



Guia do Desenvolvedor

AMIs de deep learning da AWS



AMIs de deep learning da AWS: Guia do Desenvolvedor

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

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é a DLAMI?	1
Sobre este Guia	1
Pré-requisitos	1
Exemplo de casos de uso	1
Atributos	2
Frameworks pré-instalados	2
Software de GPU pré-instalado	3
Fornecimento e visualização de modelos	3
Notas de lançamento do DLAMI	4
DLAMI compatível com P6-B200	4
DLAMI compatível com P6-B200	4
Teste de funcionalidade da GPU	5
Base DLAMIs	8
X86	8
ARM64	63
Estrutura única DLAMIs	80
PyTorch DLAMIs	80
TensorFlow DLAMIs	125
DLAMI de várias estruturas	134
X86	135
Conceitos básicos	149
Escolher uma DLAMI	149
Instalações do CUDA e associações da estrutura	150
Base	151
Conda	151
Arquitetura	153
SO	153
Escolher uma instância	154
Preços	155
Disponibilidade de regiões	155
GPU	156
CPU	157
Inferentia	157
Trainium	158

Configurar	159
Encontrar o ID da DLAMI	159
Executar uma instância do	161
Conectar a uma instância	163
Configurar o Jupyter	163
Proteger o servidor	164
Iniciar o servidor	165
Conectar cliente	165
Fazer login	167
Limpeza	169
Utilizar uma DLAMI	171
DLAMI do Conda	171
Introdução à AMI de aprendizado profundo com Conda	171
Faça login na DLAMI	172
Inicie o TensorFlow meio ambiente	172
Mude para o PyTorch ambiente Python 3	173
Remoção de ambientes	174
DLAMI base	174
Uso da AMI base de aprendizado profundo	174
Configurar as versões do CUDA	174
Notebooks Jupyter	175
Navegar pelos tutoriais instalados	176
Alternar ambientes com o Jupyter	176
Tutoriais	177
Ativar estruturas	177
Elastic Fabric Adapter	180
Monitoramento e otimização de GPU	194
AWS Inferência	204
ARM64 DLAMI	226
Inferência	229
Fornecimento de modelos	230
Como atualizar a DLAMI	234
Atualização da DLAMI	234
Atualizações de software	235
Notificações de lançamento	236
Segurança	238

Proteção de dados	239
Gerenciamento de identidade e acesso	240
Autenticação com identidades	240
Gerenciar o acesso usando políticas	243
IAM com o Amazon EMR	246
Validação de conformidade	246
Resiliência	247
Segurança da infraestrutura	248
Monitoramento	248
Rastreamento de uso	248
Política de suporte da DLAMI	250
Suporte do DLAMI FAQs	250
Quais versões da estrutura recebem patches de segurança?	251
Qual sistema operacional recebe patches de segurança?	251
Quais imagens são AWS publicadas quando novas versões do framework são lançadas? ..	251
Quais imagens recebem novos AWS recursos de SageMaker IA/I?	251
Como a versão atual é definida na tabela de Estruturas compatíveis?	251
E se eu estiver executando uma versão que não está na tabela suportada?	252
Oferecem DLAMIs suporte às versões anteriores de patch de uma versão do Framework?	252
Como posso encontrar a imagem com patch aplicado mais recente de uma versão compatível da estrutura?	252
Com que frequência novas imagens são lançadas?	252
Minha instância receberá um patch enquanto a workload estiver em execução?	253
O que acontece quando uma nova versão com patch aplicado ou atualizada da estrutura está disponível?	253
As dependências são atualizadas sem alterar a versão da estrutura?	253
Quando o suporte ativo para minha versão da estrutura termina?	253
As imagens com versões de estrutura que não são mais mantidas ativamente receberão um patch?	255
Como usar uma versão de estrutura mais antiga?	255
Como faço para up-to-date acompanhar as mudanças de suporte nas estruturas e suas versões?	255
Preciso de uma licença comercial para usar o repositório Anaconda?	255
Tabela de políticas de suporte do DLAMI	256
Versões de estrutura suportadas	256

Versões do sistema operacional suportadas	256
Versões de estrutura não suportadas	256
Versões de sistema operacional não suportadas	257
Arquivo de notas de lançamento do DLAMI não suportado	258
Base	258
Estrutura única	258
Estrutura múltipla	260
Alterações importantes	261
Alteração do driver DLAMI NVIDIA FAQs	261
O que mudou?	261
Por que essa alteração foi necessária?	262
O DLAMIs que essa mudança afetou?	263
O que isso significa para você?	263
Há alguma perda de funcionalidade com o mais novo DLAMIs?	263
Essa mudança afetou os contêineres de aprendizado profundo?	264
Informações relacionadas	265
Recursos descontinuados	266
Histórico de documentos	268
.....	cclxxi

O que AMIs de deep learning da AWSé

AMIs de deep learning da AWS (DLAMI) fornece imagens de máquina personalizadas que você pode usar para aprendizado profundo na nuvem. Eles DLAMIs estão disponíveis na maioria Regiões da AWS para uma variedade de tipos de instância do Amazon Elastic Compute Cloud (Amazon EC2), desde uma pequena instância somente de CPU até as mais recentes instâncias de alta potência com várias GPUs. Eles DLAMIs vêm pré-configurados com [NVIDIA CUDA](#) e NVIDIA [cuDNN](#) e as versões mais recentes das estruturas de aprendizado profundo mais populares.

Sobre este Guia

O conteúdo do pode ajudá-lo a lançar e usar DLAMIs o. O guia abrange vários casos de uso comuns de aprendizado profundo para treinamento e inferência. Ele também mostra como escolher a AMI correta para o objetivo e o tipo de instância de sua preferência.

Além disso, DLAMIs incluem vários tutoriais fornecidos por suas estruturas suportadas. Este guia pode mostrar como ativar cada framework e encontrar os tutoriais apropriados para começar. Ele também tem tutoriais sobre treinamento distribuído, depuração, uso de AWS Inferentia e AWS Trainium e outros conceitos-chave. Para conferir instruções sobre como configurar um servidor de cadernos Jupyter para executar os tutoriais no navegador, consulte [Configurar um servidor de cadernos Jupyter em uma instância de DLAMI](#).

Pré-requisitos

Para executar com sucesso o DLAMIs, recomendamos que você esteja familiarizado com as ferramentas de linha de comando e o Python básico.

Exemplos de casos de uso de DLAMIs

Veja a seguir exemplos de alguns casos de uso comuns para AMIs de deep learning da AWS (DLAMI).

Conhecer o aprendizado profundo: a DLAMI é uma ótima escolha para o aprendizado ou o ensino de frameworks de machine learning e aprendizado profundo. Eles DLAMIs eliminam a dor de cabeça de solucionar problemas nas instalações de cada estrutura e fazer com que elas funcionem no mesmo computador. Eles DLAMIs incluem um notebook Jupyter e facilitam a execução dos tutoriais que as estruturas fornecem para pessoas novas em aprendizado de máquina e aprendizado profundo.

Desenvolvimento de aplicações: se você trabalha com desenvolvimento de aplicações e tem interesse em usar aprendizado profundo para que suas aplicações usem os últimos avanços de IA, a DLAMI é o campo de teste perfeito para você. Cada estrutura é fornecida com tutoriais sobre como começar a usar o deep learning, e muitas têm vários modelos que facilitam testar o aprendizado profundo sem precisar criar as redes neurais você mesmo ou fazer qualquer treinamento de modelo. Alguns exemplos mostram como criar um aplicativo de detecção de imagem em apenas alguns minutos, ou como criar um aplicativo de reconhecimento de voz para seu próprio chatbot.

Machine Learning e data analytics: se você for cientista de dados ou tiver interesse em processar dados com aprendizado profundo, descobrirá que muitos dos frameworks são compatíveis com R e Spark. Você encontrará tutoriais sobre como fazer regressões simples, até a criação de sistemas de processamento de dados escaláveis para sistemas de personalização e previsões.

Pesquisa — Se você é um pesquisador que deseja experimentar uma nova estrutura, testar um novo modelo ou treinar novos modelos, o DLAMI AWS e os recursos de escala podem aliviar a dor de instalações e gerenciamento tediosos de vários nós de treinamento.

Note

Embora sua escolha inicial seja atualizar seu tipo de instância para uma instância maior com mais GPUs (até 8), você também pode escalar horizontalmente criando um cluster de instâncias DLAMI. Confira [Informações relacionadas sobre a DLAMI](#) para obter mais informações sobre criações de clusters.

Recursos da DLAMI

Os recursos do AMIs de deep learning da AWS (DLAMI) incluem estruturas de aprendizado profundo pré-instaladas, software de GPU, servidores de modelos e ferramentas de visualização de modelos.

Frameworks pré-instalados

No momento, há dois tipos principais de DLAMI com outras variações relacionadas ao sistema operacional (SO) e às versões de software:

- [AMI de aprendizado profundo com Conda](#): frameworks instalados separadamente usando pacotes conda e ambientes Python separados.
- [AMI de deep learning base](#): sem frameworks instalados; apenas [NVIDIA CUDA](#) e outras dependências.

A AMI de aprendizado profundo com Conda usa ambientes conda para isolar cada estrutura de trabalho, para que você possa alternar entre eles conforme sua necessidade e para que não se preocupe com suas dependências conflitantes. A AMI de aprendizado profundo com Conda é compatível com os seguintes frameworks:

- PyTorch
- TensorFlow 2

 Note

O DLAMI não oferece mais suporte às seguintes estruturas de aprendizado profundo: Apache MXNet, Microsoft Cognitive Toolkit (CNTK), Caffe, Caffe2, Theano, Chainer e Keras.

Software de GPU pré-instalado

[Mesmo se você usar uma instância somente de CPU, ela DLAMIs terá NVIDIA CUDA e NVIDIA cuDNN.](#) O software instalado é o mesmo, independentemente do tipo de instância. Lembre-se de que as ferramentas específicas para GPU só funcionam em instâncias que têm pelo menos uma GPU. Para ter mais informações sobre os tipos de instância, consulte [Escolher o tipo de instância de DLAMI.](#)

Para ter mais informações sobre o CUDA, consulte [Instalações do CUDA e associações da estrutura.](#)

Fornecimento e visualização de modelos

A AMI de aprendizado profundo com Conda vem pré-instalada com servidores de modelos para TensorFlow, bem como TensorBoard para visualizações de modelos. Para obter mais informações, consulte [TensorFlow Servindo.](#)

Notas de AMIs lançamento do Deep Learning

Aqui você pode encontrar notas de lançamento detalhadas para todas as opções atualmente suportadas AMIs de deep learning da AWS (DLAMI).

Para ver as notas de lançamento dos frameworks de DLAMI que não são mais compatíveis, consulte a seção [Unsupported Framework Release Notes Archive](#) da página [DLAMI Framework Support Policy](#).

Note

Eles AMIs de deep learning da AWS têm uma cadência de lançamento noturno de patches de segurança. Não incluímos esses patches de segurança incrementais nas notas de lançamento.

