch: 1 [1280/60000 (2%)]    Loss: 2.170460
Train Epoch: 1 [1920/60000 (3%)]    Loss: 2.076699
Train Epoch: 1 [2560/60000 (4%)]    Loss: 1.868078
Train Epoch: 1 [3200/60000 (5%)]    Loss: 1.414199
Train Epoch: 1 [3840/60000 (6%)]    Loss: 1.000870
```

6. 查看日志以查看训练进度。您也可以继续选中 “get pods” 以刷新状态。当状态更改为 “`Completed`” 时，您将知道训练工作已完成。

TensorFlow CPU 训练

本教程将指导您在单节点 CPU 集群上训练 TensorFlow 模型。

1. 为您的集群创建 pod 文件。pod 文件将提供有关集群应运行什么的说明。此 pod 文件将下载 Keras 并运行 Keras 示例。此示例使用 TensorFlow 框架。打开vi或vim并复制并粘贴以下内容。将此文件另存为 `tf.yaml`。你可以将其与 TensorFlow或 TensorFlow 2 一起使用。要将其与 TensorFlow 2 一起使用，请将 Docker 镜像更改为 TensorFlow 2 镜像。

```
apiVersion: v1
  kind: Pod
  metadata:
  name: tensorflow-training
  spec:
  restartPolicy: OnFailure
  containers:
  - name: tensorflow-training
  image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference:1.15.2-
cpu-py36-ubuntu18.04
  command: ["/bin/sh","-c"]
  args: ["git clone https://github.com/fchollet/keras.git && python /keras/
examples/mnist_cnn.py"]
```

2. 使用将 pod 文件分配给集群kubectl。

```
$ kubectl create -f tf.yaml
```

3. 您应看到以下输出：

```
pod/tensorflow-training created
```

4. 检查状态。任务“tensorflow-training”的名称位于 tf.yaml 文件中。它现在将显示在状态中。如果您正在运行任何其他测试或以前运行过某些测试，则它会出现在此列表中。多次运行此项，直到您看到状态更改为“Running (正在运行)”。

```
$ kubectl get pods
```

您应看到以下输出：

```
NAME READY STATUS RESTARTS AGE
  tensorflow-training 0/1 Running 8 19m
```

5. 检查日志以查看训练输出。

```
$ kubectl logs tensorflow-training
```

您应该可以看到类似于如下输出的内容：

```
Cloning into 'keras'...
  Using TensorFlow backend.
  Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz

  8192/11490434 [..............................] - ETA: 0s
  6479872/11490434 [===============>..............] - ETA: 0s
  8740864/11490434 [=====================>........] - ETA: 0s
  11493376/11490434 [==============================] - 0s 0us/step
  x_train shape: (60000, 28, 28, 1)
  60000 train samples
  10000 test samples
  Train on 60000 samples, validate on 10000 samples
  Epoch 1/12
  2019-03-19 01:52:33.863598: I tensorflow/core/platform/cpu_feature_guard.cc:141]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
 use: AVX512F
  2019-03-19 01:52:33.867616: I tensorflow/core/common_runtime/process_util.cc:69]
 Creating new thread pool with default inter op setting: 2. Tune using
 inter_op_parallelism_threads for best performance.

  128/60000 [..............................] - ETA: 10:43 - loss: 2.3076 - acc:
 0.0625
  256/60000 [..............................] - ETA: 5:59 - loss: 2.2528 - acc:
 0.1445
  384/60000 [..............................] - ETA: 4:24 - loss: 2.2183 - acc:
 0.1875
  512/60000 [..............................] - ETA: 3:35 - loss: 2.1652 - acc:
 0.1953
  640/60000 [..............................] - ETA: 3:05 - loss: 2.1078 - acc:
 0.2422
  ...
```

6. 您可以检查日志以观察训练进度。您也可以继续选中 “get pods” 以刷新状态。当状态更改为 “`Completed`” 时，您将知道训练工作已完成。

后续步骤

要使用 Deep Learning Containers 在 Amazon EKS 上学习基于 CPU TensorFlow 的推理，请参阅。TensorFlow CPU 推断

后续步骤

要使用 Deep Learning Containers 在 Amazon EKS 上学习基于 CPU PyTorch 的推理，请参阅。PyTorch CPU 推断

GPU 训练

本部分用于在基于 GPU 的集群上进行培训。

内容

- PyTorch GPU 训练
- TensorFlow GPU 训练

PyTorch GPU 训练

本教程将指导您在单节点 GPU 集群 PyTorch 上进行训练。

1. 为您的集群创建 pod 文件。pod 文件将提供有关集群应运行什么的说明。此 pod 文件将下载 PyTorch存储库并运行 MNIST 示例。打开vi或vim，然后复制并粘贴以下内容。将此文件另存为 `pytorch.yaml`。

```
apiVersion: v1
kind: Pod
metadata:
  name: pytorch-training
spec:
  restartPolicy: OnFailure
  containers:
  - name: pytorch-training
    image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.5.1-gpu-
py36-cu101-ubuntu16.04
    command:
      - "/bin/sh"
      - "-c"
    args:
    - "git clone https://github.com/pytorch/examples.git && python examples/mnist/
main.py --no-cuda"
    env:
    - name: OMP_NUM_THREADS
      value: "36"
    - name: KMP_AFFINITY
```

```
      value: "granularity=fine,verbose,compact,1,0"
    - name: KMP_BLOCKTIME
      value: "1"
```

2. 使用将 pod 文件分配给集群kubectl。

```
$ kubectl create -f pytorch.yaml
```

3. 您应看到以下输出：

```
pod/pytorch-training created
```

4. 检查状态。作业“pytorch-training”的名称位于 pytorch.yaml 文件中。它现在将显示在状态中。如果您正在运行任何其他测试或以前运行过某些测试，则它会出现在此列表中。多次运行此命令，直到看到状态更改为 “Running”。

```
$ kubectl get pods
```

您应看到以下输出：

```
NAME READY STATUS RESTARTS AGE
pytorch-training 0/1 Running 8 19m
```

5. 检查日志以查看训练输出。

```
$ kubectl logs pytorch-training
```

您应该可以看到类似于如下输出的内容：

```
Cloning into 'examples'...
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/
MNIST/raw/train-images-idx3-ubyte.gz
9920512it [00:00, 40133996.38it/s]
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/
MNIST/raw/train-labels-idx1-ubyte.gz
Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw
32768it [00:00, 831315.84it/s]
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/
MNIST/raw/t10k-images-idx3-ubyte.gz
1654784it [00:00, 13019129.43it/s]
```

```
Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/
MNIST/raw/t10k-labels-idx1-ubyte.gz
8192it [00:00, 337197.38it/s]
Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw
Processing...
Done!
Train Epoch: 1 [0/60000 (0%)]    Loss: 2.300039
Train Epoch: 1 [640/60000 (1%)]    Loss: 2.213470
Train Epoch: 1 [1280/60000 (2%)]    Loss: 2.170460
Train Epoch: 1 [1920/60000 (3%)]    Loss: 2.076699
Train Epoch: 1 [2560/60000 (4%)]    Loss: 1.868078
Train Epoch: 1 [3200/60000 (5%)]    Loss: 1.414199
Train Epoch: 1 [3840/60000 (6%)]    Loss: 1.000870
```

6. 查看日志以查看训练进度。您也可以继续选中 “get pods” 以刷新状态。当状态变为 `Completed` “” 时，训练工作就完成了。

后续步骤

要使用 Deep Learning Containers 在 Amazon EKS 上学习基于 GPU PyTorch 的推理，请参阅。PyTorch GPU 推理

TensorFlow GPU 训练

本教程将指导您在单节点 GPU 集群上训练 TensorFlow 模型。

1. 为您的集群创建 pod 文件。pod 文件将提供有关集群应运行什么的说明。此 pod 文件将下载 Keras 并运行 Keras 示例。此示例使用 TensorFlow 框架。打开vi或vim并复制并粘贴以下内容。将此文件另存为 `tf.yaml`。你可以将其与 TensorFlow或 TensorFlow 2 一起使用。要将其与 TensorFlow 2 一起使用，请将 Docker 镜像更改为 TensorFlow 2 镜像。

```
apiVersion: v1
kind: Pod
metadata:
name: tensorflow-training
spec:
restartPolicy: OnFailure
containers:
- name: tensorflow-training
image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training:1.15.2-gpu-
py37-cu100-ubuntu18.04
```

```
command: ["/bin/sh","-c"]
args: ["git clone https://github.com/fchollet/keras.git && python /keras/examples/
mnist_cnn.py"]
resources:
  limits:
    nvidia.com/gpu: 1
```

2. 使用将 pod 文件分配给集群kubectl。

```
$ kubectl create -f tf.yaml
```

3. 您应看到以下输出：

```
pod/tensorflow-training created
```

4. 检查状态。任务“tensorflow-training”的名称位于 tf.yaml 文件中。它现在将显示在状态中。如果您正在运行任何其他测试或以前运行过某些测试，则它会出现在此列表中。多次运行此命令，直到看到状态更改为 “Running”。

```
$ kubectl get pods
```

您应看到以下输出：

```
NAME READY STATUS RESTARTS AGE
tensorflow-training 0/1 Running 8 19m
```

5. 检查日志以查看训练输出。

```
$ kubectl logs tensorflow-training
```

您应该可以看到类似于如下输出的内容：

```
Cloning into 'keras'...
Using TensorFlow backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz

8192/11490434 [..............................] - ETA: 0s
6479872/11490434 [===============>..............] - ETA: 0s
8740864/11490434 [======================>........] - ETA: 0s
11493376/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
```

```
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
2019-03-19 01:52:33.863598: I tensorflow/core/platform/cpu_feature_guard.cc:141]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
 use: AVX512F
2019-03-19 01:52:33.867616: I tensorflow/core/common_runtime/process_util.cc:69]
 Creating new thread pool with default inter op setting: 2. Tune using
 inter_op_parallelism_threads for best performance.

128/60000 [..............................] - ETA: 10:43 - loss: 2.3076 - acc:
 0.0625
256/60000 [..............................] - ETA: 5:59 - loss: 2.2528 - acc: 0.1445
384/60000 [..............................] - ETA: 4:24 - loss: 2.2183 - acc: 0.1875
512/60000 [..............................] - ETA: 3:35 - loss: 2.1652 - acc: 0.1953
640/60000 [..............................] - ETA: 3:05 - loss: 2.1078 - acc: 0.2422
...
```

6. 查看日志以查看训练进度。您也可以继续选中 “get pods” 以刷新状态。当状态变为 “`Completed`” 时，训练作业就完成了。

后续步骤

要使用 Deep Learning Containers 在 Amazon EKS 上学习基于 GPU TensorFlow 的推理，请参阅。TensorFlow GPU 推理

分布式 GPU 训练

本部分针对在多节点 GPU 集群上运行分布式训练。

内容

- 设置集群以进行分布式训练
- PyTorch分布式 GPU 训练

设置集群以进行分布式训练

要在 EKS 上运行分布式训练，您需要在集群上安装以下组件。

- Kubeflow 的默认安装，其中包含所需的组件，例如 PyTorch 运算符和 NVIDIA 插件。
- MPI 运算符。

下载并运行脚本以在集群中安装所需的组件。

```
$ wget -O install_kubeflow.sh https://raw.githubusercontent.com/aws/deep-
learning-containers/master/test/dlc_tests/eks/eks_manifest_templates/kubeflow/
install_kubeflow.sh
$ chmod +x install_kubeflow.sh
$ ./install_kubeflow.sh <EKS_CLUSTER_NAME> <AWS_REGION>
```

PyTorch分布式 GPU 训练

本教程将指导您在多节点 GPU 集群 PyTorch 上进行分布式训练。它使用 Gloo 作为后端。

1. 确认已安装 PyTorch 自定义资源。

   ```
   $ kubectl get crd
   ```

   该输出应包含 `pytorchjobs.kubeflow.org`。

2. 确保 NVIDIA 插件`daemonset`正在运行。

   ```
   $ kubectl get daemonset -n kubeflow
   ```

   输出应类似于以下内容。

   ```
   NAME                          DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE
    SELECTOR AGE
   nvidia-device-plugin-daemonset   3         3      3        3        3    <none>
    35h
   ```

3. 使用以下文本创建基于 Gloo 的分布式数据并行作业。将其保存在名为的文件中`distributed.yaml`。

   ```
   apiVersion: kubeflow.org/v1
   kind: PyTorchJob
   metadata:
     name: "kubeflow-pytorch-gpu-dist-job"
   spec:
     pytorchReplicaSpecs:
       Master:
         replicas: 1
   ```

```
      restartPolicy: OnFailure
      template:
        spec:
          containers:
          - name: "pytorch"
            image: "763104351884.dkr.ecr.us-east-1.amazonaws.com/aws-samples-
pytorch-training:1.7.1-gpu-py36-cu110-ubuntu18.04-example"
            args:
              - "--backend"
              - "gloo"
              - "--epochs"
              - "5"
    Worker:
      replicas: 2
      restartPolicy: OnFailure
      template:
        spec:
          containers:
          - name: "pytorch"
            image: "763104351884.dkr.ecr.us-east-1.amazonaws.com/aws-samples-
pytorch-training:1.7.1-gpu-py36-cu110-ubuntu18.04-example"
            args:
              - "--backend"
              - "gloo"
              - "--epochs"
              - "5"
            resources:
              limits:
                nvidia.com/gpu: 1
```

4. 使用您刚刚创建的 pod 文件运行分布式训练作业。

```
$ kubectl create -f distributed.yaml
```

5. 您可以使用以下方法检查作业的状态：

```
$ kubectl logs kubeflow-pytorch-gpu-dist-job
```

要连续查看日志，请使用：

```
$ kubectl logs -f <pod>
```

推理

使用中的步骤创建集群后亚马逊 EKS 设置，即可使用它来运行推理作业。为了进行推理，您可以根据集群中的节点使用 CPU 或 GPU 示例。推理仅支持单节点配置。以下主题介绍如何使用和在 EKS 上使用 PyTorch Dee AWS p Learning Containers 运行推理。 TensorFlow

内容

- CPU 推理
- GPU 推理

CPU 推理

本节将指导你使用 PyTorch和在适用于 EKS CPU 集群的 Deep Learning Containers 上运行推理。 TensorFlow

有关深度学习容器的完整列表，请参阅可用的深度学习容器映像。

内容

- PyTorch CPU 推断
- TensorFlow CPU 推断
- 后续步骤

PyTorch CPU 推断

在这种方法中，您可以创建一个 Kubernetes 服务和一个用于运行 CPU 推理的部署。 PyTorchKubernetes 服务公开了一个进程及其端口。在创建 Kubernetes 服务时，您可以指定要使用的服务类型。`ServiceTypes`默认 `ServiceType` 为 `ClusterIP`。部署负责确保一定数量的 pod 始终处于启动和运行状态。

1. 创建命名空间。你可能需要更改 kubeconfig 以指向正确的集群。确认您已设置了 “training-cpu-1” 或将其更改为 CPU 集群的配置。有关设置集群的更多信息，请参阅亚马逊 EKS 设置。

   ```
   $ NAMESPACE=pt-inference; kubectl create namespace ${NAMESPACE}
   ```

2. （使用公共模型时的可选步骤。）在可安装的网络位置设置模型，例如在 Amazon S3 中。有关如何将经过训练的模型上传到 S3 的信息，请参阅TensorFlow CPU 推断。将密钥应用于您的命名空间。有关密钥的更多信息，请参阅 Kubernetes 密钥文档。

```
$ kubectl -n ${NAMESPACE} apply -f secret.yaml
```

3. 使用以下内容创建名为 pt_inference.yaml 的文件。此示例文件指定了模型、使用的 PyTorch 推理图像以及模型的位置。此示例使用公共模型，因此您无需对其进行修改。

```
---
kind: Service
apiVersion: v1
metadata:
  name: densenet-service
  labels:
    app: densenet-service
spec:
  ports:
  - port: 8080
    targetPort: mms
  selector:
    app: densenet-service
---
kind: Deployment
apiVersion: apps/v1
metadata:
  name: densenet-service
  labels:
    app: densenet-service
spec:
  replicas: 1
  selector:
    matchLabels:
      app: densenet-service
  template:
    metadata:
      labels:
        app: densenet-service
    spec:
      containers:
      - name: densenet-service
        image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-
inference:1.3.1-cpu-py36-ubuntu16.04
        args:
        - multi-model-server
        - --start
```

```
        - --mms-config /home/model-server/config.properties
        - --models densenet=https://dlc-samples.s3.amazonaws.com/pytorch/multi-model-server/densenet/densenet.mar
        ports:
        - name: mms
          containerPort: 8080
        - name: mms-management
          containerPort: 8081
        imagePullPolicy: IfNotPresent
```

4. 将配置应用于之前定义的命名空间中的新 pod。

```
$ kubectl -n ${NAMESPACE} apply -f pt_inference.yaml
```

您的输出应类似于以下内容：

```
service/densenet-service created
deployment.apps/densenet-service created
```

5. 检查 Pod 的状态并等待 Pod 处于 “RUNNING” 状态：

```
$ kubectl get pods -n ${NAMESPACE} -w
```

您的输出应类似于以下内容：

```
NAME                     READY     STATUS    RESTARTS   AGE
densenet-service-xvw1    1/1       Running   0          3m
```

6. 要进一步描述 pod，请运行以下命令：

```
$ kubectl describe pod <pod_name> -n ${NAMESPACE}
```

7. 由于此处的服务类型是 clusterIP，因此您可以将端口从容器转发到主机。

```
$ kubectl port-forward -n ${NAMESPACE} `kubectl get pods -n ${NAMESPACE} --
selector=app=densenet-service -o jsonpath='{.items[0].metadata.name}'` 8080:8080 &
```

8. 服务器启动后，您现在可以使用以下命令从不同的窗口运行推理：

```
$ curl -O https://s3.amazonaws.com/model-server/inputs/flower.jpg
curl -X POST http://127.0.0.1:8080/predictions/densenet -T flower.jpg
```

使用完集群后，请参阅 EKS 清除以获取有关清除集群的信息。

TensorFlow CPU 推断

在本教程中，您将创建一个 Kubernetes 服务和一个用于运行 CPU 推理的部署。TensorFlowKubernetes 服务公开了一个进程及其端口。在创建 Kubernetes 服务时，您可以指定要使用的服务类型。`ServiceTypes`默认 `ServiceType` 为 `ClusterIP`。部署负责确保一定数量的 pod 始终处于启动和运行状态。

1. 创建命名空间。你可能需要更改 kubeconfig 以指向正确的集群。确认您已设置了 “training-cpu-1” 或将其更改为 CPU 集群的配置。有关设置集群的更多信息，请参阅亚马逊 EKS 设置。

```
$ NAMESPACE=tf-inference; kubectl –kubeconfig=/home/ubuntu/.kube/eksctl/clusters/
training-cpu-1 create namespace ${NAMESPACE}
```

2. 可以用不同的方式检索用于推理的模型，例如使用共享卷和 Amazon S3。由于 Kubernetes 服务需要访问亚马逊 S3 和亚马逊 ECR，因此您必须将您的 AWS 证书存储为 Kubernetes 密钥。在本示例中，使用 S3 存储和获取经过训练的模型。

   验证您的 AWS 凭证。他们必须具有 S3 写入权限。

```
$ cat ~/.aws/credentials
```

3. 该输出值将类似于以下内容：

```
$ [default]
aws_access_key_id = YOURACCESSKEYID
aws_secret_access_key = YOURSECRETACCESSKEY
```

4. 使用 base64 对这些凭证进行编码。

   首先编码访问密钥。

```
$ echo -n 'YOURACCESSKEYID' | base64
```

   接下来编码秘密访问密钥。

```
$ echo -n 'YOURSECRETACCESSKEY' | base64
```

   您的输出应类似于以下内容：

```
$ echo -n 'YOURACCESSKEYID' | base64
RkFLRUFXU0FDQ0VTU0tFWUlE
$ echo -n 'YOURSECRETACCESSKEY' | base64
RkFLRUFXU1NFQ1JFVEFDQ0VTU0tFWQ==
```

5. 在您的主目录中创建一个名为`secret.yaml`的文件，其中包含以下内容。此文件用于存储密钥。

```
apiVersion: v1
kind: Secret
metadata:
name: aws-s3-secret
type: Opaque
data:
AWS_ACCESS_KEY_ID: YOURACCESSKEYID
AWS_SECRET_ACCESS_KEY: YOURSECRETACCESSKEY
```

6. 将密钥应用于您的命名空间。

```
$ kubectl -n ${NAMESPACE} apply -f secret.yaml
```

7. 克隆 tensorf low 服务存储库。

```
$ git clone https://github.com/tensorflow/serving/
$ cd serving/tensorflow_serving/servables/tensorflow/testdata/
```

8. 将预训练`saved_model_half_plus_two_cpu`模型同步到您的 S3 存储桶。

```
$ aws s3 sync saved_model_half_plus_two_cpu s3://<your_s3_bucket>/
saved_model_half_plus_two
```

9. 使用以下内容创建名为 `tf_inference.yaml` 的文件。更新`--model_base_path`以使用您的 S3 存储桶。你可以将其与 TensorFlow 或 TensorFlow 2 一起使用。要将其与 TensorFlow 2 一起使用，请将 Docker 镜像更改为 TensorFlow 2 镜像。

```
---
kind: Service
apiVersion: v1
metadata:
name: half-plus-two
labels:
  app: half-plus-two
```

```
spec:
ports:
- name: http-tf-serving
  port: 8500
  targetPort: 8500
- name: grpc-tf-serving
  port: 9000
  targetPort: 9000
selector:
  app: half-plus-two
  role: master
type: ClusterIP
---
kind: Deployment
apiVersion: apps/v1
metadata:
name: half-plus-two
labels:
  app: half-plus-two
  role: master
spec:
replicas: 1
selector:
  matchLabels:
    app: half-plus-two
    role: master
template:
  metadata:
    labels:
      app: half-plus-two
      role: master
  spec:
    containers:
    - name: half-plus-two
      image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference:1.15.0-cpu-py36-ubuntu18.04
      command:
      - /usr/bin/tensorflow_model_server
      args:
      - --port=9000
      - --rest_api_port=8500
      - --model_name=saved_model_half_plus_two
      - --model_base_path=s3://tensorflow-trained-models/saved_model_half_plus_two
      ports:
```

```
      - containerPort: 8500
      - containerPort: 9000
      imagePullPolicy: IfNotPresent
      env:
      - name: AWS_ACCESS_KEY_ID
        valueFrom:
          secretKeyRef:
            key: AWS_ACCESS_KEY_ID
            name: aws-s3-secret
      - name: AWS_SECRET_ACCESS_KEY
        valueFrom:
          secretKeyRef:
            key: AWS_SECRET_ACCESS_KEY
            name: aws-s3-secret
      - name: AWS_REGION
        value: us-east-1
      - name: S3_USE_HTTPS
        value: "true"
      - name: S3_VERIFY_SSL
        value: "true"
      - name: S3_ENDPOINT
        value: s3.us-east-1.amazonaws.com
```

10. 将配置应用于之前定义的命名空间中的新 pod。

```
$ kubectl -n ${NAMESPACE} apply -f tf_inference.yaml
```

您的输出应类似于以下内容：

```
service/half-plus-two created
deployment.apps/half-plus-two created
```

11. 检查 Pod 的状态。

```
$ kubectl get pods -n ${NAMESPACE}
```

重复状态检查，直到看到以下 “正在运行” 状态：

```
NAME                     READY     STATUS    RESTARTS   AGE
half-plus-two-vmwp9  1/1       Running   0          3m
```

12. 要进一步描述 pod，您可以运行：

```
$ kubectl describe pod <pod_name> -n ${NAMESPACE}
```

13. 由于服务类型为 clusterIP，因此您可以将端口从容器转发到主机。

```
$ kubectl port-forward -n ${NAMESPACE} `kubectl get pods -n ${NAMESPACE} --
selector=app=half-plus-two -o jsonpath='{.items[0].metadata.name}'` 8500:8500 &
```

14. 将以下 json 字符串放在名为的文件中 `half_plus_two_input.json`

```
{"instances": [1.0, 2.0, 5.0]}
```

15. 在模型上运行推理。

```
$ curl -d @half_plus_two_input.json -X POST http://localhost:8500/v1/models/
saved_model_half_plus_two_cpu:predict
```

您的输出应与以下内容类似：

```
{
"predictions": [2.5, 3.0, 4.5
]
}
```

后续步骤

要了解如何在 Amazon EKS 上使用带有 Deep Learning Containers 的自定义入口点，请参阅。自定义入口点

GPU 推理

本节介绍如何在适用于 EKS GPU 集群的 Deep Learning Containers 上运行推理 PyTorch，以及。TensorFlow

有关深度学习容器的完整列表，请参阅可用的深度学习容器映像。

内容

- PyTorch GPU 推理
- TensorFlow GPU 推理
- 后续步骤

PyTorch GPU 推理

在此方法中，创建一个 Kubernetes 服务和部署。Kubernetes 服务公开了一个进程及其端口。在创建 Kubernetes 服务时，您可以指定要使用的服务类型。`ServiceTypes`默认 `ServiceType` 为 `ClusterIP`。部署负责确保一定数量的 pod 始终处于启动和运行状态。

1. 对于基于 GPU 的推理，安装适用于 Kubernetes 的 NVIDIA 设备插件。

```
$ kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/
v1.12/nvidia-device-plugin.yml
```

2. 验证运行 nvidia-device-plugin-daemonset是否正常。

```
$ kubectl get daemonset -n kube-system
```

该输出值将类似于以下内容。

```
NAME                              DESIRED   CURRENT   READY     UP-TO-DATE
 AVAILABLE   NODE SELECTOR   AGE
aws-node                          3         3         3         3            3
     <none>          6d
kube-proxy                        3         3         3         3            3
     <none>          6d
nvidia-device-plugin-daemonset    3         3         3         3            3
     <none>          57s
```

3. 创建命名空间。

```
$ NAMESPACE=pt-inference; kubectl create namespace ${NAMESPACE}
```

4. （使用公共模型时的可选步骤。）在可装载的网络位置（如 S3）处设置您的模型。请参阅使用推理一节中提及的将经过训练的模型上传到 S3 的步骤。 TensorFlow将密钥应用于您的命名空间。有关密钥的更多信息，请参阅 Kubernetes 密钥文档。

```
$ kubectl -n ${NAMESPACE} apply -f secret.yaml
```

5. 创建 `pt_inference.yaml` 文件。使用下一个代码块的内容作为其内容。

```
---
kind: Service
apiVersion: v1
```

```
metadata:
  name: densenet-service
  labels:
    app: densenet-service
spec:
  ports:
  - port: 8080
    targetPort: mms
  selector:
    app: densenet-service
---
kind: Deployment
apiVersion: apps/v1
metadata:
  name: densenet-service
  labels:
    app: densenet-service
spec:
  replicas: 1
  selector:
    matchLabels:
      app: densenet-service
  template:
    metadata:
      labels:
        app: densenet-service
    spec:
      containers:
      - name: densenet-service
        image: "763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-
inference:1.3.1-gpu-py36-cu101-ubuntu16.04"
        args:
        - multi-model-server
        - --start
        - --mms-config /home/model-server/config.properties
        - --models densenet=https://dlc-samples.s3.amazonaws.com/pytorch/multi-
model-server/densenet/densenet.mar
        ports:
        - name: mms
          containerPort: 8080
        - name: mms-management
          containerPort: 8081
        imagePullPolicy: IfNotPresent
        resources:
```

```
          limits:
            cpu: 4
            memory: 4Gi
            nvidia.com/gpu: 1
          requests:
            cpu: "1"
            memory: 1Gi
```

6. 将配置应用于之前定义的命名空间中的新 pod。

```
$ kubectl -n ${NAMESPACE} apply -f pt_inference.yaml
```

您的输出应类似于以下内容：

```
service/densenet-service created
deployment.apps/densenet-service created
```

7. 检查 pod 的状态并等待 pod 处于“RUNNING (正在运行)”状态。

```
$ kubectl get pods -n ${NAMESPACE}
```

您的输出应类似于以下内容：

```
NAME                     READY     STATUS    RESTARTS   AGE
densenet-service-xvw1    1/1       Running   0          3m
```

8. 要进一步描述 pod，您可以运行：

```
$ kubectl describe pod <pod_name> -n ${NAMESPACE}
```

9. 由于此处的 serviceType 为 ClusterIP，您可以将端口从您的容器转发至您的主机（& 将在后台中运行此项）。

```
$ kubectl port-forward -n ${NAMESPACE} `kubectl get pods -n ${NAMESPACE} --
selector=app=densenet-service -o jsonpath='{.items[0].metadata.name}'` 8080:8080 &
```

10. 在启动您的服务器后，现在您可以从不同的窗口来运行推理。

```
$ curl -O https://s3.amazonaws.com/model-server/inputs/flower.jpg
curl -X POST http://127.0.0.1:8080/predictions/densenet -T flower.jpg
```

使用完集群后，请参阅 EKS 清除以获取有关清除集群的信息。

TensorFlow GPU 推理

在此方法中，创建一个 Kubernetes 服务和部署。Kubernetes 服务公开了一个进程及其端口。在创建 Kubernetes 服务时，您可以指定要使用的服务类型。`ServiceTypes`默认 `ServiceType` 为 `ClusterIP`。部署负责确保一定数量的 pod 始终处于启动和运行状态。

1. 对于基于 GPU 的推理，安装适用于 Kubernetes 的 NVIDIA 设备插件：

```
$ kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/
v1.12/nvidia-device-plugin.yml
```

2. 验证运行 nvidia-device-plugin-daemonset是否正常。

```
$ kubectl get daemonset -n kube-system
```

该输出值将类似于以下内容：

```
NAME                              DESIRED   CURRENT   READY     UP-TO-DATE
 AVAILABLE   NODE SELECTOR   AGE
aws-node                          3         3         3         3           3
     <none>          6d
kube-proxy                        3         3         3         3           3
     <none>          6d
nvidia-device-plugin-daemonset    3         3         3         3           3
     <none>          57s
```

3. 创建命名空间。您可能需要更改 kubeconfig 以指向正确的集群。验证您已设置“training-gpu-1”或将其更改为您的 GPU 集群的配置。有关设置集群的更多信息，请参阅亚马逊 EKS 设置。

```
$ NAMESPACE=tf-inference; kubectl –kubeconfig=/home/ubuntu/.kube/eksctl/clusters/
training-gpu-1 create namespace ${NAMESPACE}
```

4. 用于推理的模型可通过不同的方式进行检索，例如，使用共享卷、S3 等。由于该服务需要访问 S3 和 ECR，因此您必须将您的 AWS 证书存储为 Kubernetes 密钥。在本示例中，您将使用 S3 来存储和提取训练后的模型。

   检查您的 AWS 凭证。这些凭证必须具有 S3 写入访问权限。

```
$ cat ~/.aws/credentials
```

5. 输出将类似于以下内容：

```
$ [default]
aws_access_key_id = FAKEAWSACCESSKEYID
aws_secret_access_key = FAKEAWSSECRETACCESSKEY
```

6. 使用 base64 对这些凭证进行编码。首先编码访问密钥。

```
$ echo -n 'FAKEAWSACCESSKEYID' | base64
```

接下来编码秘密访问密钥。

```
$ echo -n 'FAKEAWSSECRETACCESSKEYID' | base64
```

您的输出应类似于以下内容：

```
$ echo -n 'FAKEAWSACCESSKEYID' | base64
RkFLRUFXU0FDQ0VTU0tFWUlE
$ echo -n 'FAKEAWSSECRETACCESSKEY' | base64
RkFLRUFXU1NFQ1JFVEFDQ0VTU0tFWQ==
```

7. 创建 yaml 文件来存储密钥。在您的主目录中将该密钥另存为 secret.yaml。

```
apiVersion: v1
kind: Secret
metadata:
name: aws-s3-secret
type: Opaque
data:
AWS_ACCESS_KEY_ID: RkFLRUFXU0FDQ0VTU0tFWUlE
AWS_SECRET_ACCESS_KEY: RkFLRUFXU1NFQ1JFVEFDQ0VTU0tFWQ==
```

8. 将密钥应用于您的命名空间：

```
$ kubectl -n ${NAMESPACE} apply -f secret.yaml
```

9. 在本示例中，您将克隆 tensorflow-serving 存储库并将预训练模型同步到 S3 存储桶。以下示例命名存储桶 `tensorflow-serving-models`。它还将已保存的模型同步到名 `saved_model_half_plus_two_gpu` 的 S3 存储桶。

```
$ git clone https://github.com/tensorflow/serving/
```

```
$ cd serving/tensorflow_serving/servables/tensorflow/testdata/
```

10. 同步 CPU 模型。

```
$ aws s3 sync saved_model_half_plus_two_gpu s3://<your_s3_bucket>/
saved_model_half_plus_two_gpu
```

11. 创建 `tf_inference.yaml` 文件。使用下一个代码块的内容作为其内容，并将 `--model_base_path` 更新为使用您的 S3 存储桶。你可以将其与 TensorFlow 或 TensorFlow 2 一起使用。要将其与 TensorFlow 2 一起使用，请将 Docker 镜像更改为 TensorFlow 2 镜像。

```
---
kind: Service
apiVersion: v1
metadata:
name: half-plus-two
labels:
  app: half-plus-two
spec:
ports:
- name: http-tf-serving
  port: 8500
  targetPort: 8500
- name: grpc-tf-serving
  port: 9000
  targetPort: 9000
selector:
  app: half-plus-two
  role: master
type: ClusterIP
---
kind: Deployment
apiVersion: apps/v1
metadata:
name: half-plus-two
labels:
  app: half-plus-two
  role: master
spec:
replicas: 1
selector:
  matchLabels:
    app: half-plus-two
```

```
    role: master
template:
  metadata:
    labels:
      app: half-plus-two
      role: master
  spec:
    containers:
    - name: half-plus-two
      image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference:1.15.0-gpu-py36-cu100-ubuntu18.04
      command:
      - /usr/bin/tensorflow_model_server
      args:
      - --port=9000
      - --rest_api_port=8500
      - --model_name=saved_model_half_plus_two_gpu
      - --model_base_path=s3://tensorflow-trained-models/
saved_model_half_plus_two_gpu
      ports:
      - containerPort: 8500
      - containerPort: 9000
      imagePullPolicy: IfNotPresent
      env:
      - name: AWS_ACCESS_KEY_ID
        valueFrom:
          secretKeyRef:
            key: AWS_ACCESS_KEY_ID
            name: aws-s3-secret
      - name: AWS_SECRET_ACCESS_KEY
        valueFrom:
          secretKeyRef:
            key: AWS_SECRET_ACCESS_KEY
            name: aws-s3-secret
      - name: AWS_REGION
        value: us-east-1
      - name: S3_USE_HTTPS
        value: "true"
      - name: S3_VERIFY_SSL
        value: "true"
      - name: S3_ENDPOINT
        value: s3.us-east-1.amazonaws.com
        resources:
          limits:
```

```
            cpu: 4
            memory: 4Gi
            nvidia.com/gpu: 1
          requests:
            cpu: "1"
            memory: 1Gi
```

12. 将配置应用于之前定义的命名空间中的新 pod：