Notas da versão

- [DLAMI compatível com P6-B200](#)
- [Notas de lançamento do Base DLAMIs](#)
- [Notas de lançamento do Single Framework DLAMIs](#)
- [Notas de lançamento do Multi-Framework DLAMIs](#)

DLAMI compatível com P6-B200

Abaixo estão os requisitos detalhados para executar o DLAMI em

Compatível com P6-B200 DLAMIs

As seguintes DLAMI oferecem suporte a instâncias P6-B200:

- [AWS AMI da Base de Aprendizado Profundo \(Amazon Linux 2023\)](#)
- [AWS AMI de base de aprendizado profundo \(Ubuntu 24.04\)](#)
- [AWS AMI de base de aprendizado profundo \(Ubuntu 22.04\)](#)

Esses DLAMI contêm o seguinte software necessário para operar instâncias P6-B200:

Software	Requisito mínimo de versão
Kit de ferramentas Nvidia CUDA	12,8
Controlador Nvidia	R570
NVLINK 5	R570
Kernel Linux	6.1
Elastic Fabric Adapter (EFA)	1.41.0
AWS Plug-in OFI NCCL	1.15.0

Confirme a funcionalidade da GPU

Para confirmar o funcionamento GPUs:

1. Execute o seguinte teste de consulta de dispositivo de GPU Nvidia

```
$ /usr/local/cuda/extras/demo_suite/deviceQuery
```

2. Confirme a seguinte saída do Device Query Run:

```
$ /usr/local/cuda/extras/demo_suite/deviceQuery
/usr/local/cuda/extras/demo_suite/deviceQuery Starting...

CUDA Device Query (Runtime API)

Detected 8 CUDA Capable device(s)
...
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.8, CUDA Runtime Version
= 12.8, NumDevs = 8, Device0 = NVIDIA B200, Device1 = NVIDIA B200, Device2 =
NVIDIA B200, Device3 = NVIDIA B200, Device4 = NVIDIA B200, Device5 = NVIDIA B200,
Device6 = NVIDIA B200, Device7 = NVIDIA B200
Result = PASS
```

Para confirmar o funcionamento do driver NVIDIA:

1. Execute a interface de gerenciamento do sistema Nvidia

```
$ nvidia-smi
```

2. Confirme a seguinte saída da interface de gerenciamento do sistema

```
+-----+
+
| NVIDIA-SMI 570.133.20              Driver Version: 570.133.20      CUDA Version:
12.8          |
|-----+-----+
+-----+
| GPU  Name                Persistence-M | Bus-Id        Disp.A | Volatile
Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |      Memory-Usage | GPU-Util
Compute M.  |
|                                           |                      |
MIG M.      |
|=====+=====|
+=====+
|   0   NVIDIA B200                 Off |  00000000:51:00.0 Off |
   0 |
| N/A   32C    P0               145W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                           |                      |
Disabled |
+-----+-----+
+-----+
|   1   NVIDIA B200                 Off |  00000000:52:00.0 Off |
   0 |
| N/A   30C    P0               140W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                           |                      |
Disabled |
+-----+-----+
+-----+
|   2   NVIDIA B200                 Off |  00000000:62:00.0 Off |
   0 |
| N/A   31C    P0               139W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                           |                      |
Disabled |
```

```

+-----+-----+
+-----+
|  3  NVIDIA B200                                Off | 00000000:63:00.0 Off |
|  0 |
| N/A   29C   P0                                139W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                     |
Disabled |
+-----+-----+
+-----+
|  4  NVIDIA B200                                Off | 00000000:75:00.0 Off |
|  0 |
| N/A   31C   P0                                141W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                     |
Disabled |
+-----+-----+
+-----+
|  5  NVIDIA B200                                Off | 00000000:76:00.0 Off |
|  0 |
| N/A   31C   P0                                141W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                     |
Disabled |
+-----+-----+
+-----+
|  6  NVIDIA B200                                Off | 00000000:86:00.0 Off |
|  0 |
| N/A   32C   P0                                141W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                     |
Disabled |
+-----+-----+
+-----+
|  7  NVIDIA B200                                Off | 00000000:87:00.0 Off |
|  0 |
| N/A   30C   P0                                138W / 1000W |      0MiB / 183359MiB |      0%
Default |
|                                     |
Disabled |
+-----+-----+
+-----+

```

```

+-----+
+
| Processes:
|
| GPU  GI  CI          PID  Type  Process name                      GPU
| Memory |
|      ID  ID
| Usage   |
|
=====
| No running processes found
|
+-----+
+

```

Se você tiver algum problema com as instâncias P6-B200, entre em contato com o Support. AWS

Notas de lançamento do Base DLAMIs

Notas de lançamento do X86 Base DLAMI

- [Notas de lançamento do X86 Base DLAMI](#)
- [ARM64 Notas de lançamento do DLAMI básico](#)

Notas de lançamento do X86 Base DLAMI

Abaixo estão as notas de lançamento do X86 Base DLAMI:

GPU

- [AWS AMI de base de aprendizado profundo \(Amazon Linux 2023\) \(compatível com P6-B200\)](#)
- [AWS AMI de base de aprendizado profundo \(Ubuntu 24.04\) \(suporta P6-B200\)](#)
- [AWS AMI de base de aprendizado profundo \(Ubuntu 22.04\) \(suporta P6-B200\)](#)
- [AWS AMI de base de aprendizado profundo \(Amazon Linux 2\)](#)

Qualcomm

- [AWS Base de aprendizado profundo Qualcomm AMI \(Amazon Linux 2\)](#)

AWS Neuron

- Consulte o Guia do usuário do [Neuron DLAMI](#).

AWS AMI de GPU de base de aprendizado profundo (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Base de aprendizado profundo OSS Nvidia Driver GPU AMI (Amazon Linux 2023) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#)
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en, P6-B200

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: x86
- A versão mais recente disponível está instalada para os seguintes pacotes:
 - Kernel Linux: 6.1
 - FSx Lustre
 - NVIDIA GDS
 - Docker
 - AWS CLI v2 em/usr/local/bin/aws2 e AWS CLI v1 em/usr/bin/aws
 - NVIDIA DCGM
 - Kit de ferramentas de contêiner Nvidia:
 - Comando de versão: nvidia-container-cli -V
 - NVIDIA-Docker 2:
 - Comando de versão: versão nvidia-docker

- Driver NVIDIA: 570.133.20
- Pilha NVIDIA CUDA 12.4-12.6 e 12.8:
 - Diretórios de instalação CUDA, NCCL e cuDDN: /usr/local/cuda
 - Exemplo: /usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
 - Versão compilada da NCCL: 2.26.5
 - CUDA padrão: 12,8
 - PATH/usr/local/cudaaponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.4/targets/x86_64-linux/lib
 - PATH para ter/usr/local/cuda-12.8/bin:/usr/local/cuda-12.8/include/
 - Para qualquer versão diferente do CUDA, atualize LD_LIBRARY_PATH adequadamente.
- Instalador EFA: 1.40.0
- Nvidia GDRCopy: 2.5.1
- AWS O plugin OFI NCCL vem com o instalador EFA
 - Os caminhos /opt/amazon/ofi-nccl/lib and /opt/amazon/ofi-nccl/efasão adicionados ao LD_LIBRARY_PATH.
- AWS CLI v2 em /usr/local/bin/aws2 e AWS CLI v1 em /usr/bin/aws
- Tipo de volume do EBS: gp3
- Python: /3.9 usr/bin/python
- NVMe Local do armazenamento de instâncias (em [EC2 instâncias compatíveis](#)): /opt/dlami/nvme
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
SSM_PARAMETER=base-oss-nvidia-driver-gpu-amazon-linux-2023/latest/ami-id \
aws ssm get-parameter --region us-east-1 \
--name /aws/service/deeplearning/ami/x86_64/$SSM_PARAMETER \
--query "Parameter.Value" \
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
```



```
--owners amazon \
--filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver GPU AMI
(Amazon Linux 2023) ????????' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
--output text
```

Avisos

Kit de ferramentas de contêiner NVIDIA 1.17.4

No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Política de suporte

Esses AMIs componentes dessa AMI, como as versões CUDA, podem ser removidos e alterados com base na [política de suporte da estrutura](#) ou para otimizar o desempenho de [contêineres de aprendizado profundo](#) ou para reduzir o tamanho da AMI em uma versão futura, sem aviso prévio. Removemos as versões CUDA AMIs se elas não forem usadas por nenhuma versão de estrutura compatível.

Instâncias P6-B200

As instâncias P6-B200 contêm 8 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
--instance-type $INSTANCETYPE \
--image-id $AMI --key-name $KEYNAME \
--iam-instance-profile "Name=dlami-builder" \
--tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
--network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
```

```

    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=5,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=6,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=7,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"

```

5 instâncias abertas

As instâncias P5en contêm 16 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```

aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    ...
    "NetworkCardIndex=15,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"

```

Instâncias P5/P5e

As instâncias P5 e P5e contêm 32 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```

aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \

```

```
--tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
--network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    ...
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
sudo dnf versionlock kernel*
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
sudo dnf versionlock delete kernel*
sudo dnf update -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-05-15

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250515

Adicionado

- Suporte adicional para instâncias [P6-B200 EC2](#)

Atualizado

- Instalador EFA atualizado da versão 1.38.1 para 1.40.0

- Atualizado GDRCopy da versão 2.4 para 2.5
- Plugin AWS OFI NCCL atualizado da versão 1.13.0-aws para 1.14.2-aws
- Versão NCCL compilada atualizada da versão 2.25.1 para 2.26.5
- Versão CUDA padrão atualizada da versão 12.6 para 12.8
- Versão Nvidia DCGM atualizada de 3.3.9 para 4.4.3

Data de lançamento: 2025-04-22

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250421

Atualizado

- [Driver Nvidia atualizado da versão 570.124.06 para 570.133.20 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de abril de 2025](#)

Data de lançamento: 2025-03-31

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250328

Adicionado

- Suporte adicional para [NVIDIA GPU Direct Storage](#) (GDS)

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 2025-02-15

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-02-05

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250205

Adicionado

- Foi adicionado o kit de ferramentas CUDA versão 12.6 no diretório `/usr/local/cuda`
- Suporte adicional para instâncias G5 EC2

Removido

- As versões 12.1 e 12.2 do CUDA foram removidas deste DLAMI. Os clientes que precisam dessas versões do kit de ferramentas CUDA podem instalá-las diretamente da NVIDIA usando o link abaixo
- <https://developer.nvidia.com/cuda-toolkit-archive>

Data de lançamento: 2025-02-03

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250131

Atualizado

- Versão EFA atualizada de 1.37.0 para 1.38.0
 - O EFA agora inclui o plug-in AWS OFI NCCL, que agora pode ser encontrado em `/ofi-nccl/opt/amazon/ofi-nccl` rather than the original `/opt/aws` Se estiver atualizando sua variável `LD_LIBRARY_PATH`, certifique-se de modificar a localização do OFI NCCL corretamente.
- Kit de ferramentas de contêiner Nvidia atualizado de 1.17.3 para 1.17.4

Data de lançamento: 2025-01-08

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250107

Atualizado

- Suporte adicionado para instâncias [G4dn](#)

Data de lançamento: 2024-12-09

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20241206

Atualizado

- Kit de ferramentas de contêiner Nvidia atualizado da versão 1.17.0 para 1.17.3

Data de lançamento: 2024-11-21

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20241121

Adicionado

- Foi adicionado suporte para instâncias P5en. EC2

Atualizado

- Instalador EFA atualizado da versão 1.35.0 para 1.37.0
- Atualize o plug-in AWS OFI NCCL da versão 1.121-aws para 1.13.0-aws

Data de lançamento: 2024-10-30

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20241030

Adicionado

- Lançamento inicial do Deep Learning Base OSS DLAMI para Amazon Linux 2023

Problemas conhecidos

- Esse DLAMI não oferece suporte às instâncias G4dn e EC2 G5 no momento. AWS está ciente de uma incompatibilidade que pode resultar em falhas de inicialização do CUDA, afetando as famílias de instâncias G4dn e G5 ao usar os drivers NVIDIA de código aberto junto com um kernel Linux versão 6.1 ou mais recente. Esse problema afeta distribuições Linux, como Amazon Linux 2023,

Ubuntu 22.04 ou mais recente, ou SUSE Linux Enterprise Server 15 SP6 ou mais recente, entre outras.

AWS AMI de GPU de base de aprendizado profundo (Ubuntu 24.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Base de aprendizado profundo OSS Nvidia Driver GPU AMI (Ubuntu 24.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en, P6-B200.

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 24.04
- Arquitetura de computação: x86
- A versão mais recente disponível está instalada para os seguintes pacotes:
 - Kernel Linux: 6. 8
 - FSx Lustre
 - Docker
 - AWS CLI v2 em/usr/bin/aws
 - NVIDIA DCGM
 - Kit de ferramentas de contêiner Nvidia:
 - Comando de versão: nvidia-container-cli -V
 - NVIDIA-Docker 2:
 - Comando de versão: versão nvidia-docker
- Driver NVIDIA: 570.133.20
- Pilha NVIDIA CUDA 12.6 e 12.8:

- Diretórios de instalação CUDA, NCCL e cuDDN: /usr/local/cuda
 - Exemplo: /usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
- Versão compilada da NCCL: 2.25.1
- CUDA padrão: 12,8
 - PATH/usr/local/cudaaponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter/64 usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/sbsa-linux/lib:/usr/local/cuda-12.8/nvvm/lib64:/usr/local/cuda-12.8/extras/CUPTI/lib
 - PATH para ter/usr/local/cuda-12.8/bin:/usr/local/cuda-12.8/include/
 - Para qualquer versão diferente do CUDA, atualize LD_LIBRARY_PATH adequadamente.
- Instalador EFA: 1.40.0
- Nvidia GDRCopy: 2.5.1
- AWS O plugin OFI NCCL vem com o instalador EFA
 - Os caminhos/opt/amazon/ofi-nccl/lib and /opt/amazon/ofi-nccl/efasão adicionados ao LD_LIBRARY_PATH.
- AWS CLI v2 em/usr/bin/aws
- Tipo de volume do EBS: gp3
- Python:/3.12 usr/bin/python
- NVMe Local do armazenamento de instâncias (em [EC2 instâncias compatíveis](#)):/opt/dlami/nvme
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
SSM_PARAMETER=base-oss-nvidia-driver-gpu-ubuntu-24.04/latest/ami-id \
aws ssm get-parameter --region us-east-1 \
--name /aws/service/deeplearning/ami/x86_64/$SSM_PARAMETER \
--query "Parameter.Value" \
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
--owners amazon \
```



```
--filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver GPU AMI
(Ubuntu 24.04) ????????' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
--output text
```

Avisos

Política de suporte

Esses AMIs componentes dessa AMI, como as versões CUDA, podem ser removidos e alterados com base na [política de suporte da estrutura](#) ou para otimizar o desempenho de [contêineres de aprendizado profundo](#) ou para reduzir o tamanho da AMI em uma versão futura, sem aviso prévio. Removemos as versões CUDA AMIs se elas não forem usadas por nenhuma versão de estrutura compatível.