```
$ kubectl -n ${NAMESPACE} apply -f tf_inference.yaml
```

您的输出应类似于以下内容：

```
service/half-plus-two created
deployment.apps/half-plus-two created
```

13. 检查 pod 的状态并等待 pod 处于“RUNNING (正在运行)”状态：

```
$ kubectl get pods -n ${NAMESPACE}
```

14. 重复检查状态步骤，直到您看到以下“RUNNING (正在运行)”状态：

```
NAME                      READY     STATUS    RESTARTS   AGE
half-plus-two-vmwp9   1/1       Running   0          3m
```

15. 要进一步描述 pod，您可以运行：

```
$ kubectl describe pod <pod_name> -n ${NAMESPACE}
```

16. 由于此处的 serviceType 为 ClusterIP，您可以将端口从您的容器转发至您的主机，（& 将在后台中运行此项）：

```
$ kubectl port-forward -n ${NAMESPACE} `kubectl get pods -n ${NAMESPACE} --
selector=app=half-plus-two -o jsonpath='{.items[0].metadata.name}'` 8500:8500 &
```

17. 将以下 json 字符串置于名为 `half_plus_two_input.json` 的文件中

```
{"instances": [1.0, 2.0, 5.0]}
```

18. 在该模型上运行推理：

```
$ curl -d @half_plus_two_input.json -X POST http://localhost:8500/v1/models/
saved_model_half_plus_two_cpu:predict
```

预期的输出如下所示：

```
{
"predictions": [2.5, 3.0, 4.5
]
}
```

后续步骤

要了解如何在 Amazon EKS 上使用带有 Deep Learning Containers 的自定义入口点，请参阅。自定义入口点

自定义入口点

对于某些图像， AWS 容器使用自定义入口点脚本。如果您要使用自己的入口点，可以按如下方式覆盖入口点。

更新 Pod 文件中的 `command` 参数。将 `args` 参数替换为您的自定义入口点脚本。

```
---
apiVersion: v1
kind: Pod
metadata:
  name: pytorch-multi-model-server-densenet
spec:
  restartPolicy: OnFailure
  containers:
  - name: pytorch-multi-model-server-densenet
    image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference:1.2.0-cpu-
py36-ubuntu16.04
    command:
      - "/bin/sh"
      - "-c"
    args:
      - /usr/local/bin/multi-model-server
      - --start
      - --mms-config /home/model-server/config.properties
```

```
      - --models densenet="https://dlc-samples.s3.amazonaws.com/pytorch/multi-model-server/densenet/densenet.mar"
```

command 是 entrypoint 的 Kubernetes 字段名称。有关详细信息，请参阅此 Kubernetes 字段名称表。

如果 EKS 集群具有过期的 IAM 权限访问包含该映像的 ECR 存储库，或者您使用的 kubectl 来自与创建该集群的用户不同的用户，则您将收到以下错误消息。

```
error: unable to recognize "job.yaml": Unauthorized
```

要解决此问题，您需要刷新 IAM 令牌。请运行以下脚本。

```
set -ex

AWS_ACCOUNT=${AWS_ACCOUNT}
AWS_REGION=us-east-1
DOCKER_REGISTRY_SERVER=https://${AWS_ACCOUNT}.dkr.ecr.${AWS_REGION}.amazonaws.com
DOCKER_USER=AWS
DOCKER_PASSWORD=`aws ecr get-login --region ${AWS_REGION} --registry-ids ${AWS_ACCOUNT}
 | cut -d' ' -f6`
kubectl delete secret aws-registry || true
kubectl create secret docker-registry aws-registry \
--docker-server=$DOCKER_REGISTRY_SERVER \
--docker-username=$DOCKER_USER \
--docker-password=$DOCKER_PASSWORD
kubectl patch serviceaccount default -p '{"imagePullSecrets":[{"name":"aws-
registry"}]}'
```

将 spec 下的以下内容附加到您的 Pod 文件中。

```
imagePullSecrets:
    - name: aws-registry
```

对 EKS 上的 AWS 深度学习容器进行故障排除

以下是在 Amazon EKS 集群上使用 Dee AWS p Learning Containers 时，命令行中可能返回的常见错误。每个错误的后面都提供了错误的解决方案。

故障排除

主题

- 设置错误
- 使用错误
- 清理错误

设置错误

在您的 Amazon EKS 集群上设置 Deep Learning Containers 时，可能会返回以下错误。

- 错误：注册表**kubeflow**不存在

```
$ ks pkg install kubeflow/tf-serving
  ERROR registry 'kubeflow' does not exist
```

要解决此错误，请运行以下命令。

```
ks registry add kubeflow github.com/google/kubefl ow/tree/master/kubeflow
```

- 错误：已超过上下文截止日期

```
$ eksctl create cluster <args>
  [#] waiting for CloudFormation stack "eksctl-training-cluster-1-nodegroup-
ng-8c4c94bc" to reach "CREATE_COMPLETE" status: RequestCanceled: waiter context
 canceled
  caused by: context deadline exceeded
```

要解决此错误，请确认您的账户没有超出容量。您也可以尝试在其他区域创建集群。

- 错误：与服务器 localhost: 8080 的连接被拒绝

```
$ kubectl get nodes
  The connection to the server localhost:8080 was refused - did you specify the right
 host or port?
```

要解决此错误，请运行以下命令将集群复制到 Kubernetes 配置。

```
cp ~/.kube/eksctl/clusters/<cluster-name> ~/.kube/config
```

- 错误：handle 对象：正在从群集中修补对象：将对象与现有状态合并：未授权

```
$ ks apply default
  ERROR handle object: patching object from cluster: merging object with existing
 state: Unauthorized
```

  此错误是由于具有不同授权或凭据的多个用户尝试在同一个集群上启动作业时可能出现并发问题。确认您正在正确的集群上启动作业。

- 错误：无法创建应用程序；目录 “/home/ubuntu/kubeflow-tf-hvd” 已存在

```
$ APP_NAME=kubeflow-tf-hvd; ks init ${APP_NAME}; cd ${APP_NAME}
  INFO Using context "arn:aws:eks:eu-west-1:999999999999:cluster/training-gpu-1" from
 kubeconfig file "/home/ubuntu/.kube/config"
  ERROR Could not create app; directory '/home/ubuntu/kubeflow-tf-hvd' already exists
```

  您可以放心地忽略这一警告。但是，您可能需要在该文件夹中进行其他清理。要简化清理工作，请删除该文件夹。

使用错误

```
ssh: Could not resolve hostname openmpi-worker-1.openmpi.kubeflow-dist-train-tf: Name
 or service not known
```

如果您在使用 Amazon EKS 集群时看到此错误消息，请再次运行 NVIDIA 设备插件安装步骤。通过传入特定的配置文件或将您的活动集群切换到目标集群，验证您的目标群集是否正确。

清理错误

清理 Amazon EKS 集群的资源时，可能会返回以下错误。

- 错误：服务器没有资源类型 “*namspace*”

```
$ kubectl delete namespace ${NAMESPACE}
  error: the server doesn't have a resource type "namspace"
```

  验证命名空间的拼写是否正确。

- 错误：服务器已要求客户端提供凭据

```
$ ks delete default
  ERROR the server has asked for the client to provide credentials
```

要解决此错误，请使用`aws configure`或导出 AWS 环境变量验证是否`~/.kube/config`指向正确的集群，以及 AWS 凭据是否已正确配置。

- 错误：从起始路径中查找应用程序根目录：：找不到 ksonnet 项目

```
$ ks delete default
  ERROR finding app root from starting path: : unable to find ksonnet project
```

要解决此错误，请确认您位于 ksonnet 应用程序创建的目录中。`ks init`这是运行所在的文件夹。

- 错误：来自服务器 (NotFound) 的错误：找不到 pod “openmpi-master”

```
$ kubectl logs -n ${NAMESPACE} -f ${COMPONENT}-master > results/benchmark_1.out
  Error from server (NotFound): pods "openmpi-master" not found
```

此错误可能是由于删除上下文后尝试访问资源所致。删除默认上下文也会导致相应的资源也被删除。

发布说明

查看为特定机器学习框架、基础架构和 AWS 服务构建的 Dee AWS p Learning Containers 的最新版本说明。

> **Tip**
> 有关可用深度学习容器的完整列表以及如何提取这些容器的信息，请参阅可用的深度学习容器映像。

Deep Learning Containers

- 基础深度学习 Containers
- vILM 深度学习容器 Deep Learning
- PyTorch 深度学习 Containers
- TensorFlow 深度学习 Containers

基础深度学习 Containers

以下是 Base Deep Learning Containers 的发行说明：

版本	类型	服务	架构	发行说明
12.8	CUDA 和 EFA	EC2 ECS EKS	X86	AWS 适用于 CUDA 12.8 的深度学习容器基础：2025 年 6 月 5 日

AWS 适用于 EC2、ECS、EKS 的深度学习基础容器（配备 NVIDIA CUDA 12.8 和 AWS EFA）

AWS Dee@@ p Learning Containers (DLCs) 现在支持 Ubuntu 24.04 的基础映像 EC2，这些镜像是在 ECS 和 EKS 上构建机器学习环境的基础层。

这些 Base DLCs 打包了基本的深度学习组件和依赖项，而不受特定框架的约束，使用户可以灵活地使用自己的首选框架对其 DLCs 进行自定义。

这些映像预先配置了 CUDA、cuDNN、Python 和 EFA 支持的核心组件，可以无缝运行 out-of-the-box，通过节点间连接提供了稳定、可靠的起点，同时保持了 ECS 和 EKS 服务的兼容性。 EC2

对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。

可用容器列表可在上找到GitHub。使用我们开发者指南中的入门部分，快速开始使用 Dee AWS p Learning Containers。为确保您使用的是最新版本的 DLC，我们邀请您订阅我们的 DLC 通知机制。如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

发布说明

- 开发工具：包括 curl、build-essential、cmake 和 git，以满足常见的开发需求
- Python 环境：预装了 AWS CLI、boto3 和请求的 Python 3.12
- GPU 支持：带有 cuda-compat 的 CUDA 12.8.1 向后兼容
- 神经网络库：用于深度神经网络操作的 cuDNN 9.8.0.87
- 分布式训练：适用于多 GPU 和多节点通信的 NCCL 2.26.2-1
- 网络性能：适用于低延迟网络通信的 EFA 1.40.0

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 支持

支持 Python 3.12。

GPU 实例类型支持

这些容器支持 Graviton GPU 实例类型 g5G，并包含以下支持 GPU 的软件组件：

- CUDA 12.8
- cudnn 9.8.0.87
- NCCL 2.26.2-1

示例 URL

```
763104351884.dkr.ecr.us-west-2.amazonaws.com/base:12.8.1-gpu-py312-cu128-ubuntu24.04-
ec2
```

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（弗吉尼亚州北部）	us-east-1
美国东部（俄亥俄州）	us-east-2
美国西部 (加利福尼亚北部)	us-west-1
美国西部（俄勒冈州）	us-west-2
亚太地区（香港）	ap-east-1
亚太地区（孟买）	ap-south-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（东京）	ap-northeast-1
亚太地区 (首尔)	ap-northeast-2
亚太地区（大阪）	ap-northeast-3
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（雅加达）	ap-southeast-3
亚太地区（墨尔本）	ap-southeast-4
亚太地区（马来西亚）	ap-southeast-5

区域	代码
亚太地区（泰国）	ap-southeast-7
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（法兰克福）	eu-central-1
欧洲（苏黎世）	eu-central-2
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（米兰）	eu-south-1
欧盟（西班牙）	eu-south-2
欧洲（斯德哥尔摩）	eu-north-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
以色列（特拉维夫）	il-central-1
南非（圣保罗）	sa-east-1
AF South（开普敦）	af-south-1
墨西哥（中部）	mx-central-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试在：p4d.24xlarge、p4de.24xlarge、p5.48xlarge
- 经过测试：openclip、nccl-tests

已知问题

- 到目前为止尚无已知问题

vlLM 深度学习容器 Deep Learning

以下是 vlLM Deep Learning Containers 的发行说明：

版本	类型	服务	架构	发行说明
最新	常规	EC2	X86	AWS 适用于 vlLM 的 Deep Learning Containers on EC2、ECS 和 EKS

AWS 支持 EFA 的 vlLM 的 Deep Learning Containers

AWS D@@ eep Learning Containers (DLCs) 现在支持针对大型语言模型服务进行优化的 vlLM 图像。

vLLM DLC 提供了一个可用于部署和服务的生产就绪环境，内置对 EFA（ LLMs 弹性结构适配器）的支持。这款专用容器预先配置了 vLLM 的高级功能和优化，为从单节点到多节点部署的各种用例提供高性能、可扩展和高效的 LLM 服务提供了理想的起点。有关如何使用 vLLM 的指南，请查看 vLLM 文档。

使用我们开发者指南中的入门部分，快速开始使用 Dee AWS p Learning Containers。

如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

为确保您使用的是最新版本的 DLC，我们邀请您订阅我们的 DLC 通知机制。

更改日志

要了解 vILM DLC 的最新变化，请查看更新日志。

可用容器列表可在上找到GitHub。

安全建议

对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。

AWS 建议客户监控安全公告中的关键AWS 安全更新。

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（弗吉尼亚州北部）	us-east-1
美国东部（俄亥俄州）	us-east-2
美国西部 (加利福尼亚北部)	us-west-1
美国西部（俄勒冈州）	us-west-2
亚太地区（香港）	ap-east-1
亚太地区（孟买）	ap-south-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（东京）	ap-northeast-1
亚太地区 (首尔)	ap-northeast-2
亚太地区（大阪）	ap-northeast-3
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（雅加达）	ap-southeast-3

区域	代码
亚太地区（墨尔本）	ap-southeast-4
亚太地区（马来西亚）	ap-southeast-5
亚太地区（泰国）	ap-southeast-7
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（法兰克福）	eu-central-1
欧洲（苏黎世）	eu-central-2
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（米兰）	eu-south-1
欧盟（西班牙）	eu-south-2
欧洲（斯德哥尔摩）	eu-north-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
以色列（特拉维夫）	il-central-1
南非（圣保罗）	sa-east-1
AF South（开普敦）	af-south-1
墨西哥（中部）	mx-central-1
中国（北京）	cn-north-1

区域	代码
中国（宁夏）	cn-northwest-1

PyTorch 深度学习 Containers

以下是 Dee PyTorch p Learning Containers 的发行说明：

版本	类型	服务	架构	发行说明
2.7	训练	SageMaker	X86	AWS 适用于 PyTorch 2.7 的 Deep Learning Container s（ SageMaker 人工智能训练）：2025 年 6 月 13 日
2.7	训练	EC2 ECS EKS	X86	AWS 适用于 PyTorch 2.7 的 Deep Learning Containers（在 EC2、ECS 和 EKS 上训练）：2025 年 6 月 13 日
2.7	训练	EC2	Arm64	AWS 适用于 PyTorch 2.7 的 Deep Learning Containers ARM64 （培训开启 EC2）：2025 年 6 月 4 日

版本	类型	服务	架构	发行说明
2.6	训练	EC2 ECS EKS	X86	AWS 适用于 PyTorch 2.6 的 Deep Learning Conta EC2 iners（在 ECS 和 EKS 上训练）：2025 年 3 月 12 日
2.6	训练	SageMaker	X86	AWS PyTorch 2.6 版 Deep Learning Container s（ SageMaker 人工智能训练）：2025 年 3 月 12 日
2.6	推理	EC2 ECS EKS	X86	AWS PyTorch 2.6 版 Deep Learning Containers（在 ECS 和 EKS 上 EC2进行推理）：2025 年 2 月 27 日
2.6	推理	SageMaker	X86	AWS PyTorch 2.6 版 Deep Learning Container s（ SageMaker 人工智能推断）：2025 年 2 月 27 日

版本	类型	服务	架构	发行说明
2.6	推理	EC2 ECS EKS	Arm64	AWS PyTorch 2.6 版 Deep Learning Containers ARM64（在 ECS 和 EKS 上 EC2进行推理）：2025 年 2 月 24 日
2.6	推理	SageMaker	Arm64	AWS PyTorch 2.6 版 Deep Learning Container s ARM64（ SageMaker 人工智能推断）：2024 年 2 月 24 日
2.5	训练	EC2 ECS EKS	X86	AWS 适用于 PyTorch 2.5 的 Deep Learning Conta EC2 iners（在 ECS 和 EKS 上训练）：2024 年 11 月 25 日

版本	类型	服务	架构	发行说明
2.5	训练	SageMaker	X86	AWS PyTorch 2.5 版 Deep Learning Containers（ SageMaker 人工智能训练）：2024 年 11 月 25 日
2.5	推理	EC2 ECS EKS	X86	AWS PyTorch 2.5 版 Deep Learning Containers（在 ECS 和 EKS 上 EC2进行推理）：2024 年 11 月 13 日
2.5	推理	SageMaker	X86	AWS PyTorch 2.5 版 Deep Learning Containers（ SageMaker 人工智能推断）：2024 年 11 月 13 日

版本	类型	服务	架构	发行说明
2.5	推理	EC2 ECS EKS	Arm64	AWS PyTorch 2.5 版 Deep Learning Containers ARM64（在 ECS 和 EKS 上 EC2进行推理）：2024 年 12 月 11 日
2.5	推理	SageMaker	Arm64	AWS PyTorch 2.5 版 Deep Learning Containers ARM64（SageMaker 人工智能推断）：2024 年 12 月 11 日
2.4	训练	EC2 ECS EKS	X86	AWS 适用于 PyTorch 2.4 的 Deep Learning Containers（在 EC2、ECS 和 EKS 上训练）：2024 年 9 月 26 日

版本	类型	服务	架构	发行说明
2.4	训练	SageMaker	X86	AWS 适用于 PyTorch 2.4 的 Deep Learning Containers（ SageMaker 人工智能训练）：2024 年 9 月 26 日
2.4	推理	EC2 ECS EKS	X86	AWS 适用于 PyTorch 2.4 的 Deep Learning Containers（在 ECS 和 EKS 上 EC2进行推理）：2024 年 9 月 26 日
2.4	推理	SageMaker	X86	AWS PyTorch 2.4 版 Deep Learning Containers（ SageMaker 人工智能推断）：2024 年 10 月 3 日

版本	类型	服务	架构	发行说明
2.4	推理	EC2 ECS EKS	Arm64	AWS 适用于 PyTorch 2.4 Graviton 的 Deep Learning Containers（在 ECS 和 EKS 上进行 EC2推理）：2024 年 7 月 11 日
2.4	推理	SageMaker	Arm64	AWS 适用于 PyTorch 2.4 Graviton 的 Deep Learning Container s（ SageMaker 人工智能推断）：2024 年 7 月 11 日

AWS Deep Learning Conta PyTorch iners 适用于 2.7 SageMaker

AWS 适用于亚马逊 SageMaker 的 Dee@@ p Learning Containers (DLCs) 现已推出 PyTorch 2.7 版本，在 Ubuntu 22.04 上支持 CUDA 12.8。你可以在任何 SageMaker 服务上启动新版本的 Deep Learning Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已经过 SageMaker 服务测试，提供稳定版本的 NVIDIA CUDA、Intel MKL 和其他组件，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 服务上使用而设计。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为训练引入了 PyTorch 2.7 版容器，该容器支持 SageMaker。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 请参考此处的 PyTorch 2.7.1 官方发行说明。
- 添加了 Python 3.12 支持
- 添加了 PyTorch 域库：
  - torchtnt 0.2.4
  - torchdata 0.11.0
  - torchaudio 2.7.1
  - torchvision 0.22.1
- 添加了 CUDA 12.8 支持
- 添加了 Ubuntu 22.04 支持
- GPU Docker 镜像包含以下库：
  - CUDA 12.8.0
  - cudnn 9.7.1.26
  - NCCL 2.26.2
  - EFA 安装程序 1.40.0（嵌入了 AWS OFI NCCL）
  - 变压器引擎 2.3
  - Flash 注意 2.7.4.post1
  - GDRCopy 2.5
- 添加了 fastai 2.8.2 支持
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch 训练容器支持 Python 3.12。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.8
- cudnn 9.7.1.26
- NCCL 2.26.2

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4

区域	代码
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
亚太地区（泰国）	ap-southeast-7
墨西哥（中部）	mx-central-1
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1

区域	代码
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试在：p4d.24xlarge、p4de.24xlarge、p5.48xlarge、g4dn.4xlarge、g5.12xlarge、g5.12xlarge
- 在 SageMaker MNIST 开启的情况下进行了测试。

AWS 适用于 PyTorch 2.7 训练的 Deep Learning Containers EC2、ECS 和 EK

AWS 适用于亚马逊 EC2、ECS 和 EKS 的 Dee@@ p Learning Containers (DLCs) 现已推出 PyTorch 2.7 版本，并在 Ubuntu 22.04 上支持 CUDA 12.8。你可以在任何 ECS 和 EKS 服务上启动新版本的 Deep Le EC2 arning Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 映像已通过 EC2、ECS 和 EKS 服务进行了测试，提供了稳定版本的 NVIDIA CUDA、Intel MKL 和其他组件，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 ECS 和 EKS 服务上使用而设计。 EC2

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为训练引入了 PyTorch 2.7 版容器 EC2，它支持 ECS 和 EKS。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 请参考此处的 PyTorch 2.7.1 官方发行说明。
- 添加了 Python 3.12 支持

- 添加了 PyTorch 域库：
  - torchtnt 0.2.4
  - torchdata 0.11.0
  - torchaudio 2.7.1
  - torchvision 0.22.1
- 添加了 CUDA 12.8 支持
- 添加了 Ubuntu 22.04 支持
- GPU Docker 镜像包含以下库：
  - CUDA 12.8.0
  - cudnn 9.7.1.26
  - NCCL 2.26.2
  - EFA 安装程序 1.40.0（嵌入了 AWS OFI NCCL）
  - 变压器引擎 2.3
  - Flash 注意 2.7.4.post1
  - GDRCopy 2.5
- 添加了 fastai 2.8.2 支持
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch 训练容器支持 Python 3.12。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.8
- cudnn 9.7.1.26
- NCCL 2.26.2

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1

区域	代码
亚太地区（马来西亚）	ap-southeast-5
亚太地区（泰国）	ap-southeast-7
墨西哥（中部）	mx-central-1
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试在：p4d.24xlarge、p4de.24xlarge、p5.48xlarge、g4dn.4xlarge、g5.24xlarge、g5.12xlarge、g5.12xlarge、g5
- 使用 Resnet50、BERT 以及上的 ImageNet 数据集、ECS AMI（亚马逊 Linu EC2 x AMI 2.0.20250605）和 EKS AMI（-1.32.3-20250610）进行了测试 amazon-eks-gpu-node

AWS Deep Learning Conta PyTorch iner ARM64 s 适用于 2.7 EC2

AWS 适用于亚马逊 EC2 的 Dee@@ p Learning Containers (DLCs) 现已适用于 ARM64 平台，包括 AWS Graviton 实例类型，在 Ubuntu 22.04 上支持 PyTorch 2.7 和 CUDA 12.8。

此版本包括用于在 GPU 上训练的容器镜像，该镜像针对性能和扩展进行了优化 AWS EC2。该镜像提供了 NVIDIA CUDA、cuDNN、NCCL 和其他组件的稳定版本。扫描此镜像中的所有软件组件是否存在安全漏洞，并根据 AWS 安全最佳实践进行更新或修补。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 引入了 PyTorch 2.7 版的容器，用于训练 EC2。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 此图像应与 g5G 实例类型一起使用，后者由 G raviton CPUs 和 NVIDIA T4G Tensor Core 提供支持。 GPUs
- 请参考此处的 PyTorch 2.7.0 官方发行说明。
- 此镜像包括以下库：
  - CUDA 12.8.0
  - cudnn 9.8.0.87
  - NCCL 2.26.2
  - EFA 安装程序 1.40.0（嵌入了 AWS OFI NCCL）
  - 变压器引擎 2.0
  - 闪电注意 2.7.3
  - GDRCopy 2.5

- 请注意，由于缺乏硬件支持，EFA、Transformer Engine、Flash Attention 和 GDRCopy 尚未经过测试。
- 可以在这里找到 Dockerfile。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch ARM64 训练容器支持 Python 3.12。

GPU 实例类型支持

这些容器支持 Graviton GPU 实例类型 g5G，并包含以下支持 GPU 的软件组件：

- CUDA 12.8
- cudnn 9.8.0.87+cuda12.8
- NCCL 2.26.2+cuda12.8

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1

区域	代码
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
亚太地区（泰国）	ap-southeast-7
墨西哥（中部）	mx-central-1
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2

区域	代码
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.12xlarge
- 测试在：c8g.4xlarge、t4g.2xlarge、r8g.2xlarge、m7g.4xlarge、g5g.16xlarge、g5g.16xlarge

已知问题

- ARM64/aarch64 还没有官方的 Triton 发行版，因此一些 torch.compile 工作负载会失败，原因是：

```
torch._dynamo.exc.BackendCompilerFailed: backend='inductor' raised:
RuntimeError: Cannot find a working triton installation. More information on
 installing Triton can be found at https://github.com/openai/triton
```

- 参见GitHub 问题：将 device_id 传递给 torch.distributed.init_process_group () 会导致 NCCL 在通信期间随机挂起。

AWS 适用于 PyTorch 2.6 训练的 Deep Learning Containers EC2、ECS 和 EK

AWS 适用于亚马逊弹性计算云 (DLCs)、亚马逊弹性@@ 容器服务 (ECSEC2) 和亚马逊弹性 Kubernetes Service (EKS) 的 Deep Learning Containers () 现已推出 2.6 版本 PyTorch ，并在 Ubuntu 22.04 上支持 CUDA 12.6。你可以在任何 ECS 和 EKS 服务上启动新版本的 Deep Learn EC2 ing Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 映像已通过 ECS 和 EKS 服务进行了测试，并提供了 NVIDIA CUDA、Intel MKL 和其他组件的稳定版本，为运行深度学习工作负载提供了优化的用户体验。 EC2 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何 ECS 和 EKS 服务上使用而设计。 EC2如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为 PyTorch 2.6.0 引入了用于训练的容器 EC2，该容器支持 ECS 和 EKS。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.6 开始，我们将从 PyPI 中删除 Conda DLCs 并安装所有 Python 软件包。
- PyTorch 2.6 在以下方面进行了多项改进 PT2：torch.compile 现在可以与 Python 3.13 一起使用；新的与性能相关的旋钮 torch.compiler.set_stance；多项增强。 AOTInductor 除了 PT2 改进之外，另一个亮点是对 X86 CPUs 的 FP16 支持。
- 请参阅此处的 PyTorch 2.6.0 官方发行说明。
- 已移除 fastai，因为它尚未发布 PyTorch 2.6 兼容版本，请参阅此问题。
- 添加了 Python 3.12 支持
- 添加了 CUDA 12.6 支持
- 添加了 Ubuntu 22.04 支持
- GPU Docker 镜像包含以下库：
  - CUDA 12.6.3
  - cudnn 9.7.0.66
  - NCCL 2.23.4

- EFA 安装程序 1.38.0（嵌入了 AWS OFI NCCL）
- 变压器引擎 2.0
- 闪电注意 2.7.3
- GDRCopy 2.5
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch 训练容器支持 Python 3.12。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.6.3
- cudnn 9.7.0.66+cuda12.6
- NCCL 2.23.4+cuda12.6

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2

区域	代码
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3

区域	代码
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 已在：g3.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4d.24xlarge、p4de.24xlarge、g4dn.xlarge、
- 使用 Resnet50、BERT 以及上的 ImageNet 数据集、ECS AMI（亚马逊 Linu EC2 x AMI 2.0.20250201）和 EKS AMI（-1.32.1-20250212）进行了测试 amazon-eks-gpu-node

已知问题

- 使用的客户TransformerEngine可能会遇到 [W init.cpp: 767] 警告：torch 脚本不再支持 nvfuser，使用 _jit_set_nvfuser_enabled 已被弃用，并且由于自 2.2 起已弃用，因此禁止操作（函数运算符()）。NVFuser PyTorch 欲了解更多信息，请查看此问题。

AWS Deep Learning Conta PyTorch iners 适用于 2.6 SageMaker

AWS 适用于亚马逊（SMDLCs）的 Dee@@ p Learning Containers SageMaker () 现已在 Ubuntu PyTorch 22.04 上推出 2.6，支持 CUDA 12.6。你可以在任何 SM 服务上启动 Deep Learning

Containers 的新版本。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已通过 SM 服务进行了测试，提供了稳定版本的 NVIDIA CUDA、Intel MKL 和其他组件，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 服务上使用而设计。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 引入了 PyTorch 2.6.0 的容器，用于支持 SageMaker 服务的训练。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.6 开始，我们将从 PyPI 中删除 Conda DLCs 并安装所有 Python 软件包。
- PyTorch 2.6 在以下方面进行了多项改进 PT2：torch.compile 现在可以与 Python 3.13 一起使用；新的与性能相关的旋钮 torch.compiler.set_stance；多项增强。 AOTInductor 除了 PT2 改进之外，另一个亮点是对 X86 CPUs 的 FP16 支持。
- 请参考此处的 PyTorch 2.6.0 官方发行说明。
- 已移除 fastai，因为它尚未发布 PyTorch 2.6 兼容版本，请参阅此问题。
- 添加了 Python 3.12 支持
- 添加了 CUDA 12.6 支持
- 添加了 Ubuntu 22.04 支持
- GPU Docker 镜像包含以下库：
  - CUDA 12.6.3
  - cudnn 9.7.0.66
  - NCCL 2.23.4
  - EFA 安装程序 1.38.0（嵌入了 AWS OFI NCCL）
  - 变压器引擎 2.0
  - 闪电注意 2.7.3
  - GDRCopy 2.5
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch 训练和推理容器支持 Python 3.12。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.6.3
- cudnn 9.7.0.66+cuda12.6
- NCCL 2.23.4+cuda12.6

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2

区域	代码
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1

区域	代码
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 已在：g3.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4d.24xlarge、p4de.24xlarge、g4dn.xlarge、
- 在 SageMaker MNIST 开启的情况下进行了测试。

已知问题

- 使用的客户TransformerEngine可能会遇到 [W init.cpp: 767] 警告：torch 脚本中不再支持 nvfuser，使用 _jit_set_nvfuser_enabled 已被弃用，并且由于自 2.2 起已弃用，因此禁止操作（函数运算符()）。NVFuser PyTorch 欲了解更多信息，请查看此问题。

AWS 适用于 PyTorch 2.6 推理的 Deep Learning Container EC2 s、ECS 和 EKS

AWS 适用于亚马逊弹性计算云 (DLCs)、亚马逊弹性@@ 容器服务 (ECSEC2) 和亚马逊弹性 Kubernetes Service (EKS) 的 Deep Learning Containers () 现已推出 2.6 版本 PyTorch ，并在 Ubuntu 22.04 上支持 CUDA 12.4。你可以在任何 ECS 和 EKS 服务上启动新版本的 Deep Learn EC2 ing Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 CPU 和 GPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这些 Docker 镜像已通过 ECS 和 EKS 服务进行了测试，并提供了 NVIDIA CUDA、cuDNN、Intel MKL 和其他组件的稳定版本，为运行深度学习工作负载提供了优化的用户体验。 EC2 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何

ECS 和 EKS 服务上使用而设计。 EC2如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为 PyTorch 2.6.0 引入了用于推理支持 EC2、ECS 和 EKS 的容器。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.6 开始，我们将从 PyPI 中删除 Conda DLCs 并安装所有 Python 软件包。
- PyTorch 2.6 在以下方面进行了多项改进 PT2：torch.compile 现在可以与 Python 3.13 一起使用；新的与性能相关的旋钮 torch.compiler.set_stance；多项增强。 AOTInductor 除了 PT2 改进之外，另一个亮点是对 X86 CPUs 的 FP16 支持。
- 请参考此处的 PyTorch 2.6.0 官方发行说明。
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch 推理容器支持 Python 3.12。

CPU 实例类型支持

这些容器支持 x86_64 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.23.4+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1

区域	代码
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试基于：c5.18xlarge、g3.16xlarge、m5.16xlarge、t3.2xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.
- 在 MNIST 和 Resn et50/ ImageNet 数据集、 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20250201）和 EKS AMI（-1.32.1-20250212）上进行了测试 amazon-eks-gpu-node

AWS Deep Learning Contain PyTorch ers 适用于 2.6 推理 SageMaker

AWS 适用于亚马逊 SageMaker 的 Dee@@ p Learning Containers (DLC) 现已在 Ubuntu PyTorch 22.04 上推出，支持 2.6，支持 CUDA 12.4。你可以在 SageMaker 服务上启动新版本的 Deep Learning Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

此版本包括用于在 CPU 和 GPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这些 Docker 镜像已经过 SageMaker 服务测试，提供稳定版本的 NVIDIA CUDA、cuDNN 和其他组件，为运行深度学习工作负载提供优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 推理服务上使用而设计。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为 PyTorch 2.6.0 引入了用于推理支持服务的容器。 SageMaker 有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.6 开始，我们将从 PyPI 中删除 Conda DLCs 并安装所有 Python 软件包。
- PyTorch 2.6 在以下方面进行了多项改进 PT2：torch.compile 现在可以与 Python 3.13 一起使用；新的与性能相关的旋钮 torch.compiler.set_stance；多项增强。 AOTInductor 除了 PT2 改进之外，另一个亮点是对 X86 CPUs 的 FP16 支持。
- 请参考此处的 PyTorch 2.6.0 官方发行说明。
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch 推理容器支持 Python 3.12。

CPU 实例类型支持

这些容器支持 x86_64 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.23.4+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4

区域	代码
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试基于：c5.18xlarge、g3.16xlarge、m5.16xlarge、t3.2xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.
- 在 SageMakerMNIST 开启的情况下进行了测试。

AWS Deep Learning C PyTorch on ARM64 tainers 适用于 2.6 推理 EC2、ECS 和 EKS

AWS 适用于亚马逊弹性 Kubernetes Service (EKS)、亚马逊弹性计算云 () 和亚马逊弹性@@ 容器服务 EC2 (ECS) 的深度学习容器 (DLC) 现已 ARM64 适用于平台，包括 Graviton 实例类型 AWS ，支持 2.6。 PyTorch

此版本包括用于在 CPU 和 GPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。CPU 映像已通过每个 EC2、ECS 和 EKS 服务进行了测试，而 GPU 映像仅支持 EC2 （参见下表）。GPU 映像提供 NVIDIA CUDA、cuDNN、NCCL 和其他组件的稳定版本。对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。

	EC2	ECS	EKS
CPU 镜像	支持	支持	支持
GPU 镜像	支持	不支持	不支持

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 在 PyTorch 2.6.0 中引入了用于推理支持的容器 EC2、ECS 和实例上的 EKS。 ARM64 有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.6 开始，我们将从 PyPI 中删除 Conda DLCs 并安装所有 Python 软件包。
- TorchServe 版本：0.12.0

- GPU 映像应与 g5G 实例类型一起使用，后者由 G raviton CPUs 和 NVIDIA T4G Tensor Core 提供支持。 GPUs
- 有关框架更新的完整说明，请参阅此处的 PyTorch 2.6.0 官方发行说明。

性能改进

它们 DLCs 继续在Graviton CPU上为BERT和Ro BERTa 情绪分析以及填充掩模模型提供最佳性能，这使Graviton3成为这些模型在云端最具成本效益的CPU平台。 AWS 欲了解更多信息，请参阅 Graviton PyTorch 用户指南。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch ARM64 推理容器支持 Python 3.12。

CPU 实例类型支持

CPU 容器支持上述每项服务所支持的 Graviton CPU 实例类型。

GPU 实例类型支持

GPU 容器支持 Graviton GPU 实例类型 g5G，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.21.5+cuda12.4

发布说明

- 在 Graviton 实例上引入了 PyTorch 2.4 版本的容器以支持推理 EC2、ECS 和 EKS。有关此版本的详细信息，请查看我们的 GitHub 发布标签：适用于 CPU 和 GPU。
- TorchServe 版本：0.11.1
- 11/01/24：更新 TorchServe 至 0.12.0（发布标签：适用于 CPU 和 GPU）
- GPU 镜像是有史以来第一个支持 Graviton (ARM64) + GPU 平台的 DLC。它应该与 g5G 实例类型一起使用，后者由 G raviton CPUs 和 NVIDIA T4G Tensor Core 提供支持。 GPUs

- 有关框架更新，请参阅此处的 PyTorch 2.4 官方发行说明。

性能改进

它们 DLCs 继续在Graviton CPU上为BERT和Ro BERTa 情绪分析以及填充掩模模型提供最佳性能，这使Graviton3成为这些模型在云端最具成本效益的CPU平台。 AWS 欲了解更多信息，请参阅 Graviton PyTorch 用户指南。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch Graviton 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持上述每项服务所支持的 Graviton CPU 实例类型。

GPU 实例类型支持

这些容器支持 Graviton GPU 实例类型 g5G，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.0
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.20.5+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1

区域	代码
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2

区域	代码
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- 经过测试：c8g.4xlarge、t4g.2xlarge、r8g.2xlarge、m7g.4xlarge、g5g.4xlarge、g5g.4xlarge

已知问题

- ARM64/aarch64 还没有官方的 Triton 发行版，因此一些 torch.compile 工作负载会失败，原因是：