EC2 instância com várias placas de rede

- Muitos tipos de instâncias que oferecem suporte ao EFA também têm várias placas de rede.
- DeviceIndex é exclusivo para cada placa de rede e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1.
- Para a interface de rede primária (índice da placa de rede 0, índice do dispositivo 0), crie uma interface EFA (EFA com ENA). Você não pode usar uma interface de rede exclusiva do EFA como interface de rede primária.
- Para cada interface de rede adicional, use o próximo índice de placa de rede não utilizada, o índice de dispositivo 1 e uma interface de rede EFA (EFA com ENA) ou somente EFA, dependendo do seu caso de uso, como requisitos de largura de banda do ENA ou espaço de endereço IP. Por exemplo, casos de uso, consulte a configuração do EFA para instâncias P5.
- Para obter mais informações, consulte o Guia EFA [aqui](#).

Instâncias P6-B200

As instâncias P6-B200 contêm 8 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
```

```
--iam-instance-profile "Name=dlami-builder" \
--tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
--network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=5,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=6,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=7,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

5 instâncias abertas

O P5en contém 16 placas de interface de rede e pode ser iniciado usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
--instance-type $INSTANCETYPE \
--image-id $AMI --key-name $KEYNAME \
--iam-instance-profile "Name=dlami-builder" \
--tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
--network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    ...
    "NetworkCardIndex=15,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Instâncias P5/P5e

As instâncias P5 e P5e contêm 32 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    ...
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
echo linux-aws hold | sudo dpkg --set-selections
echo linux-headers-aws hold | sudo dpkg --set-selections
echo linux-image-aws hold | sudo dpkg --set-selections
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
echo linux-aws install | sudo dpkg --set-selections
echo linux-headers-aws install | sudo dpkg --set-selections
echo linux-image-aws install | sudo dpkg --set-selections
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-05-22

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 24.04) 20250522

Adicionado

- Suporte adicional para instâncias [P6-B200 EC2](#)

Atualizado

- Instalador EFA atualizado da versão 1.40.0 para 1.41.0
- Versão NCCL compilada atualizada da versão 2.25.1 para 2.26.5
- Versão Nvidia DCGM atualizada de 3.3.9 para 4.4.3

Data de lançamento: 2025-05-13

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 24.04) 20250513

Adicionado

- Lançamento inicial do Deep Learning Base OSS DLAMI para Ubuntu 24.04

AWS AMI de GPU de base de aprendizado profundo (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Base de aprendizado profundo OSS Nvidia Driver GPU AMI (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P6-B200.

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: x86

- A versão mais recente disponível está instalada para os seguintes pacotes:
 - Kernel Linux: 6. 8
 - FSx Lustro
 - Docker
 - AWS CLI v2 em/usr/local/bin/aws2 e AWS CLI v1 em/usr/bin/aws
 - NVIDIA DCGM
 - Kit de ferramentas de contêiner Nvidia:
 - Comando de versão: nvidia-container-cli -V
 - NVIDIA-Docker 2:
 - Comando de versão: versão nvidia-docker
- Driver NVIDIA: 570.133.20
- Pilha NVIDIA CUDA 12.4-12.6 e 12.8:
 - Diretórios de instalação CUDA, NCCL e cuDDN:/-xx.x/ usr/local/cuda
 - Exemplo:/usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
 - Versão compilada da NCCL: 2.26.5
 - CUDA padrão: 12,8
 - PATH/usr/local/cudaaponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter/64 usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/x86_64-linux/lib:/usr/local/cuda-12.8/extras/CUPTI/lib
 - PATH para ter/usr/local/cuda-12.8/bin:/usr/local/cuda-12.8/include/
 - Para qualquer versão diferente do CUDA, atualize LD_LIBRARY_PATH adequadamente.
- Instalador EFA: 1.40.0
- Nvidia GDRCopy: 2.5.1
- AWS O plugin OFI NCCL vem com o instalador EFA
 - Os caminhos/opt/amazon/ofi-nccl/lib and /opt/amazon/ofi-nccl/efasão adicionados ao LD_LIBRARY_PATH.
- AWS CLI v2 em/usr/local/bin/aws2 e AWS CLI v1 em/usr/bin/aws
- Tipo de volume EBS: gp3
- Python:/3.10 usr/bin/python
- NVMe Local do armazenamento de instâncias (em [EC2 instâncias compatíveis](#)):/opt/dlami/nvme

- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
- Controlador OSS Nvidia:

```
SSM_PARAMETER=base-oss-nvidia-driver-gpu-ubuntu-22.04/latest/ami-id \
aws ssm get-parameter --region us-east-1 \
--name /aws/service/deeplearning/ami/x86_64/$SSM_PARAMETER \
--query "Parameter.Value" \
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
- Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
--owners amazon \
--filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver GPU AMI
(Ubuntu 22.04) ????????' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
--output text
```

Avisos

Kit de ferramentas de contêiner NVIDIA 1.17.4

No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Atualizações do EFA de 1.37 para 1.38 (lançamento em 31/01/2025)

O EFA agora inclui o plug-in AWS OFI NCCL, que agora pode ser encontrado em `/-ofi-nccl/.opt/amazon/ofi-nccl` rather than the original `/opt/aws`. Se estiver atualizando sua variável LD_LIBRARY_PATH, certifique-se de modificar a localização do OFI NCCL corretamente.

Suporte para vários ENI

- O Ubuntu 22.04 configura e configura automaticamente o roteamento de origem em vários NICs usando o cloud-init em sua inicialização inicial. Se seu fluxo de trabalho incluir attaching/detaching você ENIs enquanto uma instância está parada, uma configuração adicional deve ser adicionada

aos dados do usuário cloud-init para garantir a configuração adequada das NICs durante esses eventos. Uma amostra da configuração da nuvem é fornecida abaixo.

- [Consulte esta documentação da Canonical aqui para obter mais informações sobre como configurar a configuração de nuvem para suas instâncias - /https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automaticallysetup-multiple-nics](https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automaticallysetup-multiple-nics)

```
#cloud-config
# apply network config on every boot and hotplug event
updates:
  network:
    when: ['boot', 'hotplug']
```

Política de suporte

Esses AMIs componentes dessa AMI, como as versões CUDA, podem ser removidos e alterados com base na [política de suporte da estrutura](#) ou para otimizar o desempenho de [contêineres de aprendizado profundo](#) ou para reduzir o tamanho da AMI em uma versão futura, sem aviso prévio. Removemos as versões CUDA AMIs se elas não forem usadas por nenhuma versão de estrutura compatível.

EC2 instâncias com várias placas de rede

- Muitos tipos de instâncias que oferecem suporte ao EFA também têm várias placas de rede.
- DeviceIndex é exclusivo para cada placa de rede e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1.
- Para a interface de rede primária (índice da placa de rede 0, índice do dispositivo 0), crie uma interface EFA (EFA com ENA). Você não pode usar uma interface de rede exclusiva do EFA como interface de rede primária.
- Para cada interface de rede adicional, use o próximo índice de placa de rede não utilizada, o índice de dispositivo 1 e uma interface de rede EFA (EFA com ENA) ou somente EFA, dependendo do seu caso de uso, como requisitos de largura de banda do ENA ou espaço de endereço IP. Por exemplo, casos de uso, consulte a configuração do EFA para instâncias P5.
- Para obter mais informações, consulte o Guia EFA [aqui](#).

Instâncias P6-B200

O P6-B200 contém 8 placas de interface de rede e pode ser iniciado usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=5,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=6,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=7,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

5 instâncias abertas

O P5en contém 16 placas de interface de rede e pode ser iniciado usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    ....
    "NetworkCardIndex=15,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Instâncias P5/P5e

As instâncias P5 e P5e contém 32 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI


```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    ...
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
echo linux-aws hold | sudo dpkg --set-selections
echo linux-headers-aws hold | sudo dpkg --set-selections
echo linux-image-aws hold | sudo dpkg --set-selections
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
echo linux-aws install | sudo dpkg --set-selections
echo linux-headers-aws install | sudo dpkg --set-selections
echo linux-image-aws install | sudo dpkg --set-selections
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-05-16

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250516

Adicionado

- Suporte adicional para instâncias P6-B200 EC2

Atualizado

- Instalador EFA atualizado da versão 1.39.0 para 1.40.0
- Atualize o plug-in AWS OFI NCCL da versão 1.13.0-aws para 1.14.2-aws
- Versão NCCL compilada atualizada da versão 2.22.3 para 2.26.5
- Versão CUDA padrão atualizada da versão 12.6 para 12.8
- Versão Nvidia DCGM atualizada de 3.3.9 para 4.4.3

Data de lançamento: 2025-05-05

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250503

Atualizado

- Atualizado GDRCopy de 2.4.1 para 2.5.1

Data de lançamento: 2025-04-24

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250424

Atualizado

- [O driver Nvidia foi atualizado da versão 570.124.06 para 570.133.20 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de abril de 2025](#)

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250214

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas

bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-02-07

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250205

Adicionado

- Foi adicionado o kit de ferramentas CUDA versão 12.6 no diretório `/usr/local/cuda`

Removido

- As versões 12.1 e 12.2 do CUDA foram removidas deste DLAMI. Os clientes podem instalar essas versões da NVIDIA usando o link abaixo
 - <https://developer.nvidia.com/cuda-toolkit-archive>

Data de lançamento: 2025-01-31

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250131

Atualizado

- Versão EFA atualizada de 1.37.0 para 1.38.0
 - O EFA agora inclui o plug-in AWS OFI NCCL, que agora pode ser encontrado em `/usr/local/ofi-nccl/` rather than the original `/opt/aws`. Se estiver atualizando sua variável `LD_LIBRARY_PATH`, certifique-se de modificar a localização do OFI NCCL corretamente.
- Kit de ferramentas de contêiner Nvidia atualizado de 1.17.3 para 1.17.4

Data de lançamento: 2025-01-17

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250117

Atualizado

- [Driver Nvidia atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025](#)

Data de lançamento: 2024-11-18

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20241115

Adicionado

- FSx Pacote Amazon para suporte ao Lustre adicionado.

Fixo

- Devido a uma alteração no kernel do Ubuntu para solucionar defeitos na funcionalidade KASLR (Kernel Address Space Layout Randomization), as instâncias G4Dn/G5 não conseguem inicializar adequadamente o CUDA no driver OSS Nvidia. Para mitigar esse problema, esse DLAMI inclui uma funcionalidade que carrega dinamicamente o driver proprietário para instâncias G4Dn e G5. Aguarde um breve período de inicialização para esse carregamento para garantir que suas instâncias funcionem corretamente.

Para verificar o status e a integridade desse serviço, você pode usar o seguinte comando:

```
sudo systemctl is-active dynamic_driver_load.service
active
```

Data de lançamento: 2024-10-23

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20241023

Atualizado

- [Driver Nvidia atualizado da versão 550.90.07 para 550.127.05 para endereço CVEs presente no Boletim de Segurança do NVIDIA GPU Display de outubro de 2024](#)

Data de lançamento: 2024-10-01

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 20.04) 20240930

Atualizado

- Driver Nvidia e Fabric Manager atualizados da versão 535.183.01 para 550.90.07
- [Atualizou o Nvidia Container Toolkit da versão 1.16.1 para a 1.16.2, abordando a vulnerabilidade de segurança CVE-2024-0133.](#)
- Versão EFA atualizada de 1.32.0 para 1.34.0
- NCCL atualizado para a versão mais recente 2.22.3 para todas as versões CUDA
 - CUDA 12.1, 12.2 atualizado de 2.18.5+ 2. CUDA12
 - CUDA 12.3 atualizado da versão 2.21.5+ 4. CUDA12

Adicionado

- Foi adicionado o kit de ferramentas CUDA versão 12.4 no diretório /usr/local/cuda
- Foi adicionado suporte para instâncias P5e. EC2

Data de lançamento: 2024-08-19

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240816

Adicionado

- Foi adicionado suporte para a [instância G6e. EC2](#)

Data de lançamento: 2024-06-06

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240606

Atualizado

- Versão atualizada do driver Nvidia para 535.183.01 de 535.161.08

Data de lançamento: 2024-05-15

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240513

Removido

- O suporte do Amazon FSx for Lustre foi removido nesta versão devido à incompatibilidade com as versões mais recentes do kernel Ubuntu 22.04. O suporte FSx para o Lustre será restabelecido assim que a versão mais recente do kernel for suportada. Os clientes que precisam FSx do Lustre devem continuar usando o [Deep Learning Base GPU AMI \(Ubuntu 20.04\)](#).

Data de lançamento: 2024-04-29

Nome da AMI: Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240429

Adicionado

- Lançamento inicial do Deep Learning Base OSS DLAMI para Ubuntu 22.04

AWS AMI de base de aprendizado profundo (Amazon Linux 2)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão \$ {XX.X}
- Versão \$ {XX.X} do driver Nvidia Driver AMI (Amazon Linux 2) da Deep Learning Base

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en
- O Deep Learning com driver proprietário da Nvidia suporta G3 (G3.16x não suportado), P3, P3dn

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2
- Arquitetura de computação: x86
- A versão mais recente disponível está instalada para os seguintes pacotes:

- Kernel Linux: 5.10
- Docker
- AWS CLI v2 em `/usr/local/bin/aws2` e AWS CLI v1 em `/usr/bin/aws`
- Kit de ferramentas de contêiner Nvidia:
 - Comando de versão: `nvidia-container-cli -V`
- NVIDIA-Docker 2:
 - Comando de versão: `versão nvidia-docker`
- Python: `/3.7 usr/bin/python`
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 550.163.01
 - Driver proprietário da Nvidia: 550.163.01
- Pilha NVIDIA CUDA 12.1-12.4:
 - Diretórios de instalação CUDA, NCCL e cuDDN: `/-xx.x/ usr/local/cuda`
 - CUDA padrão: 12.1
 - `PATH/usr/local/cuda` aponta para CUDA 12.1
 - Variáveis de ambiente atualizadas abaixo:
 - `LD_LIBRARY_PATH` para `ter/usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1:/usr/local/cuda-12.1/targets/x86_64-linux/lib`
 - `PATH` para `ter/usr/local/cuda-12.1/bin:/usr/local/cuda-12.1/include/`
 - Para qualquer versão diferente do CUDA, atualize `LD_LIBRARY_PATH` adequadamente.
 - Versão compilada da NCCL: 2.22.3
 - Local dos testes NCCL:
 - `all_reduce, all_gather e reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test`
 - Para executar testes NCCL, o `LD_LIBRARY_PATH` precisa ser aprovado com as atualizações abaixo.
 - `PATHs` Os comuns já foram adicionados ao `LD_LIBRARY_PATH`:
 - `/opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib`
 - Para qualquer versão diferente do CUDA, atualize `LD_LIBRARY_PATH` adequadamente.
 - Instalador EFA: 1.38.0
 - Nvidia GDRCopy: 2.4
 - AWS OFI NCCL: 1.13.2

- AWS O OFI NCCL agora suporta várias versões do NCCL com uma única compilação
- O caminho de instalação: /opt/amazon/ofi-nccl/. Path /opt/amazon/ofi-nccl/lib64 é adicionado ao LD_LIBRARY_PATH.
- Tipo de volume do EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
- Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/x86_64/base-oss-nvidia-driver-amazon-
linux-2/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Driver proprietário da Nvidia:

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/x86_64/base-proprietary-nvidia-driver-
amazon-linux-2/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
- Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon \
  --filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver AMI (Amazon
Linux 2) Version ??.' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```

- Driver proprietário da Nvidia:

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon \
  --filters 'Name=name,Values=Deep Learning Base Proprietary Nvidia Driver AMI
(Amazon Linux 2) Version ??.' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```


Avisos

Kit de ferramentas de contêiner NVIDIA 1.17.4

No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Atualizações do EFA de 1.37 para 1.38 (lançamento em 04/02/2025)

O EFA agora inclui o plug-in AWS OFI NCCL, que agora pode ser encontrado em `/-ofi-nccl/.opt/amazon/ofi-nccl` rather than the original `/opt/aws`. Se estiver atualizando sua variável LD_LIBRARY_PATH, certifique-se de modificar a localização do OFI NCCL corretamente.

Política de suporte

Esses AMIs componentes dessa AMI, como as versões CUDA, podem ser removidos e alterados com base na [política de suporte da estrutura](#) ou para otimizar o desempenho de [contêineres de aprendizado profundo](#) ou para reduzir o tamanho da AMI em uma versão futura, sem aviso prévio. Removemos as versões CUDA AMIs se elas não forem usadas por nenhuma versão de estrutura compatível.

EC2 instâncias com várias placas de rede

- Muitos tipos de instâncias que oferecem suporte ao EFA também têm várias placas de rede.
- DeviceIndex é exclusivo para cada placa de rede e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard. Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1.
- Para a interface de rede primária (índice da placa de rede 0, índice do dispositivo 0), crie uma interface EFA (EFA com ENA). Você não pode usar uma interface de rede exclusiva do EFA como interface de rede primária.
- Para cada interface de rede adicional, use o próximo índice de placa de rede não utilizada, o índice de dispositivo 1 e uma interface de rede EFA (EFA com ENA) ou somente EFA, dependendo do seu caso de uso, como requisitos de largura de banda do ENA ou espaço de endereço IP. Por exemplo, casos de uso, consulte a configuração do EFA para instâncias P5.
- Para obter mais informações, consulte o Guia EFA [aqui](#).

Instâncias P5/P5e

- As instâncias P5 e P5e contêm 32 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    ... \
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

5 instâncias abertas

- O P5en contém 16 placas de interface de rede e pode ser iniciado usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    ... \
    "NetworkCardIndex=15,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