```
torch._dynamo.exc.BackendCompilerFailed: backend='inductor' raised:
RuntimeError: Cannot find a working triton installation. More information on
 installing Triton can be found at https://github.com/openai/triton
```

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

AWS Deep Learning Con ARM64 tain PyTorch ers 适用于 2.6 推理 SageMaker

AWS 适用于亚马逊 SageMaker 的 D@@ eep Learning Containers (DLCs) 现已适用于 ARM64 平台，包括支持 2.6 的 AWS Graviton 实例类型。 PyTorch 你可以在上启动新版本的 DLC。 SageMaker

此版本包括一个用于在 CPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这个 Docker 镜像已在上面进行了测试。 SageMaker它为在上运行深度学习工作负载提供了优化的用户体验 SageMaker。扫描此镜像中的所有软件组件是否存在安全漏洞，并根据 AWS 安全最佳实践进行更新或修补。

可用容器列表可以在我们的文档中找到。请参阅 SageMaker Graviton 博客和 DLC 开发者指南，将深度学习工作负载迁移到 Graviton 实例。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 引入了 PyTorch 2.6.0 的容器，用于在实例上提供推理支持 SageMaker 服务。 ARM64 有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.6 开始，我们将从 PyPI 中删除 Conda DLCs 并安装所有 Python 软件包。
- TorchServe 版本：0.12.0
- 有关框架更新的完整说明，请参阅此处的 PyTorch 2.6.0 官方发行说明。

性能改进

它们 DLCs 继续在Graviton上为BERT和Ro BERTa 情绪分析以及填充掩模模型提供最佳性能，使Graviton3成为这些模型在云端最具成本效益的CPU平台。 AWS 欲了解更多信息，请参阅 Graviton PyTorch 用户指南。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.12 Support

PyTorch ARM64 推理容器支持 Python 3.12。

CPU 实例类型支持

这些容器支持下支持的 Graviton CPU 实例类型。 SageMaker

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1

区域	代码
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- 经过测试：c8g.4xlarge、t4g.2xlarge、r8g.2xlarge、m7g.4xlarge、g5g.4xlarge、g5g.4xlarge

已知问题

- 无

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

AWS Deep Learning Containers 用于在 EC2 ECS 和 EKS 上进行 PyTorch 2.5

AWS 适用于亚马逊弹性计算云 (DLCs)、亚马逊弹性@@ 容器服务 (ECSEC2) 和亚马逊弹性 Kubernetes Service (EKS) 的 Deep Learning Containers () 现已推出 2.5 版本 PyTorch ，并在 Ubuntu 22.04 上支持 CUDA 12.4。你可以在任何 ECS 和 EKS 服务上启动新版本的 Deep Learn EC2 ing Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已通过 ECS 和 EKS 服务进行了测试，并提供了 NVIDIA CUDA、Intel MKL 和其他组件的稳定版本，为运行深度学习工作负载提供了优化的用户体验。 EC2 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何 ECS 和 EKS 服务上使用而设计。 EC2如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- PyTorch 为 2.5.1 引入了用于训练的容器 EC2，该容器支持 ECS 和 EKS。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- PyTorch 2.5 为 SDPA 提供了全新 cuDNN 后端，默认情况下，在 H100 或更高版本上使用 SDPA 的用户启用加速功能。 GPUs此外，torch.compile 的区域编译允许用户无需重新编译即可编译重复的 nn.Module（例如 LLM 中的变压器层），从而缩短 torch.compile 的冷启动时间。最后，TorchInductor CPP 后端提供了稳定的性能加速，并提供了许多增强功能，例如 FP16 支持、CPP 封装、AOT-Inductor 模式和最大自动调谐模式。
- 包括修复 PyPI 中的轮子无法在基于 RPM 的 Linux 发行版 out-of-the-box上使用，如 2.5.1 中所述。PyTorch
- 有关更新的完整说明，请参阅此处的 PyTorch 2.5.0 官方发行说明和此处的 PyTorch 2.5.1 发行说明。
- NVidia/Apex 已被移除，取而代之的是原生火炬操作。有关从 apex 迁移到 torch 内置操作的更多信息，请参阅此处。
- 添加了 Python 3.11 支持
- 添加了 CUDA 12.4 支持
- 添加了 Ubuntu 22.04 支持

- GPU Docker 镜像包含以下库：
  - CUDA 12.4.1
  - cudnn 9.1.0.70
  - NCCL 2.23.4
  - AWS OFI NCCL 插件 1.12.1
  - EFA 安装程序 1.36.0
  - 变形金刚引擎 1.11
  - 闪电注意 2.6.3
  - GDRCopy 2.4.2
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 训练容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.23.4+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1

区域	代码
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 已在：g3.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4d.24xlarge、p4de.24xlarge、g4dn.xlarge、
- 使用 Resnet50、BERT 以及上的 ImageNet EC2数据集、ECS AMI（亚马逊 Linux AMI 2.0.20240515）和 EKS AMI（-1.25.16-20240514）进行了测试 amazon-eks-gpu-node

已知问题

- 使用的客户TransformerEngine可能会遇到 [W init.cpp: 767] 警告：torch 脚本不再支持 nvfuser，使用 _jit_set_nvfuser_enabled 已被弃用，并且由于自 2.2 起已弃用，因此禁止操作（函数运算符()）。NVFuser PyTorch 欲了解更多信息，请查看此问题。

AWS Deep Learning Conta PyTorch iners 适用于 2.5 SageMaker

AWS 适用于亚马逊（SMDLCs）的 Dee@@ p Learning Containers SageMaker () 现已推出 PyTorch 2.5 版，并在 Ubuntu 22.04 上支持 CUDA 12.4。你可以在任何 SM 服务上启动 Deep Learning Containers 的新版本。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已通过 SM 服务进行了测试，提供了稳定版本的 NVIDIA CUDA、Intel MKL 和其他组件，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 服务上使用而设计。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- PyTorch 为 2.5.1 引入了支持 SageMaker 服务的训练容器。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- PyTorch 2.5 为 SDPA 提供了全新 cuDNN 后端，默认情况下，在 H100 或更高版本上使用 SDPA 的用户启用加速功能。 GPUs此外，torch.compile 的区域编译允许用户无需重新编译即可编译重复的 nn.Module（例如 LLM 中的变压器层），从而缩短 torch.compile 的冷启动时间。最后，TorchInductor CPP 后端提供了稳定的性能加速，并提供了许多增强功能，例如 FP16 支持、CPP 封装、AOT-Inductor 模式和最大自动调谐模式。
- 包括修复 PyPI 中的轮子无法在基于 RPM 的 Linux 发行版 out-of-the-box上使用，如 2.5.1 中所述。PyTorch
- 有关更新的完整说明，请参阅此处的 PyTorch 2.5.0 官方发行说明和此处的 PyTorch 2.5.1 发行说明。
- NVidia/Apex 已被移除，取而代之的是原生火炬操作。有关从 apex 迁移到 torch 内置操作的更多信息，请参阅此处。
- 添加了 Python 3.11 支持
- 添加了 CUDA 12.4 支持
- 添加了 Ubuntu 22.04 支持
- GPU Docker 镜像包含以下库：

- CUDA 12.4.1
- cudnn 9.1.0.70
- NCCL 2.23.4
- AWS OFI NCCL 插件 1.12.1
- EFA 安装程序 1.36.0
- 变形金刚引擎 1.11
- 闪电注意 2.6.3
- GDRCopy 2.4.2

- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 训练和推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.23.4+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1

区域	代码
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 已在：g3.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4d.24xlarge、p4de.24xlarge、g4dn.xlarge、
- 使用 Resnet50、BERT 以及上的 ImageNet EC2数据集、ECS AMI（亚马逊 Linux AMI 2.0.20240515）和 EKS AMI（-1.25.16-20240514）进行了测试 amazon-eks-gpu-node

已知问题

- 使用的客户TransformerEngine可能会遇到 [W init.cpp: 767] 警告：torch 脚本不再支持 nvfuser，使用 _jit_set_nvfuser_enabled 已被弃用，并且由于自 2.2 起已弃用，因此禁止操作（函数运算符()）。NVFuser PyTorch 欲了解更多信息，请查看此问题。

AWS 适用于 PyTorch 2.5 推理的 Deep Learning Container EC2 s、ECS 和 EKS

AWS 适用于亚马逊弹性计算云 (DLCs)、亚马逊弹性@@ 容器服务 (ECSEC2) 和亚马逊弹性 Kubernetes Service (EKS) 的 Deep Learning Containers () 现已推出 2.5 版本 PyTorch ，并在 Ubuntu 22.04 上支持 CUDA 12.4。你可以在任何 ECS 和 EKS 服务上启动新版本的 Deep Learn EC2 ing Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 CPU 和 GPU 上进行推理的容器映像，并针对性能和扩展进行了 AWS优化。这些 Docker 镜像已通过 ECS 和 EKS 服务进行了测试，并提供了 NVIDIA CUDA、cuDNN、Intel MKL 和其他组件的稳定版本，为运行深度学习工作负载提供了优化的用户体验。 EC2 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何 ECS 和 EKS 服务上使用而设计。 EC2如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- PyTorch 为 2.5.1 引入了用于推理支持 EC2、ECS 和 EKS 的容器。有关此版本的详细信息，请查看我们的 GitHub发布标签。
- PyTorch 2.5 为 SDPA 提供了全新 cuDNN 后端，默认情况下，在 H100 或更高版本上使用 SDPA 的用户启用加速功能。 GPUs此外，torch.compile 的区域编译提供了一种缩短 torch.compile 冷启动时间的方法，它允许用户在不重新编译的情况下编译重复的 nn.Module（例如 LLM 中的变压器层）。最后， TorchInductor CPP 后端提供了稳定的性能加速，并提供了许多增强功能，例如 FP16 支持、CPP 封装、AOT-Inductor 模式和最大自动调谐模式。
- 包括修复 PyPI 中的轮子无法在基于 RPM 的 Linux 发行版 out-of-the-box上使用，如 2.5.1 中所述。PyTorch
- 有关更新的完整说明，请参阅此处的 PyTorch 2.5.0 官方发行说明和此处的 PyTorch 2.5.1 发行说明。
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.23.4+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2

区域	代码
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1

区域	代码
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试基于：c5.18xlarge、g3.16xlarge、m5.16xlarge、t3.2xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.
- 在 EC2 ECS AMI（亚马逊 Linux AMI2 .0.20221102）和 EKS AMI（-1.25.16-20240307）上的 MNIST 和 Resnet50/ ImageNet 数据集、ECS AMI（-1.25.16-20240307）上进行了测试 amazon-eks-gpu-node

AWS Deep Learning Contain PyTorch ers 适用于 2.5 推理 SageMaker

AWS 适用于亚马逊 SageMaker 的 Dee@@ p Learning Containers (DLC) 现已在 Ubuntu PyTorch 22.04 上市，支持 2.5，支持 CUDA 12.4。你可以在 SageMaker 服务上启动新版本的 Deep Learning Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

此版本包括用于在 CPU 和 GPU 上进行推理的容器映像，并针对性能和扩展进行了 AWS优化。这些 Docker 镜像已经过 SageMaker 服务测试，提供稳定版本的 NVIDIA CUDA、cuDNN 和其他组件，为运行深度学习工作负载提供优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 推理服务上使用而设计。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- PyTorch 为 2.5.1 引入了用于推理支持 SageMaker 服务的容器。有关此版本的详细信息，请查看我们的 GitHub发布标签。

- PyTorch 2.5 为 SDPA 提供了全新 cuDNN 后端，默认情况下，在 H100 或更高版本上使用 SDPA 的用户启用加速功能。 GPUs此外，torch.compile 的区域编译提供了一种缩短 torch.compile 冷启动时间的方法，它允许用户在不重新编译的情况下编译重复的 nn.Module（例如 LLM 中的变压器层）。最后， TorchInductor CPP 后端提供了稳定的性能加速，并提供了许多增强功能，例如 FP16 支持、CPP 封装、AOT-Inductor 模式和最大自动调谐模式。
- 包括修复 PyPI 中的轮子无法在基于 RPM 的 Linux 发行版 out-of-the-box上使用，如 2.5.1 中所述。PyTorch
- 有关更新的完整说明，请参阅此处的 PyTorch 2.5.0 官方发行说明和此处的 PyTorch 2.5.1 发行说明。
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.23.4+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1

区域	代码
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试基于：c5.18xlarge、g3.16xlarge、m5.16xlarge、t3.2xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.
- 在 EC2 ECS AMI（亚马逊 Linux AMI2 .0.20221102）和 EKS AMI（-1.25.16-20240307）上的 MNIST 和 Resnet50/ ImageNet 数据集、ECS AMI（-1.25.16-20240307）上进行了测试 amazon-eks-gpu-node

AWS 适用于 ECS 和 EKS 上的 PyTorch 2.5 ARM64 推理的 Deep Lear EC2 ning Containers

AWS 适用于亚马逊弹性 Kubernetes Service (EKS)、亚马逊弹性计算云 () 和亚马逊弹性@@ 容器服务 EC2 (ECS) 的深度学习容器 (DLC) 现已 ARM64 适用于平台，包括 Graviton 实例类型 AWS ，支持 2.5。 PyTorch

此版本包括用于在 CPU 和 GPU 上进行推理的容器映像，并针对性能和扩展进行了 AWS优化。CPU 映像已通过每个 EC2、ECS 和 EKS 服务进行了测试，而 GPU 映像仅支持 EC2 （参见下表）。GPU 映像提供 NVIDIA CUDA、cuDNN、NCCL 和其他组件的稳定版本。对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。

	EC2	ECS	EKS
CPU 镜像	支持	支持	支持
GPU 镜像	支持	不支持	不支持

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 在 PyTorch 2.5.1 中引入了用于推理支持的容器 EC2、ECS 和实例上 ARM64 的 EKS。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.5 开始，我们将 Graviton 的名称更改 DLCs 为， ARM64 DLCs 以概括其用法。例如，ECR 存储库名称现在为 “pytorch-inference-arm64”，而不是 “” pytorch-inference-graviton。Graviton DLCs 和，在功能上 ARM64 DLCs 是等同的。
- TorchServe 版本：0.12.0
- GPU 映像应与 g5G 实例类型一起使用，后者由 G raviton CPUs 和 NVIDIA T4G Tensor Core 提供支持。 GPUs
- 包括修复 PyPI 中的轮子无法在基于 RPM 的 Linux 发行版 out-of-the-box上使用，如 2.5.1 中所述。 PyTorch
- 有关更新的完整说明，请参阅此处的 PyTorch 2.5.0 官方发行说明和此处的 PyTorch 2.5.1 发行说明。

性能改进

它们 DLCs 继续在Graviton CPU上为BERT和Ro BERTa 情绪分析以及填充掩模模型提供最佳性能，这使Graviton3成为这些模型在云端最具成本效益的CPU平台。 AWS 欲了解更多信息，请参阅 Graviton PyTorch 用户指南。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch ARM64 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持上述每项服务所支持的 Graviton CPU 实例类型。

GPU 实例类型支持

这些容器支持 Graviton GPU 实例类型 g5G，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.21.5+cuda12.4

AWS; 区域支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1

区域	代码
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
中部（加拿大）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1

区域	代码
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- 经过测试：c8g.4xlarge、t4g.2xlarge、r8g.2xlarge、m7g.4xlarge、g5g.4xlarge、g5g.4xlarge
- 在 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20220822 arm64）和 EKS AMI（1.25.6-20230304 arm64）上的 MNIST 和 Resnet50/ DenseNet 数据集上进行了测试

已知问题

- ARM64/aarch64 还没有官方的 Triton 发行版，因此一些 torch.compile 工作负载会失败，原因是：

```
torch._dynamo.exc.BackendCompilerFailed: backend='inductor' raised:
RuntimeError: Cannot find a working triton installation. More information on
 installing Triton can be found at https://github.com/openai/triton
```

AWS 在 Sagemaker 上用于 PyTorch 2.5 ARM64 推理的深度学习容器

AWS 适用于亚马逊 SageMaker 的 Deep Learning Containers (DLCs) 现已适用于 ARM64 平台，包括支持 2.5 的 AWS Graviton 实例类型。 PyTorch 你可以在上启动新版本的 DLC。 SageMaker

此版本包括一个用于在 CPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这个 Docker 镜像已在上面进行了测试。 SageMaker它为在上运行深度学习工作负载提供了优化的用户体验 SageMaker。扫描此镜像中的所有软件组件是否存在安全漏洞，并根据 AWS 安全最佳实践进行更新或修补。

可用容器列表可以在我们的文档中找到。请参阅 SageMaker Graviton 博客和 DLC 开发者指南，将深度学习工作负载迁移到 Graviton 实例。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 引入了 PyTorch 2.5.1 的容器，用于实例上的推理支持 SageMaker 服务。 ARM64 有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- 从 PyTorch 2.5 开始，我们将 Graviton 的名称更改 DLCs 为， ARM64 DLCs 以概括其用法。例如，ECR 存储库名称现在为 “pytorch-inference-arm64”，而不是 “” pytorch-inference-graviton。Graviton DLCs 和，在功能上 ARM64 DLCs 是等同的。
- TorchServe 版本：0.12.0
- 包括修复 PyPI 中的轮子无法在基于 RPM 的 Linux 发行版 out-of-the-box上使用，如 2.5.1 中所述。PyTorch
- 有关更新的完整说明，请参阅此处的 PyTorch 2.5.0 官方发行说明和此处的 PyTorch 2.5.1 发行说明。

性能改进

它们 DLCs 继续在Graviton上为BERT和Ro BERTa 情绪分析以及填充掩模模型提供最佳性能，使Graviton3成为这些模型在云端最具成本效益的CPU平台。 AWS 欲了解更多信息，请参阅 Graviton PyTorch 用户指南。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch ARM64 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持下支持的 Graviton CPU 实例类型。 SageMaker

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1

区域	代码
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- 经过测试：c8g.4xlarge、t4g.2xlarge、r8g.2xlarge、m7g.4xlarge、g5g.4xlarge、g5g.4xlarge
- 在 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20220822 arm64）和 EKS AMI（1.25.6-20230304 arm64）上的 MNIST 和 Resnet50/ DenseNet 数据集上进行了测试

已知问题

- 无

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

AWS 适用于 PyTorch 2.4 训练的 Deep Learning Containers EC2、ECS 和 EK

AWS 适用于亚马逊弹性计算云 (DLCs)、亚马逊弹性@@ 容器服务 (ECSEC2) 和亚马逊弹性 Kubernetes Service (EKS) 的 Deep Learning Containers () 现已推出 2.4 版本 PyTorch ，并在 Ubuntu 22.04 上支持 CUDA 12.4。你可以在任何 ECS 和 EKS 服务上启动新版本的 Deep Learn EC2 ing Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已通过 ECS 和 EKS 服务进行了测试，并提供了 NVIDIA CUDA、Intel MKL 和其他组件的稳定版本，为运行深度学习工作负载提供了优化的用户体验。 EC2 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何 ECS 和 EKS 服务上使用而设计。 EC2如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- PyTorch 为 2.4.0 引入了用于训练的容器 EC2，该容器支持 ECS 和 EKS。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- PyTorch 2.4 支持 python 自定义运算符 API，允许用户将 Triton 内核等自定义内核集成到 torch.compile 中。
- PyTorch 2.4 还提供 AOTInductor 冻结功能，允许进行更多 AOTInductor 优化。它还提供了一个使用libuv的新默认 TCPStore 服务器后端，这应该可以减少大规模作业的初始化时间。
- 有关更新的完整说明，请参阅此处的 PyTorch 2.4 官方发行说明。
- 添加了 Python 3.11 支持
- 添加了 CUDA 12.4 支持
- 添加了 Ubuntu 22.04 支持

- GPU Docker 镜像包含以下库：
  - CUDA 12.4.1
  - cudnn 9.1.0.70
  - NCCL 2.22.3
  - AWS OFI NCCL 插件 1.11.0
  - EFA 安装程序 1.34.0
  - 变压器引擎 1.9
  - 闪电注意 2.4.2
  - GDRCopy 2.4.1
  - Apex 24.04.01
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 训练容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.22.3+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
中部（加拿大）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1

区域	代码
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 已在：g3.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4d.24xlarge、p4de.24xlarge、g4dn.xlarge、
- 使用 Resnet50、BERT 以及上的 ImageNet EC2数据集、ECS AMI（亚马逊 Linux AMI 2.0.20240515）和 EKS AMI（-1.25.16-20240514）进行了测试 amazon-eks-gpu-node

已知问题

- 使用的客户TransformerEngine可能会遇到 [W init.cpp: 767] 警告：torch 脚本不再支持 nvfuser，使用 _jit_set_nvfuser_enabled 已被弃用，并且由于自 2.2 起已弃用，因此禁止操作（函数运算符()）。NVFuser PyTorch 欲了解更多信息，请查看此问题。

AWS 适用于 PyTorch 2.4 训练的深度学习容器 SageMaker

AWS 适用于亚马逊（SMDLCs）的 Dee@@ p Learning Containers SageMaker () 现已推出 PyTorch 2.4 版本，并在 Ubuntu 22.04 上支持 CUDA 12.4。你可以在任何 SM 服务上启动 Deep Learning Containers 的新版本。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已通过 SM 服务进行了测试，提供了稳定版本的 NVIDIA CUDA、Intel MKL 和其他组件，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新的 DLC 专为在 SM 服务上使用而设计。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为 PyTorch 2.4.0引入了支持 SageMaker 服务的训练容器。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- PyTorch 2.4 支持 python 自定义运算符 API，允许用户将 Triton 内核等自定义内核集成到 torch.compile 中。
- PyTorch 2.4 还提供 AOTInductor 冻结功能，允许进行更多 AOTInductor 优化。它还提供了一个使用libuv的新默认 TCPStore 服务器后端，这应该可以减少大规模作业的初始化时间。
- 有关更新的完整说明，请参阅此处的 PyTorch 2.4 官方发行说明。
- 添加了 EC2 P5 实例支持
- 添加了 Python 3.11 支持
- 添加了 CUDA 12.4 支持
- 添加了 Ubuntu 22.04 支持
- GPU Docker 镜像包含以下库：
  - CUDA 12.4.1
  - cudnn 9.1.0.70
  - NCCL 2.22.3
  - AWS OFI NCCL 插件 1.11.0
  - EFA 安装程序 1.34.0

- 变压器引擎 1.9
- 闪电注意 2.4.2
- GDRCopy 2.4.1
- Apex 24.04.01
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 训练和推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.22.3+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2

区域	代码
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3

区域	代码
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 已在：g3.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4d.24xlarge、p4de.24xlarge、g4dn.xlarge、
- 使用 Resnet50、BERT 以及上的 ImageNet EC2数据集、ECS AMI（亚马逊 Linux AMI 2.0.20240515）和 EKS AMI（-1.25.16-20240514）进行了测试 amazon-eks-gpu-node

已知问题

- 使用的客户TransformerEngine可能会遇到 [W init.cpp: 767] 警告：torch 脚本中不再支持 nvfuser，使用 _jit_set_nvfuser_enabled 已被弃用，并且由于自 2.2 起已弃用，因此禁止操作（函数运算符()）。NVFuser PyTorch 欲了解更多信息，请查看此问题。

AWS 适用于 PyTorch 2.4 推理的 Deep Learning Container EC2 s、ECS 和 EKS

AWS 适用于亚马逊弹性计算云 (DLCs)、亚马逊弹性@@ 容器服务 (ECSEC2) 和亚马逊弹性 Kubernetes Service (EKS) 的 Deep Learning Containers () 现已推出 2.4 版本 PyTorch ，并在 Ubuntu 22.04 上支持 CUDA 12.4。你可以在任何 ECS 和 EKS 服务上启动新版本的 Deep Learn EC2 ing Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下文。

此版本包括用于在 CPU 和 GPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这些 Docker 镜像已通过 ECS 和 EKS 服务进行了测试，并提供了 NVIDIA CUDA、cuDNN、Intel MKL 和其他组件的稳定版本，为运行深度学习工作负载提供了优化的用户体验。 EC2 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何 ECS 和 EKS 服务上使用而设计。 EC2如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- PyTorch 为 2.4.0 引入了用于推理支持 EC2、ECS 和 EKS 的容器。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- PyTorch 2.4 支持 python 自定义运算符 API，允许用户将 Triton 内核等自定义内核集成到 torch.compile 中。
- 有关更新的完整说明，请参阅此处的 PyTorch 2.4 官方发行说明。
- 添加了 Python 3.11 支持
- 添加了 CUDA 12.4 支持
- 添加了 Ubuntu 22.04 支持
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.22.3+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2

区域	代码
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
中部（加拿大）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1

区域	代码
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试基于：c5.18xlarge、g3.16xlarge、m5.16xlarge、t3.2xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.
- 在 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20221102）和 EKS AMI（-1.25.16-20240307）上的 MNIST 和 Resnet50/ ImageNet 数据集、ECS AMI（-1.25.16-20240307）上进行了测试 amazon-eks-gpu-node

AWS Deep Learning Contain PyTorch ers 适用于 2.4 推理 SageMaker

AWS 适用于亚马逊 SageMaker 的 Dee@@ p Learning Containers (DLC) 现已在 Ubuntu 2 PyTorch 2.04 上市，支持 2.4 版本和 CUDA 12.4。你可以在 SageMaker 服务上启动新版本的 Deep Learning Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

此版本包括用于在 CPU 和 GPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这些 Docker 镜像已经过 SageMaker 服务测试，提供稳定版本的 NVIDIA CUDA、cuDNN 和其他组件，为运行深度学习工作负载提供优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 推理服务上使用而设计。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为 PyTorch 2.4 引入了用于推理支持 SageMaker 服务的容器。有关此版本的详细信息，请查看我们的 GitHub 发布标签。

- PyTorch 2.4 支持 python 自定义运算符 API，允许用户将 Triton 内核等自定义内核集成到 torch.compile 中。
- 有关更新的完整说明，请参阅此处的 PyTorch 2.4 官方发行说明。
- 添加了 Python 3.11 支持
- 添加了 CUDA 12.4 支持
- 添加了 Ubuntu 22.04 支持
- 添加了 TorchServe 0.11.1 支持
- 10/25/24：更新 TorchServe 至 0.12.0（发布标签）
- 适用于 CPU 的 Dockerfile 可以在这里找到，GPU 的 Dockerfile 可以在这里找到。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持 x86_64 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.1
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.22.3+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1

区域	代码
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试基于：c5.18xlarge、g3.16xlarge、m5.16xlarge、t3.2xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.
- 在 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20221102）和 EKS AMI（-1.25.16-20240307）上的 MNIST 和 Resnet50/ ImageNet 数据集上进行了测试 amazon-eks-gpu-node

AWS 适用于 PyTorch 2.4 Graviton on EC2、ECS 和 EKS 的 Deep Learning Containers

AWS 适用于亚马逊弹性 Kubernetes Service (EKS)、亚马逊弹性计算云 () 和亚马逊弹性@@ 容器服务 EC2 (ECS) 的深度学习容器 (DLC) 现已可用于支持 2.4 的 Graviton 实例类型。 PyTorch

此版本包括用于在 CPU 和 GPU 上进行推理的容器映像，并针对性能和扩展进行了 AWS优化。CPU 映像已通过每个 EC2、ECS 和 EKS 服务进行了测试，而 GPU 映像仅支持 EC2 （参见下表）。GPU 映像提供 NVIDIA CUDA、cuDNN、NCCL 和其他组件的稳定版本。对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。

	EC2	ECS	EKS
Graviton CPU	支持	支持	支持
Graviton 搭载 NVIDIA G	支持	不支持	不支持

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 在 Graviton 实例上引入了 PyTorch 2.4 版本的容器以支持推理 EC2、ECS 和 EKS。有关此版本的详细信息，请查看我们的 GitHub 发布标签：适用于 CPU 和 GPU。
- TorchServe 版本：0.11.1
- 11/01/24：更新 TorchServe 至 0.12.0（发布标签：适用于 CPU 和 GPU）
- GPU 镜像是有史以来第一个支持 Graviton (ARM64) + GPU 平台的 DLC。它应该与 g5G 实例类型一起使用，后者由 G raviton CPUs 和 NVIDIA T4G Tensor Core 提供支持。 GPUs
- 有关框架更新，请参阅此处的 PyTorch 2.4 官方发行说明。

性能改进

它们 DLCs 继续在Graviton CPU上为BERT和Ro BERTa 情绪分析以及填充掩模模型提供最佳性能，这使Graviton3成为这些模型在云端最具成本效益的CPU平台。 AWS 欲了解更多信息，请参阅 Graviton PyTorch 用户指南。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch Graviton 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持上述每项服务所支持的 Graviton CPU 实例类型。

GPU 实例类型支持

这些容器支持 Graviton GPU 实例类型 g5G，并包含以下支持 GPU 的软件组件：

- CUDA 12.4.0
- cudnn 9.1.0.70+cuda12.4
- NCCL 2.20.5+cuda12.4

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1

区域	代码
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
中部（加拿大）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1

区域	代码
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- 经过测试：c7g.4xlarge、c6g.4xlarge、t4g.2xlarge、r6g.2xlarge、m6g.4xlarge、g5g.4xlarge、g5g.4xlarge、g5g.4xla
- 在 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20220822 arm64）和 EKS AMI（1.25.6-20230304 arm64）上的 MNIST 和 Resnet50/ DenseNet 数据集上进行了测试

已知问题

- ARM64/aarch64 还没有官方的 Triton 发行版，因此一些 torch.compile 工作负载会失败，原因是：