```
"NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
...
"NetworkCardIndex=15,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
sudo yum versionlock kernel*
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para liberar suas versões do kernel:

```
sudo yum versionlock delete kernel*
sudo yum update -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-04-22

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 69.3
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 67.0

Atualizado

- [Driver Nvidia atualizado da versão 550.144.03 para 550.163.01 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de abril de 2025](#)

Data de lançamento: 2025-02-17

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 68.5
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 66.3

Atualizado

- O NVIDIA Container Toolkit foi atualizado da versão 1.17.3 para a versão 1.17.4. Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>

Removido

- As bibliotecas de espaço do usuário cuobj e nvdasm foram removidas fornecidas pelo kit de ferramentas NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas [NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-02-04

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 68.4
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 66.1

Atualizado

- Versão EFA atualizada de 1.37.0 para 1.38.0

Data de lançamento: 2025-01-17

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 68.3
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 66.0

Atualizado

- [Driver Nvidia atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025](#)

Data de lançamento: 2025-01-06

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 68.2
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 65.9

Atualizado

- EFA atualizado da versão 1.34.0 para 1.37.0
- AWS OFI NCCL atualizado da versão 1.11.0 para 1.13.0

Data de lançamento: 2024-12-09

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 68.1
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 65.8

Atualizado

- Kit de ferramentas de contêiner Nvidia atualizado da versão 1.17.0 para 1.17.3

Data de lançamento: 2024-11-09

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 67.9
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 65.6

Atualizado

- [Atualizou o Nvidia Container Toolkit da versão 1.16.2 para a 1.17.0, abordando a vulnerabilidade de segurança CVE-2024-0134.](#)

Data de lançamento: 2024-10-22

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 67.7
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 65.4

Atualizado

- [Driver Nvidia atualizado da versão 550.90.07 para 550.127.05 para endereço CVEs presente no Boletim de Segurança do NVIDIA GPU Display de outubro de 2024](#)

Data de lançamento: 2024-10-03

Nomes da AMI

- Versão AMI do driver OSS Nvidia da Base de Aprendizado Profundo (Amazon Linux 2)
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 65.2

Atualizado

- [Atualizou o Nvidia Container Toolkit da versão 1.16.1 para a 1.16.2, abordando a vulnerabilidade de segurança CVE-2024-0133.](#)

Data de lançamento: 2024-08-27

Nome da AMI: Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) versão 67.0

Atualizado

- Driver Nvidia e Fabric Manager atualizados da versão 535.183.01 para 550.90.07
 - Removido o requisito de shell multiusuário do Fabric Manager com base nas recomendações da Nvidia
 - [Consulte os problemas conhecidos do driver Tesla 550.90.07 aqui para obter mais informações](#)
- Versão EFA atualizada de 1.32.0 para 1.34.0
- NCCL atualizado para a versão mais recente 2.22.3 para todas as versões CUDA
 - CUDA 12.1, 12.2 atualizado de 2.18.5+ 2. CUDA12

- CUDA 12.3 atualizado de 2.21.5+ 4. CUDA12

Adicionado

- Foi adicionado o kit de ferramentas CUDA versão 12.4 no diretório/-12.4 usr/local/cuda
- Foi adicionado suporte para instâncias P5e. EC2

Removido

- Removida a pilha do CUDA Toolkit versão 11.8 presente no diretório/-11.8 usr/local/cuda

Data de lançamento: 2024-08-19

Nome da AMI: Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) versão 66.3

Adicionado

- Foi adicionado suporte para instâncias G6e. EC2

Data de lançamento: 2024-06-06

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 65.4
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 63.9

Atualizado

- Versão atualizada do driver Nvidia para 535.183.01 de 535.161.08

Data de lançamento: 2024-05-02

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 64.7
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 63.2

Atualizado

- Versão EFA atualizada da versão 1.30 para a versão 1.32
- Plugin AWS OFI NCCL atualizado da versão 1.7.4 para a versão 1.9.1
- Kit de ferramentas de contêiner Nvidia atualizado da versão 1.13.5 para a versão 1.15.0

Adicionado

- Foi adicionada uma pilha CUDA12 3.0 com 3.3, NCCL CUDA12 2.21.5, cuDNN 8.9.7

A versão 1.15.0 NÃO inclui os pacotes nvidia-container-runtime nvidia-docker2. É recomendável usar nvidia-container-toolkit pacotes diretamente seguindo a documentação do [kit de ferramentas de contêineres da Nvidia](#).

Removido

- Foram removidas as pilhas de CUDA11 0,7, CUDA12 0,0 presentes em /-12,0 usr/local/cuda-11.7 and /usr/local/cuda
- O pacote nvidia-docker2 e seu comando nvidia-docker foram removidos como parte da atualização do kit de ferramentas de contêiner Nvidia de 1.13.5 para 1.15.0, que NÃO inclui os pacotes nvidia-docker2 e nvidia-docker2. nvidia-container-runtime

Data de lançamento: 2024-04-04

Nome da AMI: Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) versão 64.0

Adicionado

- Para o driver OSS Nvidia DLAMIs, foi adicionado suporte às instâncias G6 e Gr6 EC2

Data de lançamento: 2024-03-29

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 62.3
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 63.2

Atualizado

- Driver Nvidia atualizado de 535.104.12 para 535.161.08 nos drivers Proprietário e OSS Nvidia. DLAMIs
- As novas instâncias suportadas para cada DLAMI são as seguintes:
 - O Deep Learning com driver proprietário da Nvidia suporta G3 (G3.16x não suportado), P3, P3dn
 - Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, P4d, P4de, P5.

Removido

- Removido o suporte às EC2 instâncias G4dn, G5 e G3.16x do driver proprietário da Nvidia DLAMI.

Data de lançamento: 2024-03-20

Nome da AMI: Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) versão 63.1

Adicionado

- Foi adicionado awscliv2 na AMI como /usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awsno OSS Nvidia Driver AMI

Data de lançamento: 2024-03-13

Nome da AMI: Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) versão 63.0

Atualizado

- O driver OSS Nvidia DLAMI atualizado com suporte para G4dn e G5, com base nele, o suporte atual é o seguinte:
 - A AMI de driver Nvidia proprietária da Deep Learning Base (Amazon Linux 2) suporta P3, P3dn, G3, G4dn, G5.
 - O Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) suporta G4dn, G5, P4, P5.
- Recomenda-se que o driver DLAMIs OSS Nvidia seja usado para G4dn, G5, P4, P5.

Data de lançamento: 2024-02-13

Nomes da AMI

- Base de aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 62.1
- AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 62.1

Atualizado

- Driver OSS Nvidia atualizado de 535.129.03 para 535.154.05
- EFA atualizado de 1.29.0 para 1.30.0
- AWS OFI NCCL atualizado de 1.7.3-aws para 1.7.4-aws

Data de lançamento: 2024-02-01

Nome da AMI: AMI de driver Nvidia proprietário da Deep Learning Base (Amazon Linux 2) versão 62.0

Segurança

- [Versão atualizada do pacote runc para consumir o patch para CVE-2024-21626.](#)

Versão 6.1.4

Nome da AMI: Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) versão 61.4

Atualizado

- Driver OSS Nvidia atualizado de 535.104.12 para 535.129.03

Versão 61.0

Nome da AMI: Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) versão 61.4

Atualizado

- EFA atualizado de 1.26.1 para 1.29.0
- GDRCopy atualizado de 2.3 para 2.4

Adicionado

- AWS A AMI de aprendizado profundo (DLAMI) é dividida em dois grupos separados:
 - DLAMI que usa o driver proprietário da Nvidia (para suportar P3, P3dn, G3, G5, G4dn).
 - DLAMI que usa o driver Nvidia OSS para habilitar o EFA (para suportar P4, P5).
- Consulte o [anúncio público](#) para obter mais informações sobre a divisão do DLAMI.
- Para AWS CLI consultas, consulte o marcador Query AMI-ID AWSCLI com (exemplo, a região é us-east-1)

Versão 6.0.6

Nome da AMI: Deep Learning Base AMI (Amazon Linux 2) versão 60.6

Atualizado

- AWS Plugin OFI NCCL atualizado da versão 1.7.2 para a versão 1.7.3
- Diretórios CUDA 12.0-12.1 atualizados com NCCL versão 2.18.5
- CUDA12.1 atualizado como a versão padrão do CUDA
 - LD_LIBRARY_PATH atualizado para ter//usr/local/cuda-12.1/targets/x86_64-linux/lib:/usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1 and PATH to have /usr/local/cuda-12.1/bin
 - Para clientes que desejam mudar para qualquer versão diferente do CUDA, defina as variáveis LD_LIBRARY_PATH e PATH adequadamente.

Adicionado

- O Kernel Live Patching agora está habilitado. A aplicação dinâmica de patches permite que os clientes apliquem vulnerabilidades de segurança e correções de bugs críticos a um kernel Linux em execução, sem reinicializações ou interrupções na execução de aplicativos. Observe que o suporte de patches ao vivo para o kernel 5.10.192 terminará em 30/11/23.

Versão 60.5

Nome da AMI: Deep Learning Base AMI (Amazon Linux 2) versão 60.5

Atualizado

- Driver NVIDIA atualizado de 535.54.03 para 535.104.12

Esse driver mais recente corrige as alterações importantes da ABI do NVML encontradas no driver 535.54.03, bem como a regressão do driver encontrada no driver 535.86.10 que afetou os kits de ferramentas CUDA nas instâncias P5. Consulte as seguintes notas de lançamento da NVIDIA para obter detalhes sobre as correções:

- [4235941](#) - Correção de alteração de falhas na ABI do NVML
- [4228552](#) - Correção de erro do kit de ferramentas CUDA
- Diretórios CUDA 12.2 atualizados com NCCL 2.18.5
- EFA atualizado de 1.24.1 para a última 1.26.1

Adicionado

- Adicionado CUDA12 1.2 em `usr/local/cuda /-12.2`

Removido

- Suporte removido para CUDA 11.5 e CUDA 11.6

Versão 60.2

Nome da AMI: Deep Learning Base AMI (Amazon Linux 2) versão 60.2

Atualizado

- `aws-ofi-ncclPlugin` atualizado de v1.7.1 para v1.7.2

Versão 60.0

Data de lançamento: 2023-08-11

Adicionado

- Essa AMI agora fornece suporte para a funcionalidade de treinamento de vários nós no P5 e em todas as instâncias suportadas anteriormente EC2

- Para EC2 instâncias P5, recomenda-se que o NCCL 2.18 seja usado e foi adicionado a CUDA12 .0 e .1. CUDA12

Removido

- Suporte removido para CUDA11 .5.

Versão 59.2

Data de lançamento: 2023-08-08

Removido

- CUDA-11.3 e CUDA-11.4 removidos

Versão 59.1

Data de lançamento: 2023-08-03

Atualizado

- Plugin AWS OFI NCCL atualizado para v1.7.1
- Feito em CUDA11 0,8 como padrão, pois PyTorch 2.0 suporta 11,8 e, para a EC2 instância P5, é recomendável usar $\geq$.8 CUDA11
 - LD_LIBRARY_PATH atualizado para ter `//usr/local/cuda-11.8/targets/x86_64-linux/lib/:usr/local/cuda-11.8/lib:usr/local/cuda-11.8/lib64:usr/local/cuda-11.8` and PATH to have `/usr/local/cuda-11.8/bin`
 - Para qualquer versão diferente do cuda, defina LD_LIBRARY_PATH adequadamente.

Fixo

- Corrigido o problema de carregamento do pacote Nvidia Fabric Manager (FM) mencionado na data de lançamento anterior de 2023-07-19.

Versão 58.9

Data de lançamento: 2023-07-19

Atualizado

- Driver Nvidia atualizado de 525.85.12 para 535.54.03
- Instalador EFA atualizado de 1.22.1 para 1.24.1

Adicionado

- Foram adicionadas alterações no estado c para desativar o estado ocioso do processador, definindo o estado c máximo como C1. Essa alteração é feita definindo ``intel_idle.max_cstate=1 processor.max_cstate=1`` nos argumentos de inicialização do linux no arquivo `/etc/default/grub`
- AWS EC2 Suporte à instância P5:
 - Foi adicionado suporte à EC2 instância P5 para fluxos de trabalho usando um único nó/instância. O suporte a vários nós (por exemplo, para treinamento em vários nós) usando o EFA (Elastic Fabric Adapter) e o plug-in AWS OFI NCCL será adicionado em uma versão futura.
 - Use `CUDA>=11,8` para um desempenho ideal.
 - Problema conhecido: o pacote Nvidia Fabric Manager (FM) leva tempo para carregar no P5, os clientes precisam esperar de 2 a 3 minutos até que o FM seja carregado após iniciar a instância P5. Para verificar se o FM foi iniciado, execute o comando `sudo systemctl is-active nvidia-fabricmanager`, ele deve retornar ativo antes de iniciar qualquer fluxo de trabalho. Isso será corrigido na próxima versão.

Versão 58.0

Data de lançamento: 2023-05-19

Removido

- A pilha CUDA11 .0-11.2 foi removida de acordo com a política de suporte mencionada na seção superior deste documento.

Versão 5.7.3

Data de lançamento: 2023-04-06

Adicionado

- Adicionado Nvidia 2.3 GDRCopy

Versão 5.6.8

Data de lançamento: 2023-03-09

Atualizado

- Driver NVIDIA atualizado de 515.65.01 para 525.85.12

Adicionado

- Adicionado cuda-11.8 em `usr/local/cuda`

Versão 56.0

Data de lançamento: 2022-12-06

Atualizado

- Versão EFA atualizada de 1.17.2 para 1.19.0

Versão 55.0

Data de lançamento: 2022-11-04

Atualizado

- Driver NVIDIA atualizado de 510.47.03 para 515.65.01

Adicionado

- Adicionado cuda-11.7 em `usr/local/cuda`

Versão 54.0

Data de lançamento: 2022-09-15

Atualizado

- Versão EFA atualizada de 1.16.0 para 1.17.2

Versão 53.3

Data de lançamento: 2022-05-25

Atualizado

- Atualizado aws-efa-installer para a versão 1.15.2
- Atualizado aws-ofi-nccl para a versão 1.3.0-aws, que inclui a topologia para p4de.24xlarge.

Adicionado

- Esta versão adiciona suporte para instâncias EC2 p4de.24xlarge.

Versão 53.0

Data de lançamento: 2022-04-28

Adicionado

- CloudWatch Agente Amazon adicionado
- Foram adicionados três serviços systemd que usam arquivos json predefinidos disponíveis em path/opt/aws/amazon-cloudwatch-agent/etc/para configurar métricas de GPU usando o usuário linux cwagent
- dlami-cloudwatch-agent@minimal
 - Comandos para ativar as métricas da GPU:

```
sudo systemctl enable dlami-cloudwatch-agent@minimal
sudo systemctl start dlami-cloudwatch-agent@minimal
```

- Ele cria essas métricas:utilization_gpu, utilization_memory
- dlami-cloudwatch-agent@partial
 - Comandos para ativar as métricas da GPU:

```
sudo systemctl enable dlami-cloudwatch-agent@partial
sudo systemctl start dlami-cloudwatch-agent@partial
```

- Ele cria essas métricas:utilization_gpu,utilization_memory,memory_total,memory_used, memory_free

- dlami-cloudwatch-agent@all
- Comandos para ativar as métricas da GPU:

```
sudo systemctl enable dlami-cloudwatch-agent@all  
sudo systemctl start dlami-cloudwatch-agent@all
```

- Ele cria todas as métricas de GPU disponíveis

Versão 52.0

Data de lançamento: 2022-03-08

Atualizado

- Versão atualizada do Kernel para 5.10

Versão 51.0

Data de lançamento: 2022-03-04

Atualizado

- Driver Nvidia atualizado para 510.47.03

Versão 50.0

Data de lançamento: 2022-02-17

Atualizado

- Bloqueados aws-neuron-dkms e à tensorflow-model-server-neuron medida que são atualizados para versões mais recentes, que não são suportadas pelos pacotes Neuron presentes na AMI
 - Comandos se o cliente quiser desbloquear o pacote para atualizá-lo para a versão mais recente:
`sudo yum versionlock delete sudo yum versionlock delete aws-neuron-dkms tensorflow-model-server-neuron`

Versão 49.0

Data de lançamento: 2022-01-13

Adicionado

- Foi adicionado CUDA11 1.2 com os seguintes componentes:
 - cuDNN v8.1.1.33
 - NCCL 2.8.4
 - CUDA 11.2.2

Atualizado

- Link simbólico atualizado pip para pip3

Defasagens

- Suporte obsoleto para o tipo de instância P2
- O python2.7 foi descontinuado e os pacotes relacionados do python2.7 foram removidos, como “python-dev”, “python-pip” e “python-tk”

Versão 48.0

Data de lançamento: 2021-12-27

Atualizado

- O org.apache.ant_1.9.2.v201404171502\lib\ ant-apache-log 4j.jar das versões cuda foi removido, pois ele não está sendo usado e não há risco para os usuários que têm os arquivos Log4j. Para obter mais informações, consulte https://nvidia.custhelp.com/app/answers/detail/a_id/5294.

Versão 47.0

Data de lançamento: 2021-11-24

Atualizado

- EFA atualizado para 1.14.1

Versão 46.0

Data de lançamento: 2021-11-12

Atualizado

- Pacotes Neuron atualizados de aws-neuron-dkms =1.5. *, aws-neuron-runtime-base =1,5. *, aws-neuron-tools =1,6.* a =2,2. aws-neuron-dkms *, aws-neuron-runtime-base =1,6. *, aws-neuron-tools =2,0.*.
- O pacote Neuron foi removido aws-neuron-runtime =1.5.*, pois o Neuron não tem mais um tempo de execução em execução como daemon e o tempo de execução agora está integrado à estrutura como uma biblioteca.

Versão 45.0

Data de lançamento: 2021-10-21

Adicionado

- Os relatórios de verificação de segurança no formato JSON estão disponíveis em/opt/aws/dlami/info/.

Versão 44.0

Data de lançamento: 2021-10-08

Alterado

- Para cada execução de instância usando DLAMI, a tag aws-dlami-autogenerated-tag "do-not-delete-" será adicionada, AWS permitindo coletar o tipo de instância, o ID da instância, o tipo de DLAMI e as informações do sistema operacional. Nenhuma informação sobre os comandos usados na DLAMI é coletada ou retida. Nenhuma outra informação sobre a DLAMI é coletada ou retida. Para desativar o rastreamento de uso do seu DLAMI, adicione uma tag à sua instância da EC2 Amazon durante o lançamento. A tag deve usar a chave OPT_OUT_TRACKING com o valor associado definido como true. Para obter mais informações, consulte [Marcar seus EC2 recursos da Amazon](#).

Segurança

- Versão atualizada do docker para docker-20.10.7-3

Versão 43.0

Data de lançamento: 2021-08-24

Alterado

- “notebook” atualizado para a versão “6.4.1”.

Versão 42.0

Data de lançamento: 2021-07-23

Alterado

- O driver Nvidia e a versão do Fabric Manager foram atualizados para 450.142.00.

Versão 41.0

Data de lançamento: 2021-06-24

Alterado

- Pacotes Neuron atualizados de acordo com o Neuron Release v1.14.0

Versão 40.0

Data de lançamento: 2021-06-10

Alterado

- Versão awscli atualizada para 1.19.89

Versão 39.0

Data de lançamento: 2021-05-27

Segurança

- Componentes vulneráveis do CUDA-10.0 (Visual Profiler, Nsight EE e JRE) foram removidos da instalação do CUDA-10.0 (/usr/local/cuda)

Versão 38.0

Data de lançamento: 2021-05-25

Alterado

- Runc atualizado para o mais recente

Versão 37.0

Data de lançamento: 2021-04-23

Alterado

- O driver Nvidia Tesla e a versão do Fabric Manager foram atualizados para 450.119.03.

Versão 36.1

Data de lançamento: 2021-04-21

Fixo

- Corrigido um problema que diminuía a velocidade de inicialização da instância.

Versão 36.0

Data de lançamento: 2021-03-24

Adicionado

- Adicionado tensorflow-model-server-neuron para oferecer suporte ao serviço de modelos de neurônios.

Alterado

- O jupyterlab foi atualizado para a versão 3.0.8 para python3.

Fixo

- A instalação antiga do OpenMPI em `/usr/local/mpi` caused `/opt/amazon/openmpi/bin/mpirun` to be linked incorrectly. To fix the link issue, we removed `/usr/local/mpi` installation, OpenMPI installation in `/opt/amazon/openmpi` está disponível.
- Remova a definição duplicada e inexistente de ambientes de shell que estão poluindo as variáveis de ambiente de shell, como `PATH` e `LD_LIBRARY_PATH`. Como resultado, `~/.dlami` e `/etc/profile.d/var.sh` has been removed, and `/etc/profile.d/dlami.sh` foram adicionados.

Segurança

- [Criptografia de pacote atualizada para endereçar CVE-2020-36242](#)

Versão 35.0

Data de lançamento: 2021-03-08

Adicionado

- Instalação adicionada do [TensorRT](#) CUDA 11.0

Versão 34.3

Data de lançamento: 2021-02-25

Fixo

- Corrigido um erro de digitação no MOTD (mensagem do dia) que exibia incorretamente a versão 34.1.

Versão 34.2

Data de lançamento: 2021-02-24

Segurança

- Python2 e python3 corrigidos para CVE-2021-3177

Problema conhecido

- Há um erro de digitação na MOTD (mensagem do dia) que exibiu incorretamente a versão 34.1. Lançaremos a versão 34.3 para resolver esse problema.

Versão 34.0

Data de lançamento: 2021-02-09

Alterado

- Fixado o pip na versão 20.3.4 para python2, esta é a última versão do pip que suporta python2 e python3.5.

Versão 33.0

Data de lançamento: 2021-01-19

Alterado

- Versão cuDNN atualizada para CUDA11 v8.0.5.39 em .0 e .1. CUDA11

Versão 32.0

Data de lançamento: 2020-12-01

Adicionado

- Foi adicionado CUDA11 1 com NCCL 2.7.8, cuDNN 8.0.4.30 para AMI de aprendizado profundo (Amazon Linux 2), AMI de aprendizado profundo (Ubuntu 16.04), AMI de aprendizado profundo (Ubuntu 18.04), AMI de base de aprendizado profundo (Ubuntu 16.04), AMI de base de aprendizado profundo (Ubuntu 18.04), AMI de base de aprendizado profundo (Amazon Linux 2).

Versão 31.0

Data de lançamento: 2020-11-02

Alterado

- Instalador EFA atualizado para a versão 1.10.0.

- Versão cuDNN atualizada para v8.0.4.30 para CUDA 11.0.
- AWS Neuron atualizado para a versão 1.1

Versão 30.0

Data de lançamento: 2020-10-08

Alterado

- Versões atualizadas do driver NVIDIA e do Fabric Manager para 450.80.02
- NCCL atualizado para 2.7.8 in for 0. CUDA11

Fixo

- Corrigido um problema em que o pacote python gerenciado yum era substituído por instalações gerenciadas por pip. Os executáveis pip, pip3 e pip3.7 foram movidos de/parte dessa correção.
usr/binto /usr/local/binas

Versão 29.0

Data de lançamento: 2020-09-11

Alterado

- Driver NVIDIA atualizado da versão 450.51.05 para 450.51.06
- Adicionado o NVIDIA Fabric Manager versão 450.51.06
- EFA atualizado para 1.9.4

Versão 28.0

Data de lançamento: 2020-08-19

Alterado

- Foi adicionada a pilha CUDA 11.0 com NCCL 2.7.6 e cuDNN 8.0.2.39

Versão 27.0

Data de lançamento: 2020-08-07

Alterado

- EFA atualizado da versão 1.7.1 para 1.9.3 em /opt/amazon/efa
- O Open MPI atualizado da versão 4.0.3 para 4.0.4 em /usr/local/mpi. Open MPI at '/opt/amazon/openmpi/bin/mpirun '/' ainda está na versão 4.0.3
- Driver NVIDIA atualizado de 440.33.01 para 450.51.05
- Versão NCCL atualizada de 2.6.4 para 2.7.6 em 0.2 CUDA1

Versão 26.0

Data de lançamento: 2020-08-03

Alterado

- AWS OFI NCCL atualizado para o mais recente, veja [aqui](#) para obter mais detalhes.
- O Cuda 8.0/9.0/9.2 foi removido da AMI

Fixo

- Corrigido um erro em que o arquivo de objeto compartilhado: libopencv_dnn.so.4.2 não podia ser aberto.

Versão 25.0

Data de lançamento: 2020-07-19

Alterado

- Versão EFA atualizada para 1.7.1 para suportar NCCL 2.6.4
- Versão NCCL atualizada para 2.6.4 para CUDA 10.2
- versão awscli atualizada de 1.16.76 para 1.18.80
- versão boto3 atualizada de 1.9.72 para 1.14.3

Versão 24.1

Data de lançamento: 2020-06-14

Alterado

- Versão do Docker atualizada para 19.03.6

Versão 24.0

Data de lançamento: 2020-05-20

Alterado

- Versão do Docker atualizada para 19.03.6

Versão 23.0

Data de lançamento: 2020-04-29

Alterado

- Versões atualizadas do pacote python

Versão 22.0

Data de lançamento: 2020-03-04

Alterado

- Foi adicionada a pilha CUDA 10.2
- CUDA 10.0 e 10.1 atualizados para as versões cuDNN e NCCL

AWS Base de aprendizado profundo Qualcomm AMI (Amazon Linux 2)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Base de aprendizado profundo Qualcomm AMI (Amazon Linux 2) \$ {YYYY-MM-DD}

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2

- Sistema operacional: Amazon Linux 2
- Arquitetura de computação: x86
- Kernel Linux: 5.10.210-201.852.amzn2.x86_64
- Localização do SDK: /opt/qai-sdk
- Localização dos utilitários QTI: /opt/qti-aic
- Versão do SDK da plataforma: 1.12.0.88
- Versão do SDK do Apps: 1.12.0.87
- Tipo de volume EBS: gp3
- Python: python3.8 python3.8
- EC2 Instâncias suportadas: dl2q
- Consulte o AMI-ID com AWS CLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon \
  --filters 'Name=name,Values=Deep Learning Base Qualcomm AMI (Amazon Linux
2) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```

Avisos

[4/12/24] Remoção do Audit Package

- DLAMIs lançados entre 26 de março de 2024 (2024-03-26) e 12 de abril de 2024 (2024-04-12) foram enviados sem o pacote de auditoria. Se você precisar desse pacote específico para suas necessidades de registro e monitoramento, migre seus fluxos de trabalho para o DLAMI mais recente para consumir aqueles com o pacote de auditoria instalado.

Ambiente QAIC:

- Por padrão, as qaic-pytools não são habilitadas por meio do SDK do Qualcomm 00 Apps. A11 Para habilitar e criar o ambiente qaic pip qaic-env, execute os seguintes comandos:

```
echo 'yes' | bash /opt/qai-sdk/qaic-apps-*/uninstall.sh
cd /opt/qai-sdk/qaic-apps-*/
```

```
sudo sed -i "s/python3 -V/python3.8 -V/g" install.sh
./install.sh --enable-qaic-pytools
```

Versão 20240314

Data de lançamento: 2024-03-14

Nome da AMI: Base de aprendizado profundo Qualcomm AMI (Amazon Linux 2) 20240314

Adicionado

- SDK da plataforma AI 100 atualizado da versão 1.10.0.200 para a versão 1.12.0.88
- AI 100 Apps SDK atualizado da versão 1.10.0.193 para a versão 1.12.0.87
- A versão 1.12 do SDK adiciona suporte para modelos de decodificadores de transformadores, como Llama-2 e Starcoder

Versão 20240110

Data de lançamento: 2024-01-10

Nome da AMI: Base de aprendizado profundo Qualcomm AMI (Amazon Linux 2) 20240103

Adicionado

- Imagem da plataforma Qualcomm AI 100 atualizada para 1.10.0.200

Versão 20231115

Data de lançamento: 2023-11-15

Nome da AMI: Base de aprendizado profundo Qualcomm AMI (Amazon Linux 2) 20231115

Adicionado

- Lançamento inicial da série Deep Learning Base Qualcomm AMI (Amazon Linux 2).
 - [Consulte a documentação oficial da Qualcomm para obter mais informações sobre o SDK de plataformas e aplicativos: https://quic.github.io//cloud-ai-sdk-pages](https://quic.github.io//cloud-ai-sdk-pages)
 - [Consulte a AWS documentação oficial para saber mais sobre instâncias dl2q: DL2q instâncias.](#)

ARM64 Notas de lançamento do DLAMI básico

- [ARM64 Notas de lançamento do DLAMI básico](#)

ARM64 Notas de lançamento do DLAMI básico

Abaixo estão as notas de lançamento do ARM64 DLAMI Base:

GPU

- [AWS ARM64 AMI de base de aprendizado profundo \(Amazon Linux 2023\) \(compatível com P6e-GB2 00\)](#)
- [AWS ARM64 AMI de base de aprendizado profundo \(Ubuntu 22.04\) \(suporta P6e-00\) GB2](#)
- [AWS ARM64 AMI de base de aprendizado profundo \(Amazon Linux 2\)](#)

AWS Neuron

- Consulte o Guia do usuário do [Neuron DLAMI](#).

AWS AMI de GPU de ARM64 base de aprendizado profundo (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 Base de aprendizado profundo OSS Nvidia Driver GPU AMI (Amazon Linux 2023) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g, P6e- GB2 00 (CUDA $\geq$ 12.8 é suportado em P6e- 00) GB2

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: ARM64
- A versão mais recente disponível está instalada para os seguintes pacotes:
 - Kernel Linux: 6. 12

- FSx Lustre
- Docker
- AWS CLI v2 em /usr/bin/aws
- NVIDIA DCGM
- Kit de ferramentas de contêiner Nvidia:
 - Comando de versão: `nvidia-container-cli -V`
- NVIDIA-Docker 2:
 - Comando de versão: `versão nvidia-docker`
- Driver NVIDIA: 570.158.01
- Pilha NVIDIA CUDA 12.4, 12.5, 12.6, 12.8:
 - Diretórios de instalação CUDA, NCCL e cuDDN: `/usr/local/cuda`
 - Exemplo: `/usr/local/cuda-12.8/` , `/usr/local/cuda-12.8/`
 - Versão compilada da NCCL:
 - Para o diretório CUDA de 12.4, compilou a versão NCCL 2.22.3+ .4 CUDA12
 - Para o diretório CUDA de 12.5, compilou a versão NCCL 2.22.3+ .5 CUDA12
 - Para o diretório CUDA de 12.6, compilou a versão NCCL 2.24.3+ .6 CUDA12
 - Para o diretório CUDA de 12.8, compilou a versão NCCL 2.27.5+ .8 CUDA12
 - CUDA padrão: 12.8
 - `PATH` /usr/local/cuda aponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - `LD_LIBRARY_PATH` deve ter `/usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/sbsa-linux/lib:/usr/local/cuda-12.8/nvvm/lib64:/usr/local/cuda-12.8/extras/CUPTI/lib`
 - `PATH` para ter `/usr/local/cuda-12.8/bin:/usr/local/cuda-12.8/include/`
 - Para qualquer versão diferente do CUDA, atualize `LD_LIBRARY_PATH` adequadamente.
- Instalador EFA: 1.42.0
- Nvidia GDRCopy: 2.5.1
- AWS O plugin OFI NCCL vem com o instalador EFA
 - Os caminhos `/opt/amazon/ofi-nccl/lib` and `/opt/amazon/ofi-nccl/efasão` adicionados ao `LD_LIBRARY_PATH`.
- AWS CLI v2 em /usr/local/bin/aws

- Tipo de volume do EBS: gp3
- Python:/3.9 usr/bin/python
- Consulte o AMI-ID com o parâmetro SSM (a região de exemplo é us-east-1):

```
SSM_PARAMETER=base-oss-nvidia-driver-gpu-amazon-linux-2023/latest/ami-id \
aws ssm get-parameter --region us-east-1 \
--name /aws/service/deeplearning/ami/arm64/$SSM_PARAMETER \
--query "Parameter.Value" \
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters
'Name=name,Values=Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux
2023) ????????' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
&CreationDate))[:1].ImageId' --output text
```

P6-00 instâncias GB2

As instâncias P6e- GB2 00 contêm 17 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
--instance-type $INSTANCETYPE \
--image-id $AMI --key-name $KEYNAME \
--iam-instance-profile "Name=dlami-builder" \
--tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
--network-interfaces \
    "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=interface" \
    "NetworkCardIndex=1,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=2,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=3,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=4,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=5,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
```

```

    "NetworkCardIndex=6,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=7,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=8,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=9,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=10,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=11,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=12,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=13,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=14,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=15,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=16,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only"

```

Avisos

Kit de ferramentas de contêiner NVIDIA 1.17.4

No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Política de suporte

Esses AMIs componentes dessa AMI, como as versões CUDA, podem ser removidos e alterados com base na [política de suporte da estrutura](#) ou para otimizar o desempenho de [contêineres de aprendizado profundo](#) ou para reduzir o tamanho da AMI em uma versão futura, sem aviso prévio. Removemos as versões CUDA AMIs se elas não forem usadas por nenhuma versão de estrutura compatível.