```
torch._dynamo.exc.BackendCompilerFailed: backend='inductor' raised:
RuntimeError: Cannot find a working triton installation. More information on
 installing Triton can be found at https://github.com/openai/triton
```

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

AWS PyTorch 2.4 Graviton 的 Deep Learning Containers SageMaker

AWS 适用于亚马逊 SageMaker 的 De@@ ep Learning Containers (DLCs) 现已可用于 PyTorch 支持 2.4 的 Graviton 实例类型。你可以在上启动新版本的 DLC。 SageMaker

此版本包括一个用于在 CPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这个 Docker 镜像已在上面进行了测试。 SageMaker它为在上运行深度学习工作负载提供了优化的用户体验 SageMaker。扫描此镜像中的所有软件组件是否存在安全漏洞，并根据 AWS 安全最佳实践进行更新或修补。

可用容器列表可以在我们的文档中找到。请参阅 SageMaker Graviton 博客和 DLC 开发者指南，将深度学习工作负载迁移到 Graviton 实例。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 引入了 PyTorch 2.4 版的容器，用于在 Graviton SageMaker 实例上提供推理支持服务。有关此版本的详细信息，请查看我们的 GitHub 发布标签。
- TorchServe 版本：0.11.1
- 10/25/24：更新 TorchServe 至 0.12.0（发布标签）
- 有关框架更新，请参阅此处的 PyTorch 2.4 官方发行说明。

性能改进

它们 DLCs 继续在Graviton上为BERT和Ro BERTa 情绪分析以及填充掩模模型提供最佳性能，使Graviton3成为这些模型在云端最具成本效益的CPU平台。 AWS 欲了解更多信息，请参阅 Graviton PyTorch 用户指南。

安全建议

AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 3.11 Support

PyTorch Graviton 推理容器支持 Python 3.11。

CPU 实例类型支持

这些容器支持下支持的 Graviton CPU 实例类型。 SageMaker

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1

区域	代码
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- 经过测试：c7g.4xlarge、c6g.4xlarge、t4g.2xlarge、r6g.2xlarge、m6g.4xlarge、m6g.4xlarge
- 在 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20220822 arm64）和 EKS AMI（1.25.6-20230304 arm64）上的 MNIST 和 Resnet50/ DenseNet 数据集上进行了测试

已知问题

- 无

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

TensorFlow 深度学习 Containers

以下是 Dee TensorFlow p Learning Containers 的发行说明：

版本	类型	服务	架构	发行说明
2.18	训练	EC2 ECS EKS	X86	AWS 适用于 TensorFlow 2.18 的 Deep Learning Containers（在 EC2、ECS 和 EKS 上训练）：2025 年 1 月 29 日
2.18	训练	SageMaker	X86	AWS 适用于 TensorFlow 2.18 的 Deep Learning Containers（ SageMaker 人工智能训练）：2025 年 1 月 29 日
2.18	推理	EC2 ECS EKS	X86	AWS 适用于 TensorFlow 2.18 的 Deep Learning Containers（在

版本	类型	服务	架构	发行说明
				ECS 和 EKS 上 EC2进行推理）：2025 年 1 月 29 日
2.18	推理	SageMaker	X86	AWS 适用于 TensorFlow 2.18 的 Deep Learning Containers（ SageMaker 人工智能推断）：2025 年 1 月 29 日
2.18	推理	EC2 ECS EKS	Arm64	AWS 适用于 TensorFlow 2.18 的 Deep Learning Containers ARM64（在 ECS 和 EKS 上 EC2进行推理）：2025 年 3 月 3 日
2.18	推理	SageMaker	Arm64	AWS 适用于 TensorFlow 2.18 的 Deep Learning Containers ARM64（ SageMaker 人工智能推断）：2025 年 3 月 3 日

AWS 适用于 TensorFlow 2.18 训练的 Deep Learning Contain EC2 ers、ECS 和 EKS

AWS 适用于亚马逊弹性 Kubernetes Service (EKS)、亚马逊弹性计算云 () 和亚马逊弹性@@ 容器服务 (ECS) 的深度学习容器 EC2 (DLC) 现已在 Ubuntu 22.04 上 TensorFlow 支持 2.18 和 CUDA 12.5。你可以在任何 ECS EC2、EKS 服务上启动 Deep Learning Containers 的新版本。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

此版本包括用于在 CPU 和 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已在 ECS EC2、EKS 服务中进行了测试，并提供了稳定版本的 NVIDIA CUDA、cuDNN 和其他组件，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何 ECS EC2、EKS 服务上使用而设计。如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为 ECS EC2、EKS 引入了 TensorFlow 2.18 版的容器
- 有关 TensorFlow 2.18 EC2、ECS、EKS Training DLC 的更多详细信息，请参阅 v1.0-tf-ec2-2.18.0-tr-py310。
- 由于 Nvidia 驱动程序不兼容，此 DLC EC2 无法在 P2 实例系列上运行。

弃用

1. Horovod 包裹：TF 2.14 及更高版本的 Horovod 包裹已停止发货。 DLCs 客户将能够通过 forderedlistlist遵循指南来安装horovod库，并将其安装在他们的 DLCs 分布式训练作业上。
2. 为了改善代码运行状况，CUDA 版本中已禁用 TensorRT 支持，请参阅 TF 2.18 版本。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 支持

已安装的深度学习框架的容器中支持 Python 3.10。

CPU 实例类型支持

容器支持 CPU 实例类型。 TensorFlow 是在支持 OneDNN 库的基础上构建的。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含支持 GPU 的 forderedListlowing 软件组件。

- CUDA 12.5
- cudnn 9.3
- NCCL 2.23.4-1

AWS 地区支持

这些容器在以下排序列表区域可用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2

区域	代码
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧盟（Stockhorderedlistm）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1

区域	代码
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 经过测试的 DLC 图片：c4.8xlarge、c5.18xlarge、m4.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4de.24xlarge、p4
- 在 EC2 ECS AMI（亚马逊 Linux AMI 2.0.20250121）和 EKS AMI（1.25-v20250116）上的 MNIST 和 Resnet50/ ImageNet 数据集上进行了测试

已知问题

1. 使用 s3 文件系统时，Tensorflow IO 软件包会引发异常（问题链接）。因此，在上游提供修复程序之前，此 DLC 将不支持依赖于 Tensorflow IO 的 s3 功能的功能。很少有这样的不支持的功能是 s3 插件、s3 检查点和 s3 记录获取。

AWS 适用于 TensorFlow 2.18 训练的 Deep Learning Containers SageMaker

AWS 适用于亚马逊 SageMaker 的 Dee@@ p Learning Containers (DLC) 现已上市，在 Ubunt TensorFlow u 22.04 上支持 2.18，支持 CUDA 12.5。你可以在 SageMaker 服务上启动新版本的 Deep Learning Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

此版本包括用于在 CPU 和 GPU 上训练的容器镜像，针对性能和扩展进行了优化 AWS。这些 Docker 镜像已经过 SageMaker 服务测试，提供稳定版本的 NVIDIA CUDA、cuDNN 和其他组件，为运行深度学习工作负载提供优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 训练服务中使用而设计。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- TensorFlow 为 2.18 引入了容器 SageMaker
- 有关 TensorFlow 2.18 训练 DLCs的更多详情，请参阅 v 1.0-tf-sagemaker-2.18.0-tr-py310。
- 由于 Nvidia 驱动程序不兼容，此 DLC SageMaker 无法在 P2 实例系列上运行。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

Package 弃用

- Sagemaker Tensorflow 软件包未在 TF2 .16 DLCs 及以上版本中维护，因此不随此 DLC 一起提供。因此，这些 Sag DLCs emaker 将不支持管道模式。
- TF 2.14 DLCs 及以上版本的 Horovod 包裹已停止发货。客户将能够通过forderedlistlist遵循指南来安装horovod库，并将其安装在他们的 DLCs 分布式训练作业上。
- SageMaker TF 2.14 DLCs 及更高版本不包括@@ 数据并行。此功能在我们的最新 PyTorch 图像中仍然可用。
- 为了改善代码运行状况，CUDA 版本中已禁用 TensorRT 支持，请参阅 TF 2.18 版本。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 支持

已安装的深度学习框架的容器中支持 Python 3.10。

CPU 实例类型支持

容器支持 CPU 实例类型。 TensorFlow 是在支持 OneDNN 库的基础上构建的。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含支持 GPU 的 forderedListlowing 软件组件。

- CUDA 12.5
- cudnn 9.3

- NCCL 2.23.4-1

AWS 地区支持

这些容器在以下排序列表区域可用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1

区域	代码
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧盟（Stockhorderedlistm）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 经过测试的 DLC 图片：
  c4.8xlarge、c5.18xlarge、m4.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、p4de.24xlarge、p4

已知问题

1. 使用 s3 文件系统时，Tensorflow IO 软件包会引发异常（问题链接）。因此，在上游提供修复程序之前，此 DLC 将不支持依赖于 Tensorflow IO 的 s3 功能的功能。很少有这样不支持的功能是 s3 插件、s3 检查点、s3 记录获取和 Sagemaker 上的 Parameter Server 训练。

AWS 适用于 TensorFlow 2.18 推理的 Deep Learning Containers EC2、ECS 和 EKS

AWS 适用于亚马逊弹性 Kubernetes Service (EKS)、亚马逊弹性计算云 () 和亚马逊弹性@@ 容器服务 (ECS) 的深度学习容器 EC2 (DLC) 现已在 Ubuntu 20.04 上 TensorFlow 支持 2.18 和 CUDA 12.2。你可以在任何 ECS EC2、EKS 服务上启动 Deep Learning Containers 的新版本。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

-此版本包括用于在 CPU 和 GPU 上进行推理的容器映像，针对性能和扩展进行了优化。 AWS这些 Docker 镜像已在 ECS EC2、EKS 服务中进行了测试，并提供了 NVIDIA CUDA、cuDNN 和其他组件的稳定版本，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在任何 ECS EC2、EKS 服务上使用而设计。如果您正在寻找可搭配使用的 DLC SageMaker，请参阅此文档。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 为 ECS EC2、EKS 引入了 TensorFlow 2.18 版的容器
- 有关 TensorFlow 2.18 EC2、ECS、EKS Inference DLC 的更多详细信息，请参阅 v1.0-tf-ec2-2.18.0-inf-py310。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 支持

已安装的深度学习框架的容器中支持 Python 3.10。

CPU 实例类型支持

这些容器支持 x86_64 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件。

- CUDA 12.2
- cudnn 8.9.4.25-1+cuda12.2
- NCCL 2.21.5-1+cuda12.2

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2

区域	代码
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1

区域	代码
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 经过测试的 DLC 图片：
  c4.8xlarge、c5.18xlarge、m4.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.xlarge、g4dn

AWS 适用于 TensorFlow 2.18 推理的 Deep Learning Containers SageMaker

AWS 适用于亚马逊 SageMaker 的 Dee@@ p Learning Containers (DLC) 现已上市，支持 TensorFlow 2.18 推理，并在 Ubuntu 20.04 上支持 CUDA 12.2。你可以在 SageMaker 服务上启动新版本的 Deep Learning Containers。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

此版本包括用于在 CPU 和 GPU 上进行推理的容器镜像，针对性能和扩展进行了 AWS优化。这些 Docker 镜像已经过 SageMaker 服务测试，提供稳定版本的 NVIDIA CUDA、cuDNN 和其他组件，为运行深度学习工作负载提供优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。这些新 DLC 专为在 SageMaker 推理服务上使用而设计。

可用容器列表可以在我们的文档中找到。有关最新更新，另请参阅 aws/ 存储deep-learning-containers GitHub 库。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 引入了 TensorFlow 2.18 的容器用于推断。 SageMaker
- 有关 2.18 Inference 的更多详细信息，请参阅 v1.0-tf-sagemaker- TensorFlow 2.18.0-in DLCs f-py310。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 支持

TensorFlow 推理容器支持 Python 3.10。

CPU 实例类型支持

容器支持 CPU 实例类型。

GPU 实例类型支持

这些容器支持 GPU 实例类型，并包含以下支持 GPU 的软件组件：

- CUDA 12.2
- cudnn 8.9.4.25-1+cuda12.2
- NCCL 2.21.5-1+cuda12.2

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2

区域	代码
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1

区域	代码
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c5.18xlarge
- 测试在：c4.8xlarge、c5.18xlarge、m4.16xlarge、p3.16xlarge、p3dn.24xlarge、p4d.24xlarge、g4dn.xlarge、g4dn

AWS 适用于 TensorFlow 2.18 ARM64 推理的 Deep Learning Containers 在 ECS EC2 和 EKS

AWS 适用于亚马逊 Elastic Kubernetes Service (EKS)、亚马逊弹性计算云 () 和亚马逊弹性@@ 容器服务 EC2 (ECS) 的深度学习容器 (DLC) 现已 ARM64 适用于平台，包括支持 Service 2.18 的 Graviton 实例类型。 TensorFlow 你可以在任何 ECS EC2、EKS 服务上启动 Deep Learning Containers 的新版本。有关 Dee AWS p Learning Containers 支持的框架和版本的完整列表，请参阅下面的发行说明。

此版本包括用于在 CPU 上进行推理的容器映像，针对性能和扩展进行了 AWS优化。这些 Docker 镜像已在 ECS EC2、EKS 服务中进行了测试，为运行深度学习工作负载提供了优化的用户体验。 AWS对这些映像中的所有软件组件进行安全漏洞扫描，并根据 AWS 安全最佳实践进行更新或修补。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。有关运行时配置和性能调整，请参阅 AWS Graviton 技术指南。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- 支持在 ECS 和 EKS TensorFlow 服务上 EC2提供 2.18 服务。有关此版本的详细信息，请查看我们的 GitHub v1.0-tf-arm64-ec2-2.18.0-310 inf-cpu-py

- 从 TensorFlow 2.18 开始，我们将 Graviton 的名称更改为 DLCs ， ARM64 DLCs 以概括其用法。例如，ECR 存储库名称现在为 “tensorflow-inference-arm64”，而不是 “” tensorflow-inference-graviton。Graviton DLCs 和，在功能上 ARM64 DLCs 是等同的。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 支持

已安装的深度学习框架的容器中支持 Python 3.10。

CPU 实例类型支持

这些容器支持 Graviton CPU 实例类型。

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3

区域	代码
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1

区域	代码
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- DLC 图片经过测试：c8g.4xlarge、t4g.2xlarge、r8g.2xlarge、m7g.4xlarge、m7g.4xlarge

AWS 适用于 TensorFlow 2.18 ARM64 推理的 Deep Learning Containers SageMaker

AWS 适用于亚马逊 SageMaker的 Dee@@ p Learning Containers (DLCs) 现已适用于 ARM64 平台，包括支持 Service 2.18 的 Graviton 实例类型 TensorFlow 。你可以在上启动新版本的 Deep Learning Containers SageMaker。

此版本包括一个容器镜像，用于在 CPU 上进行推理，针对性能和扩展进行了优化。 AWS此 Docker 镜像已在亚马逊 SageMaker上进行了测试。它为在上运行深度学习工作负载提供了优化的用户体验 SageMaker。扫描此镜像中的所有软件组件是否存在安全漏洞，并根据 AWS 安全最佳实践进行更新或修补。

可用容器列表可以在我们的文档中找到。使用我们开发者指南中的入门指南和从初学者到高级级别的教程，快速开始使用 Dee AWS p Learning Containers。您也可以订阅我们的讨论论坛以获取发布公告并发布您的问题。

发布说明

- Suppor TensorFlow t 支持在亚马逊 SageMaker上提供 2.18。有关此版本的详细信息，请查看我们的 GitHub v1.0-tf-arm64-sagemaker-2.18.0-310 inf-cpu-py
- 从 TensorFlow 2.18 开始，我们将 Graviton 的名称更改为 DLCs ， ARM64 DLCs 以概括其用法。例如，ECR 存储库名称现在为 “tensorflow-inference-arm64”，而不是 “” tensorflow-inference-graviton。Graviton DLCs 和，在功能上 ARM64 DLCs 是等同的。

有关最新更新，请参阅 aws/ 存储deep-learning-containers GitHub 库。

安全建议

- AWS 建议客户监控安全公告中的关键AWS 安全更新。

Python 支持

已安装的深度学习框架的容器中支持 Python 3.10。

CPU 实例类型支持

该容器支持 Graviton CPU 实例类型。

AWS 地区支持

这些容器可在以下地区使用：

区域	代码
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
美国西部（加利福尼亚北部）	us-west-1
AF South（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（孟买）	ap-south-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区（东京）	ap-northeast-1

区域	代码
亚太地区（墨尔本）	ap-southeast-4
亚太地区（雅加达）	ap-southeast-3
亚太地区（悉尼）	ap-southeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（马来西亚）	ap-southeast-5
加拿大（中部）	ca-central-1
加拿大（卡尔加里）	ca-west-1
欧洲（苏黎世）	eu-central-2
欧洲（法兰克福）	eu-central-1
欧洲（爱尔兰）	eu-west-1
欧洲（伦敦）	eu-west-2
欧盟（巴黎）	eu-west-3
欧盟（西班牙）	eu-south-2
欧盟（米兰）	eu-south-1
欧洲（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南非（圣保罗）	sa-east-1
中国（北京）	cn-north-1

区域	代码
中国（宁夏）	cn-northwest-1

构建和测试

- 建立在：c6g.2xlarge
- DLC 图片经过测试：c8g.4xlarge、t4g.2xlarge、r8g.2xlarge、m7g.4xlarge、m7g.4xlarge

有新的更新时收到通知

每当有新 DLC 发布时，您都会收到通知。通过 Amazon SNS 使用以下主题发布通知。

```
arn:aws:sns:us-west-2:767397762724:dlc-updates
```

发布新 DLC 时，会在此处发布消息。消息中将包含容器的版本、元数据和区域图像 URI。

可以使用几种不同的方法接收这些消息。我们建议您使用以下方法。

1. 打开 Amazon SNS 控制台。
2. 如有必要，在导航栏中将 AWS 区域更改为美国西部（俄勒冈）。必须选择在其中创建您订阅的 SNS 通知的区域。
3. 在导航窗格中，依次选择订阅、创建订阅。
4. 对于 Create subscription 对话框，执行以下操作：
   a. 对于主题 ARN，复制并粘贴以下 Amazon 资源名称（ARN）：**`arn:aws:sns:us-west-2:767397762724:dlc-updates`**。
   b. 对于协议，请从 [Amazon SQS、Lambda AWS 、电子邮件、 email-JSON] 中选择一个
   c. 对于端点，输入您将用于接收通知的资源的电子邮件地址或 Amazon 资源名称（ARN）。
   d. 选择创建订阅。
5. 您将收到一封主题行为 AWS 通知 - 订阅确认 的确认电子邮件。打开该电子邮件，选择确认订阅来完成订阅。

支持策略

AWS Dee@@ p Learning Containers (DLCs) 简化了深度学习工作负载的图像配置，并使用最新的框架、硬件、驱动程序、库和操作系统进行了优化。本页详细介绍了的框架支持政策 DLCs。

支持的框架

参考以下 Dee AWS p Learning Containers Framework 支持策略表，查看哪些框架和版本受到积极支持。

请参阅补丁结束，以查看对于由原始框架维护团队主动支持的当前版本， AWS 可以支持多长时间。框架和版本以单框架 DLCs形式提供。

> **Note**
>
> 在框架版本 x.y.z 中，x 表示主要版本，y 表示次要版本，z 表示补丁版本。例如，对于 TensorFlow 2.6.5，主版本为 2，次要版本为 6，补丁版本为 5。

常见问题

- 哪些框架版本会获得安全补丁？
- AWS 发布新框架版本时会发布哪些镜像？
- 哪些图像获得了新的 SageMaker AI/AWS 功能？
- “支持的框架”表中是如何定义当前版本的？
- 如果我运行的版本不在“支持的框架”表中，该怎么办？
- 是否 DLCs 支持以前版本的 TensorFlow？
- 如何找到支持的框架版本的最新补丁映像？
- 多长时间发布一次新映像？
- 运行工作负载时，能在我的实例上以替代方式安装补丁吗？
- 如果有新的补丁或更新的框架版本可用，会发生什么呢？
- 是否可在不更改框架版本的情况下更新依赖项？
- 对我的框架版本的主动支持何时结束？
- 对于框架版本不再主动维护的映像，会为其安装补丁吗？

- 如何使用旧框架版本？
- 如何保持框架及其版本 up-to-date的支持变更？
- 是否需要商业许可证才能使用 Anaconda 存储库？

哪些框架版本会获得安全补丁？

如果框架版本在 Dee AWS p Learning Containers Framework 支持策略表中标有 “支持”，则它将获得安全补丁。

AWS 发布新框架版本时会发布哪些镜像？

我们会在 TensorFlow 和 PyTorch的新版本发布后 DLCs 不久发布新版本。这包括框架的主要版本、主要的次要版本和 major-minor-patch版本。当新版本的驱动程序和库可用时，我们也会更新映像。有关映像维护的更多信息，请参阅 对我的框架版本的主动支持何时结束？。

哪些图像获得了新的 SageMaker AI/AWS 功能？

新功能通常在最新版本的 for DLCs PyTorch 和中发布 TensorFlow。有关新 SageMaker AI 或 AWS 功能的详细信息，请参阅特定图像的发行说明。有关可用容器的列表 DLCs，请参阅 Dee AWS p Learning Containers 的发行说明。有关映像维护的更多信息，请参阅 对我的框架版本的主动支持何时结束？。

“支持的框架”表中是如何定义当前版本的？

Dee AWS p Learning Containers Framework Support Policy 表中的当前版本是 AWS 指在上发布的最新框架版本 GitHub。每个最新版本都包含对 DLC 中驱动程序、库和相关软件包的更新。有关映像维护的信息，请参阅 对我的框架版本的主动支持何时结束？

如果我运行的版本不在“支持的框架”表中，该怎么办？

如果您运行的版本不在 Dee AWS p Learning Containers Framework 支持策略表中，则可能没有最新的驱动程序、库和相关包。要获得更多 up-to-date版本，我们建议您使用您选择的最新 DLC 升级到支持的框架之一。有关可用容器的列表 DLCs，请参阅 Dee AWS p Learning Containers 的发行说明。

是否 DLCs 支持以前版本的 TensorFlow？

不是。 我们支持每个框架最新主要版本的最新补丁版本，如AWS 深度学习容器框架支持政策表中所述，自首次 GitHub 发布之日起 365 天内发布。有关更多信息，请参阅 如果我运行的版本不在“支持的框架”表中，该怎么办？。

如何找到支持的框架版本的最新补丁映像？

要使用最新框架版本的 DLC，请浏览 DLC GitHub 版本标签以找到您选择的示例图像 URI，然后使用它来提取最新的可用的 Docker 镜像。您选择的框架版本必须在 Dee AWS p Learn in g Containers 框架支持策略表中标记为支持。

多长时间发布一次新映像？

提供更新的补丁版本是我们的首要任务。我们通常会尽早创建安装了补丁的映像。我们会监控新修补的框架版本（例如 TensorFlow 2.9 到 TensorFlow 2.9.1）和新的次要发行版本（例如 TensorFlow 2.9 到 TensorFlow 2.10），并尽早提供它们。当现有版本与新版本 TensorFlow 的 CUDA 一起发布时，我们会为该版本发布一个支持新 CUDA 版本的 TensorFlow 新 DLC。

运行工作负载时，能在我的实例上以替代方式安装补丁吗？

不是。 DLC 的补丁更新不是 “就地” 更新。

您必须删除实例上的现有镜像，并在不终止实例的情况下提取最新的容器镜像。

如果有新的补丁或更新的框架版本可用，会发生什么呢？

请定期查看发布说明页面以获取您的映像。我们鼓励您在新的补丁或更新的框架可用时将框架升级。有关可用容器的列表 DLCs，请参阅 Dee AWS p Learning Containers 的发行说明。

是否可在不更改框架版本的情况下更新依赖项？

我们在不更改框架版本的情况下更新依赖项。但是，如果依赖项更新导致不兼容，我们就会创建不同版本的映像。请务必查看 Dee AWS p Learning Containers 的发行说明，了解更新的依赖项信息。

对我的框架版本的主动支持何时结束？

DLC 图像是不可变的。一旦创建，就不会改变。结束对框架版本的主动支持涉及四个主要原因：

- 框架版本（补丁）升级
- AWS 安全补丁
- 补丁结束日期（已过期）
- 依赖关系 end-of-support

> **Note**
>
> 由于版本补丁升级和安全补丁的频率很高，我们建议您经常查看 DLC 的发行说明页面，并在进行更改时进行升级。

框架版本（补丁）升级

如果你有基于 TensorFlow 2.7.0 的 DLC 工作负载并且在 2.7.1 版本之后 TensorFlow发布 GitHub，那么就要发布一个 2.7.1 AWS 版本的新 DLC。 TensorFlow 2.7. TensorFlow 1 版本的新图像发布后，之前版本为 2.7.0 的图像将更长时间保持活跃。 TensorFlow 2.7.0 版本的 DLC 不会收到更多补丁。然后， TensorFlow 2.7 的 DLC 发行说明页面将使用最新信息进行更新。没有为每个次要补丁提供单独的发布说明页面。

由于补丁升级而 DLCs 创建的新版本标有更新的发行标签。如果更改不向后兼容，则标签将更改主要版本而不是次要版本（例如，v1.0 将更改为 v2.0 而不是 v 1.2）。

AWS 安全补丁

如果你的工作负载基于 TensorFlow 2.7.0 版本的镜像并 AWS 制作了安全补丁，那么将发布适用于 2.7.0 的 DLC 的新版本。 TensorFlow TensorFlow 2.7.0 版图像的先前版本已不再活跃维护。有关更多信息，请参阅运行工作负载时，能在我的实例上以替代方式安装补丁吗？有关查找最新 DLC 的步骤，请参阅 如何找到支持的框架版本的最新补丁映像？

由于补丁升级而 DLCs 创建的新版本标有更新的发行标签。如果更改不向后兼容，则标签将更改主要版本而不是次要版本（例如，v1.0 将更改为 v2.0 而不是 v 1.2）。

补丁结束日期（已过期）

DLCs 在 GitHub 发布日期 365 天后，他们的补丁结束日期。

> **Important**
>
> 当有重大框架更新时，我们会例外处理。例如。如果 TensorFlow 1.15 更新到 TensorFlow 2.0，那么我们将在自 GitHub 发布之日起两年内继续支持最新版本的 TensorFlow 1.15，或者在 Origin 框架维护团队取消支持后的六个月内（以较早的日期为准）。

依赖关系 end-of-support

如果你正在使用 Python 3.6 在 TensorFlow 2.7.0 的 DLC 镜像上运行工作负载，并且该版本的 Python 已标记 end-of-support，那么所有基于 Python 3.6 的 DLC 镜像都将不再被主动维护。同样，如果标记了像 Ubuntu 16.04 这样的操作系统版本 end-of-support，则所有依赖于 Ubuntu 16.04 的 DLC 镜像都将不再被主动维护。

对于框架版本不再主动维护的映像，会为其安装补丁吗？

不会。不再主动维护的图像就不会有新版本。

如何使用旧框架版本？

要使用带有旧框架版本的 DLC，请浏览 DLC GitHub 版本标签以找到您选择的图像 URI，然后使用它来提取 docker 镜像。

如何保持框架及其版本 up-to-date的支持变更？

up-to-date使用 DLC 发行说明和可用的 Deep Learning Containers Im ages 页面继续关注 DLC 框架和版本。

是否需要商业许可证才能使用 Anaconda 存储库？

Anaconda 转向了针对某些用户的商业许可模式。积极维护 DLCs 已从Anaconda频道迁移到公开可用的开源版本的Conda（conda-forge）。

> **Warning**
>
> 如果您正在积极使用 Anaconda 在不再积极维护的 DLC 中安装和管理您的软件包及其依赖关系，那么如果您确定这些条款适用于您，则您有责任遵守 Anaconda Reposit ory 中的管理许可。或者，您可以迁移到 Dee AWS p Learning Containers Framework 支持策略表中 DLCs 列出的当前支持的软件之一，也可以使用 conda-forge 作为源代码安装软件包。

框架 Support 政策表

有关更多详细信息，请参阅 Framewor k Support 政策。

支持的框架版本

这些容器可在以下地区使用：

框架	当前版本	CUDA 版本	GitHub GA	补丁结束
PyTorch	2.7	12.8	2025-04-23	2026-04-23
PyTorch	2.6	• 12.6 用于训练 DLCs • 12.4 用于推理 DLCs	2025-01-29	2026-01-29
PyTorch	2.5	12.4	2024-10-29	2025-10-29
PyTorch	2.4	12.4	2024-07-24	2025-07-24
TensorFlow	2.18	• 12.5 用于训练 DLCs • 12.2 用于推理 DLCs	2025-01-24	2026-01-24

不支持的框架版本

此表中列出的版本将在其支持日期之后出现 2 年。

框架	当前版本	CUDA 版本	GitHub GA	补丁结束
PyTorch	2.3	12.1	2024-04-24	2025-04-24
PyTorch	2.2	• 12.1 用于训练 DLCs	2024-01-30	2025-01-30

框架	当前版本	CUDA 版本	GitHub GA	补丁结束
		• 11.8 用于推理 DLCs		
PyTorch	2.1	• 12.1 用于训练 DLCs • 11.8 用于推理 DLCs	2023-10-04	2024-10-04
PyTorch	2.0	• 12.1 适用于训练 DLC • 11.8 用于训练和推理 DLCs	2023-03-15	2024-03-15
PyTorch	1.13	11.7	2022-10-28	2024-10-28
PyTorch	1.12	• 11.3 适用于 SageMaker DLC • 11.6 适用于 EC2 /ECS/ EKS DLC	2022-07-01	2023-07-01
PyTorch	1.11	• 11.3 适用于 SageMaker DLC • /ECS EC2 / EKS DLC 为 11.5	2022-03-10	2023-03-10
TensorFlow	2.16	• 12.3 用于训练 DLCs • 12.2 用于推理 DLCs	2024-03-07	2025-03-07
TensorFlow	2.14	11.8	2023-09-26	2024-09-26

框架	当前版本	CUDA 版本	GitHub GA	补丁结束
TensorFlow	2.13	11.8	2023-07-19	2024-07-19
TensorFlow	2.12	11.8	2023-03-23	2024-03-23
TensorFlow	2.1.1	11.2	2022-11-18	2023-11-18
TensorFlow	2.10	11.2	2022-09-06	2023-09-06
TensorFlow	2.9	11.2	2022-05-17	2023-05-17
TensorFlow	2.8	11.2	2022-02-02	2023-02-02
TensorFlow	2.7	11.2	2021-11-04	2022-11-04

更新

- 2024-08-14 将版本格式从 major.minor.patch 更新为 major.minor 以减少混乱。
- 2023-10-30 将对 PT 1.13 的支持延长一年，因为它是发布的最后一个 1.x 版本。 PyTorch
- 2023-10-01 终止对和 Elastic Inference 的 MXNet 支持

不支持的框架

从 2023 年 10 月 1 日起 MXNet，Elastic Inference 框架将不再受支持，因为这些框架缺乏上游支持。请按照本文档中 “支持的框架版本” 部分使用我们支持的框架之一。

AWS 深度学习 Containers 中的安全

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Deep Learning Containers 的合规计划，请参阅AWS 按合规计划划分的范围内。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 Deep Learning Containers 时如何应用分担责任模型。以下主题向您展示了如何配置 Deep Learning Containers 以实现您的安全和合规目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Deep Learning Containers 资源。

有关更多信息，请参阅亚马逊安全 EC2、亚马逊 ECS 中的安全、亚马逊 EKS 中的安全以及亚马逊的安全 SageMaker。

主题

- AWS 深度学习 Containers 中的数据保护
- AWS 深度学习容器中的身份和访问管理 Deep Learning Containers
- Dee AWS p Learning Containers 中的监控和使用跟踪
- AWS 深度学习 Containers 的合规性验证
- AWS 深度学习 Containers 中的弹性
- AWS 深度学习 Containers 中的基础设施安全

AWS 深度学习 Containers 中的数据保护

分担责任模型 AWS 分担责任模型适用于Dee AWS p Learning Containers中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS 云。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，

请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括你使用控制台、API 或 AWS 服务 使用深度学习 Containers 或其他容器时 AWS SDKs。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

AWS 深度学习容器中的身份和访问管理 Deep Learning Containers

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用深度学习容器资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

有关身份和访问管理的更多信息，请参阅亚马逊的身份和访问管理、亚马逊 ECS 的身份和访问管理 EC2、亚马逊 EKS 的身份和访问管理，以及亚马逊的身份和访问管理，以及亚马逊的身份和访问管理 SageMaker。

主题

- 使用身份进行身份验证

- 使用策略管理访问
- IAM 与 Amazon EMR 结合使用

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的用于签署 API 请求的AWS 签名版本 4。

无论使用何种身份验证方法，您都可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的 IAM 中的AWS 多重身份验证。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的用例，应在需要时更新访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并向该群组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 用户的使用案例。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。要在中临时担任 IAM 角色 AWS Management Console，您可以从用户切换到 IAM 角色（控制台）。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的代入角色的方法。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问：要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM 用户指南》中的针对第三方身份提供商创建角色（联合身份验证）。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限：IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取：您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南中的 IAM 中的跨账户资源访问。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序 EC2 或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
- 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 A@@ mazon 上运行的应用程序 EC2 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这比在 EC2 实例中存储访问密钥更可取。要为 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在 EC2 实例上运行的程序获得临时证书。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色向在 A mazon EC2 实例上运行的应用程序授予权限。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户托管策略定义自定义 IAM 权限。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括

AWS 托管策略和客户托管策略。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持的服务示例 ACLs。 AWS WAF要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs) — SCPs 是 JSON 策略，用于指定中组织或组织单位 (OU) 的最大权限 AWS Organizations。 AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organization SCPs s 和的更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs) — RCPs 是 JSON 策略，您可以使用它来设置账户中资源的最大可用权限，而无需更新附加到您拥有的每个资源的 IAM 策略。RCP 限制成员账户中资源的权限，并可能影响身份（包括身份）的有效权限 AWS 账户根用户，无论这些身份是否属于您的组织。

有关 Organizations 的更多信息 RCPs，包括 AWS 服务 该支持的列表 RCPs，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。

- 会话策略：会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

IAM 与 Amazon EMR 结合使用

您可以 AWS Identity and Access Management 与 Amazon EMR 一起使用来定义用户、 AWS 资源、群组、角色和策略。您还可以控制这些用户和角色可以访问哪些 AWS 服务。

有关将 IAM 与 Amazon EMR 结合使用的更多信息，请参阅 Amazon EMR 的AWS Identity and Access Management。

Dee AWS p Learning Containers 中的监控和使用跟踪

你的 D AWS eep Learning Containers 不附带监控工具。有关监控的信息，请参阅 GPU 监控和优化、监控亚马逊、监控 Amazon ECS EC2、监控 Amaz on EKS 和监控 A mazon SageMaker Studio。

使用情况跟踪

AWS 使用客户反馈和使用信息来提高我们向客户提供的服务和软件的质量。我们在支持的 Dee AWS p Learning Containers 中增加了使用数据收集功能，以便更好地了解客户的使用情况并指导未来的改进。默认情况下，Deep Learning Containers 的使用情况跟踪处于激活状态。客户可以随时更改其设置，以激活或停用使用情况跟踪。

Dee AWS p Learning Containers 的使用情况跟踪会收集用于容器的实例 ID、框架、框架版本、容器类型和 Python 版本。 AWS 还会记录它接收此元数据的事件时间。

不会收集或保留有关容器内使用的命令的信息。不会收集或保留有关容器的其他信息。

要选择退出使用情况跟踪，请将`OPT_OUT_TRACKING`环境变量设置为 true。

```
OPT_OUT_TRACKING=true
```

故障率跟踪

使用第一方 Dee Amazon SageMaker AI AWS p Learning Containers 容器时， SageMaker AI 团队将收集故障率元数据以提高 AWS 深度学习容器的质量。默认情况下，Dee AWS p Learning Containers 的故障率跟踪处于活动状态。客户可以在创建 Amazon SageMaker AI 端点时更改其设置以激活或停用故障率跟踪。

Dee AWS p Learning Containers 的故障率跟踪会收集实例 ID、ModelServer 名称ErrorType、ModelServer 版本和ErrorCode。 AWS 还会记录它接收此元数据的事件时间。

不会收集或保留有关容器内使用的命令的信息。不会收集或保留有关容器的其他信息。

要选择退出故障率跟踪，请将`OPT_OUT_TRACKING`环境变量设置为`true`。

```
OPT_OUT_TRACKING=true
```

以下框架版本中的使用情况跟踪

虽然我们建议更新到支持的 Deep Learning Containers，但要选择退出使用这些框架的 Deep Learning Containers 的使用情况跟踪，请将`OPT_OUT_TRACKING`环境变量设置为 true，然后使用自定义入口点来禁用对以下服务的调用：

- Amazon EC2 自定义入口点
- Amazon ECS 自定义入口点
- 亚马逊 EKS 自定义入口点

AWS 深度学习 Containers 的合规性验证

作为多个合规计划的一部分，第三方审计师对服务的安全性和 AWS 合规性进行评估。有关支持的合规计划的信息，请参阅亚马逊合规验证 EC2、Amazon ECS 合规验证、Ama zon E KS 合规验证和亚马逊合规验证 SageMaker。

有关特定合规计划范围内的 AWS 服务列表，请参阅按合规计划划分的范围内的AWSAWS 服务按合规计划。有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅在 Artifact 中下载报告在 AWS Ar 。

您在使用 Deep Learning Containers 时的合规责任取决于您的数据的敏感度、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全性与合规性快速入门指南 - 这些部署指南讨论了架构注意事项，并提供了在 AWS上部署基于安全性和合规性的基准环境的步骤。
- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 此 AWS 服务可全面了解您的安全状态 AWS ，帮助您检查是否符合安全行业标准和最佳实践。

AWS 深度学习 Containers 中的弹性

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。 AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础结构相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

有关有助于支持您的数据弹性和备份需求的功能的信息，请参阅 Amazon 中的弹性 EC2、Amazon EKS 中的弹性和亚马逊的弹性。 SageMaker

AWS 深度学习 Containers 中的基础设施安全

Dee AWS p Learning Containers 的基础设施安全由亚马逊 EC2、亚马逊 ECS、Amazon EKS 或 SageMaker AI 提供支持。有关更多信息，请参阅 Amazon 中的基础设施安全 EC2、Ama zon ECS 中的基础设施安全、Amazon E KS 中的基础设施安全和亚马逊的基础设施安全 SageMaker。

Deep Learning Containers 文档历史记录开发者指南

下表描述了此版本的 Deep Learning Containers 的文档。

- API 版本：最新
- 最新文档更新：2020 年 2 月 26 日

变更	说明	日期
《深度学习容器》开发者指南发布会	开发者指南中添加了 Deep Learning Containers 设置和教程。	2020 年 2 月 17 日

AWS 词汇表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。