Kernel

- A versão do kernel é fixada usando o comando:

```
sudo dnf versionlock kernel*
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para liberar suas versões do kernel:

```
sudo dnf versionlock delete kernel*  
sudo dnf update -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-07-04

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023)
20250704

Atualizado

- Foi adicionado suporte à EC2 instância P6e- 00. GB2 Observe que CUDA>=12.8 é suportado no P6e- 00 GB2
- Adicionar EFA 1.42.0
- Driver Nvidia atualizado da versão 570.133.20 para 570.158.01
- Pilha CUDA 12.8 atualizada com NCCL 2.27.5

Data de lançamento: 2025-04-24

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023)
20250424

Atualizado

- [O driver Nvidia foi atualizado da versão 570.86.15 para 570.133.20 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de abril de 2025](#)

- Pilha CUDA12 3.8 atualizada com NCCL 2.26.2
- CUDA padrão atualizado de 12,6 para 12,8

Data de lançamento: 2025-04-22

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023)
20250421

Atualizado

- [Driver Nvidia atualizado da versão 570.124.06 para 570.133.20 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de abril de 2025](#)

Data de lançamento: 2025-04-04

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023)
20250404

Atualizado

- Versão do kernel atualizada de 6.1 para 6.12

Data de lançamento: 2025-03-03

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023)
20250303

Atualizado

- Driver Nvidia de 550.144.03 a 570.86.15
- O CUDA padrão foi alterado de CUDA12 4,4 para 0,6 CUDA12

Adicionado

- Diretório CUDA de 12.5 com NCCL versão CUDA12 2.22.3+ .5 compilada e cuDNN 9.7.1.26
- Diretório CUDA de 12.6 com NCCL versão CUDA12 2.24.3+ .6 compilada e cuDNN 9.7.1.26
- Diretório CUDA de 12.8 com NCCL compilado versão CUDA12 2.25.1+ .8 e cuDNN 9.7.1.26

Data de lançamento: 2025-02-14

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250214

Adicionado

- Lançamento inicial do Deep Learning ARM64 Base OSS DLAMI para Amazon Linux 2023

AWS AMI de GPU de ARM64 base de aprendizado profundo (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 Base de aprendizado profundo OSS Nvidia Driver GPU AMI (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g, P6e- GB2 00 (CUDA>=12.8 é suportado em P6e- 00) GB2

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: ARM64
- A versão mais recente disponível está instalada para os seguintes pacotes:
 - Kernel Linux: 6. 8
 - FSx Lustro
 - Docker
 - AWS CLI v2 em/usr/bin/aws
 - NVIDIA DCGM
 - Kit de ferramentas de contêiner Nvidia:
 - Comando de versão: nvidia-container-cli -V
 - NVIDIA-Docker 2:

- Comando de versão: versão nvidia-docker
- Driver NVIDIA: 570.158.01
- Pilha NVIDIA CUDA 12.4, 12,5, 12,6, 12,8:
 - Diretórios de instalação CUDA, NCCL e cuDDN: /usr/local/cuda
 - Exemplo: /usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
- Versão compilada da NCCL:
 - Para o diretório CUDA de 12.4, compilou a versão NCCL 2.22.3+ .4 CUDA12
 - Para o diretório CUDA de 12.5, compilou a versão NCCL 2.22.3+ .5 CUDA12
 - Para o diretório CUDA de 12.6, compilou a versão NCCL 2.24.3+ .6 CUDA12
 - Para o diretório CUDA de 12.8, compilou a versão NCCL 2.27.5+ .8 CUDA12
- CUDA padrão: 12,8
 - PATH/usr/local/cudaaponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter/64 usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/sbsa-linux/lib:/usr/local/cuda-12.8/nvvm/lib64:/usr/local/cuda-12.8/extras/CUPTI/lib
 - PATH para ter/usr/local/cuda-12.8/bin:/usr/local/cuda-12.8/include/
 - Para qualquer versão diferente do CUDA, atualize LD_LIBRARY_PATH adequadamente.
- Instalador EFA: 1.42.0
- Nvidia GDRCopy: 2.5.1
- AWS O plugin OFI NCCL vem com o instalador EFA
 - Os caminhos/opt/amazon/ofi-nccl/lib and /opt/amazon/ofi-nccl/efasão adicionados ao LD_LIBRARY_PATH.
- AWS CLI v2 em/usr/local/bin/aws2 e AWS CLI v1 em/usr/bin/aws
- Tipo de volume do EBS: gp3
- Python:/3.10 usr/bin/python
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):

```
SSM_PARAMETER=base-oss-nvidia-driver-gpu-ubuntu-22.04/latest/ami-id \
aws ssm get-parameter --region us-east-1 \
--name /aws/service/deeplearning/ami/arm64/$SSM_PARAMETER \
--query "Parameter.Value" \
```

```
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon --filters 'Name=name,Values=Deep Learning ARM64 Base OSS Nvidia
  Driver GPU AMI (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```

P6-00 instâncias GB2

As instâncias P6e- GB2 00 contêm 17 placas de interface de rede e podem ser iniciadas usando o seguinte comando: AWS CLI

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces \
    "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=interface" \
    "NetworkCardIndex=1,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=2,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=3,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=4,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=5,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=6,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=7,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=8,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=9,DeviceIndex=0,Groups=$SG,SubnetId=
  $SUBNET,InterfaceType=efa-only" \
```

```

    "NetworkCardIndex=10,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=11,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=12,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=13,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=14,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=15,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only" \
    "NetworkCardIndex=16,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa-only"

```

Avisos

Kit de ferramentas de contêiner NVIDIA 1.17.4

No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Suporte para vários ENI

- O Ubuntu 22.04 configura e configura automaticamente o roteamento de origem em vários NICs via cloud-init em sua inicialização inicial. Se seu fluxo de trabalho incluir attaching/detaching suas ENIs enquanto uma instância estiver parada, uma configuração adicional deverá ser adicionada aos dados do usuário cloud-init para garantir a configuração adequada das NICs durante esses eventos. Uma amostra da configuração da nuvem é fornecida abaixo.
- [Consulte esta documentação da Canonical aqui para obter mais informações sobre como configurar a configuração de nuvem para suas instâncias - /https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automaticallysetup-multiple-nics](https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automaticallysetup-multiple-nics)

```

#cloud-config
# apply network config on every boot and hotplug event
updates:
  network:

```

```
when: ['boot', 'hotplug']
```

Política de suporte

Esses AMIs componentes dessa AMI, como as versões CUDA, podem ser removidos e alterados com base na [política de suporte da estrutura](#) ou para otimizar o desempenho de [contêineres de aprendizado profundo](#) ou para reduzir o tamanho da AMI em uma versão futura, sem aviso prévio. Removemos as versões CUDA AMIs se elas não forem usadas por nenhuma versão de estrutura compatível.

Kernel

- A versão do kernel é fixada usando o comando:

```
echo linux-aws hold | sudo dpkg --set-selections
echo linux-headers-aws hold | sudo dpkg --set-selections
echo linux-image-aws hold | sudo dpkg --set-selections
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
echo linux-aws install | sudo dpkg --set-selections
echo linux-headers-aws install | sudo dpkg --set-selections
echo linux-image-aws install | sudo dpkg --set-selections
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-07-04

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250704

Atualizado

- Foi adicionado suporte à EC2 instância P6e- 00. GB2 Observe que CUDA>=12.8 é suportado no P6e- 00 GB2
- Adicionar EFA 1.42.0
- Driver Nvidia atualizado da versão 570.133.20 para 570.158.01

- Pilha CUDA 12.8 atualizada com NCCL 2.27.5

Data de lançamento: 2025-04-24

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250424

Atualizado

- [O driver Nvidia foi atualizado da versão 570.86.15 para 570.133.20 para abordar os CVEs presentes no boletim de segurança do driver de vídeo GPU NVIDIA de abril de 2025](#)
- Pilha CUDA 12.8 atualizada com NCCL 2.26.2
- CUDA padrão atualizado de 12,6 para 12,8
- CUDA 12.3 removido

Data de lançamento: 2025-03-03

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250303

Atualizado

- Driver Nvidia de 550.144.03 a 570.86.15
- O CUDA padrão foi alterado de CUDA12 0,1 para 6. CUDA12

Adicionado

- Diretório CUDA de 12.4 com NCCL versão CUDA12 2.22.3+ .4 compilada e cuDNN 9.7.1.26
- Diretório CUDA de 12.5 com NCCL versão CUDA12 2.22.3+ .5 compilada e cuDNN 9.7.1.26
- Diretório CUDA de 12.6 com NCCL versão CUDA12 2.24.3+ .6 compilada e cuDNN 9.7.1.26
- Diretório CUDA de 12.8 com NCCL compilado versão CUDA12 2.25.1+ .8 e cuDNN 9.7.1.26

Removido

- Diretório CUDA de 12.1 e 12.2

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250214

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-17

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250117

Atualizado

- [Driver Nvidia atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025](#)

Data de lançamento: 2024-10-23

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20241023

Atualizado

- [Driver Nvidia atualizado da versão 550.90.07 para 550.127.05 para endereço CVEs presente no Boletim de Segurança do NVIDIA GPU Display de outubro de 2024](#)

Data de lançamento: 2024-06-06

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240606

Atualizado

- Versão atualizada do driver Nvidia para 535.183.01 de 535.161.08

Data de lançamento: 2024-05-15

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240514

Adicionado

- Lançamento inicial do Deep Learning ARM64 Base OSS DLAMI para Ubuntu 22.04

AWS AMI de GPU de ARM64 base de aprendizado profundo (Amazon Linux 2)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 Base de aprendizado profundo OSS Nvidia Driver GPU AMI (Amazon Linux 2) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2
- Arquitetura de computação: ARM64
- Kernel Linux: 5.10
- Driver NVIDIA: 550.144.03
- Pilha NVIDIA CUDA 12.1, 12.2, 12.3:
 - Diretórios de instalação CUDA, NCCL e cuDDN:
 - Exemplo: /usr/local/cuda-12.1/ , /usr/local/cuda-12.1/
 - Versão compilada da NCCL:
 - Para o diretório CUDA de 12.3, compilou a versão NCCL 2.21.5+ .4 CUDA12

- Para o diretório CUDA de 12.1, 12.2, compilou a versão NCCL 1.8.5+ .2 CUDA12
- CUDA padrão: 12.1
- PATH/usr/local/cudaaponta para CUDA 12.1
- Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter/64 usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1:/usr/local/cuda-12.1/targets/sbsa-linux/lib:/usr/local/cuda-12.1/nvvm/lib64:/usr/local/cuda-12.1/extras/CUPTI/lib
 - PATH para ter/usr/local/cuda-12.1/bin:/usr/local/cuda-12.1/include/
 - Para qualquer versão diferente do CUDA, atualize LD_LIBRARY_PATH adequadamente.
- AWS CLI v2 em/usr/local/bin/aws2 e AWS CLI v1 em/usr/bin/aws
- Tipo de volume do EBS: gp3
- Kit de ferramentas de contêiner Nvidia: 1.16.2
 - Comando de versão: nvidia-container-cli -V
- Docker: 26.1.2
- Python:/3.10 usr/bin/python
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):

```
SSM_PARAMETER=base-oss-nvidia-driver-gpu-amazon-linux-2/latest/ami-id \
aws ssm get-parameter --region us-east-1 \
--name /aws/service/deeplearning/ami/arm64/$SSM_PARAMETER \
--query "Parameter.Value" \
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 \
--owners amazon \
--filters 'Name=name,Values=Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2) ????????' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
--output text
```

Avisos

Kit de ferramentas de contêiner NVIDIA 1.17.4

No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêineres, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Política de suporte

Esses AMIs componentes dessa AMI, como as versões CUDA, podem ser removidos e alterados com base na [política de suporte da estrutura](#) ou para otimizar o desempenho de [contêineres de aprendizado profundo](#) ou para reduzir o tamanho da AMI em uma versão futura, sem aviso prévio. Removemos as versões CUDA AMIs se elas não forem usadas por nenhuma versão de estrutura compatível.

Kernel

- A versão do kernel é fixada usando o comando:

```
sudo yum versionlock kernel*
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
sudo yum versionlock delete kernel*  
sudo yum update -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2)
20250214

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>

- Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêineres, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-17

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2)
20250117

Atualizado

- [Driver Nvidia atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025](#)

Data de lançamento: 2024-10-22

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2)
20241022

Atualizado

- [Driver Nvidia atualizado da versão 550.90.07 para 550.127.05 para endereço CVEs presente no Boletim de Segurança do NVIDIA GPU Display de outubro de 2024](#)

Data de lançamento: 2024-10-08

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2)
20241008

Atualizado

- [Atualizou o Nvidia Container Toolkit da versão 1.16.1 para a 1.16.2, abordando a vulnerabilidade de segurança CVE-2024-0133.](#)

Data de lançamento: 2024-06-06

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2)
20240606

Atualizado

- Versão atualizada do driver Nvidia para 535.183.01 de 535.161.08

Data de lançamento: 2024-05-14

Nome da AMI: Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2)
20240514

Adicionado

- Lançamento inicial do Deep Learning ARM64 Base OSS DLAMI para Amazon Linux 2

Notas de lançamento do Single Framework DLAMIs

Notas de lançamento do DLAMI de estrutura única

- [PyTorch DLAMIs](#)
- [TensorFlow DLAMIs](#)

PyTorch DLAMIs

Notas de lançamento do Multi Framework DLAMI

- [Notas de lançamento do X86 PyTorch DLAMI](#)
- [ARM64 PyTorch Notas de lançamento do DLAMI](#)

Notas de lançamento do X86 PyTorch DLAMI

Abaixo estão as notas de lançamento do X86: PyTorch DLAMIs

GPU

- [AWS AMI GPU PyTorch 2.7 de aprendizado profundo \(Amazon Linux 2023\)](#)
- [AWS AMI GPU PyTorch 2.7 de aprendizado profundo \(Ubuntu 22.04\)](#)
- [AWS AMI GPU PyTorch 2.6 de aprendizado profundo \(Amazon Linux 2023\)](#)
- [AWS AMI GPU PyTorch 2.6 de aprendizado profundo \(Ubuntu 22.04\)](#)
- [AWS GPU AMI de aprendizado profundo PyTorch 2.5 \(Amazon Linux 2023\)](#)
- [AWS GPU AMI de aprendizado profundo PyTorch 2.5 \(Ubuntu 22.04\)](#)
- [AWS GPU AMI de aprendizado profundo PyTorch 2.4 \(Ubuntu 22.04\)](#)

AWS Neuron

- Consulte o Guia do usuário do [Neuron DLAMI](#)

AWS GPU PyTorch 2.7 OSS AMI de aprendizado profundo (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI GPU 2.7 PyTorch (Amazon Linux 2023) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#)
- G4dn, G5, G5, G6, P4, P4de, P5, P5e, P5en, P6-B200

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: x86

- Kernel Linux: 6.1
- Driver NVIDIA: 570.133.20
- Pilha NVIDIA CUDA 12.8:
 - Diretórios de instalação CUDA, NCCL e cuDDN: /usr/local/cuda
 - Local dos testes NCCL:
 - all_reduce, all_gather e reduce_scatter:

```
/usr/local/cuda-12.8/efa/test-cuda-12.8/
```

- Para executar testes NCCL, o LD_LIBRARY_PATH já está atualizado com os caminhos necessários.
- PATHs Os comuns já foram adicionados ao LD_LIBRARY_PATH:

```
/opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/amazon/ofi-nccl/lib:/usr/local/lib:/usr/lib
```

- O LD_LIBRARY_PATH é atualizado com os caminhos da versão CUDA:

```
/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
```

- Versão compilada da NCCL:
 - Para o diretório CUDA de 12.8, compilou a versão NCCL 2.26.2+ .8 CUDA12
- CUDA padrão: 12,8
 - PATH/usr/local/cudaaponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda/targets/x86_64-linux/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
- Instalador EFA: 1.40.0
- Nvidia GDRCopy: 2,5
- AWS OFI NCCL: 1.14.2-aws
 - Caminho de instalação:/opt/amazon/ofi-nccl/. Path /opt/amazon/ofi-nccl/libé adicionado ao LD_LIBRARY_PATH
- AWS CLI v2 em/usr/local/bin/aws

- Tipo de volume do EBS: gp3
- Kit de ferramentas de contêiner Nvidia: 1.17.7
 - Comando de versão: `nvidia-container-cli -V`
- Docker: 25.0.8
- Python: /3.12 usr/bin/python
- Consulte o AMI-ID com o parâmetro SSM (a região de exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-pytorch-2.7-
amazon-linux-2023/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters
'Name=name,Values=Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Amazon Linux
2023) ????????' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
&CreationDate))[1].ImageId' --output text
```

Avisos

Instâncias P6-B200

- As instâncias P6-B200 exigem CUDA versão 12.8 ou superior e driver NVIDIA 570 ou drivers mais recentes.
- O P6-B200 contém 8 placas de interface de rede e pode ser iniciado usando o seguinte comando CLI AWS :

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces ""NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
```

```
"NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
"NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
....
....
....
"NetworkCardIndex=7,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Instâncias P5/P5e

- DeviceIndex é exclusivo para cada um NetworkCard e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1. Abaixo está um exemplo do comando de inicialização da instância EC2 P5 usando awscli exibido NetworkCardIndex para os números de 0 a 31 e DeviceIndex como 0 para a primeira interface e 1 para as 31 interfaces restantes.

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces ""NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  ....
  ....
  ....
  "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
sudo dnf versionlock kernel*
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões

do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
sudo dnf versionlock delete kernel*
sudo dnf update -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

PyTorch Depreciação do Anaconda Channel

[A partir da PyTorch versão 2.6, PyTorch o suporte para o Conda foi descontinuado \(veja o anúncio oficial\)](#). Como resultado, a versão PyTorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o PyTorch venv, use a fonte/opt/pytorch/bin/activate

Data de lançamento: 2025-05-22

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU 2.7 PyTorch (Amazon Linux 2023) 20250520

Adicionado

- Lançamento inicial da série Deep Learning AMI GPU PyTorch 2.7 (Amazon Linux 2023). Incluindo um ambiente virtual Python pytorch (source/opt/pytorch/bin/activate) complementado com NVIDIA Driver R570, CUDA=12.8, cuDNN=9.10, NCCL=2.26.2 e EFA=1.40.0. PyTorch

AWS GPU OSS AMI PyTorch 2.7 de aprendizado profundo (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI GPU PyTorch 2.7 (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#)
- G4dn, G5, G5, G6, P4, P4de, P5, P5e, P5en, P6-B200

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: x86
- Kernel Linux: 6.8
- Driver NVIDIA: 570.133.20
- Pilha NVIDIA CUDA 12.8:
 - Diretórios de instalação CUDA, NCCL e cuDDN: /usr/local/cuda
 - Local dos testes NCCL:
 - all_reduce, all_gather e reduce_scatter:

```
/usr/local/cuda-12.8/efa/test-cuda-12.8/
```

- Para executar testes NCCL, o LD_LIBRARY_PATH já está atualizado com os caminhos necessários.
- PATHs Os comuns já foram adicionados ao LD_LIBRARY_PATH:

```
/opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/amazon/ofc-nccl/lib:/usr/local/lib:/usr/lib
```

- O LD_LIBRARY_PATH é atualizado com os caminhos da versão CUDA:

```
/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
```

- Versão compilada da NCCL:
 - Para o diretório CUDA de 12.8, compilou a versão NCCL 2.26.2+ .8 CUDA12
- CUDA padrão: 12,8
- PATH/usr/local/cudaaponta para CUDA 12.8
- Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda/targets/x86_64-linux/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/

- Nvidia GDRCopy: 2,5
- Motor Nvidia Transformer: 1.11.0
- AWS OFI NCCL: 1.14.2-aws
 - Caminho de instalação: /opt/amazon/ofi-nccl/. Path /opt/amazon/ofi-nccl/lib é adicionado ao LD_LIBRARY_PATH
- AWS CLI v2 em /usr/local/bin/aws
- Tipo de volume do EBS: gp3
- Kit de ferramentas de contêiner Nvidia: 1.17.7
 - Comando de versão: nvidia-container-cli -V
- Docker: 20.10.2
- Python: /usr/bin/python3
- Consulte o AMI-ID com o parâmetro SSM (a região de exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-pytorch-2.7-
  ubuntu-22.04/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters
  'Name=name,Values=Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Ubuntu
  22.04) ????????' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
  &CreationDate))[:1].ImageId' --output text
```

Avisos

Atenção instantânea

- O Flash attention ainda não tem um [lançamento oficial para PyTorch 2.7](#). Por esse motivo, ele é removido temporariamente dessa AMI. Depois que um lançamento oficial for feito para o PyTorch 2.7, nós o incluiremos nessa AMI.
- Sem a atenção do flash, o motor do transformador usa como padrão a atenção fundida cuDNN. Atualmente, existem problemas conhecidos com a atenção combinada e o Blackwell GPUs, como instâncias P6-B200.

- “Com o recurso computacional sm10.0 (arquitetura Blackwell) GPUs, o FP8 tipo de dados com atenção escalonada ao produto contém um impasse que faz com que o kernel pare em algumas circunstâncias, como quando o tamanho do problema é grande ou a GPU está executando vários kernels simultaneamente. Uma correção está planejada para uma versão futura.” [Notas de [lançamento do cuDNN 9.10.0](#)]
- Para usuários que desejam executar instâncias P6-B200 com FP8 dados e atenção escalonada ao produto, considere instalar o flash attention manualmente.

Instâncias P6-B200

- As instâncias P6-B200 exigem CUDA versão 12.8 ou superior e driver NVIDIA 570 ou drivers mais recentes.
- O P6-B200 contém 8 placas de interface de rede e pode ser iniciado usando o seguinte comando CLI AWS :

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces ""NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
  ....
  ....
  ....
  "NetworkCardIndex=7,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Instâncias P5/P5e

- DeviceIndex é exclusivo para cada um NetworkCard e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1. Abaixo está um exemplo do comando de inicialização da instância EC2 P5 usando awscli exibido NetworkCardIndex para

os números de 0 a 31 e DeviceIndex como 0 para a primeira interface e 1 para as 31 interfaces restantes.

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces ""NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    ....
    ....
    ....
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
echo linux-aws hold | sudo dpkg --set-selections
echo linux-headers-aws hold | sudo dpkg --set-selections
echo linux-image-aws hold | sudo dpkg --set-selections
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
echo linux-aws install | sudo dpkg --set-selections
echo linux-headers-aws install | sudo dpkg --set-selections
echo linux-image-aws install | sudo dpkg --set-selections
apt-get upgrade -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

PyTorch Depreciação do Anaconda Channel

[A partir da PyTorch versão 2.6, PyTorch o suporte para o Conda foi descontinuado \(veja o anúncio oficial\)](#). Como resultado, a versão PyTorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o PyTorch venv, use a fonte/opt/pytorch/bin/activate

Data de lançamento: 2025-06-03

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Ubuntu 22.04) 20250602

Adicionado

- Lançamento inicial da série Deep Learning AMI GPU PyTorch 2.7 (Ubuntu 22.04). Incluindo um ambiente virtual Python pytorch (source/opt/pytorch/bin/activate) complementado com NVIDIA Driver R570, CUDA=12.8, cuDNN=9.10, NCCL=2.26.5 e EFA=1.40.0. PyTorch

Problemas conhecidos

- “Com o recurso computacional sm10.0 (arquitetura Blackwell) GPUs, o FP8 tipo de dados com atenção escalonada ao produto contém um impasse que faz com que o kernel pare em algumas circunstâncias, como quando o tamanho do problema é grande ou a GPU está executando vários kernels simultaneamente. Uma correção está planejada para uma versão futura.” [Notas de [lançamento do cuDNN 9.10.0](#)]
- Para usuários que desejam executar instâncias P6-B200 com FP8 dados e atenção escalonada ao produto, considere instalar o flash attention manualmente.

AWS AMI GPU PyTorch 2.6 de aprendizado profundo (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS NVIDIA de aprendizado profundo AMI GPU PyTorch 2.6.0 (Amazon Linux 2023) \$ {YYY-MM-DD}

EC2 Instâncias suportadas:

- Consulte [Alterações importantes no DLAMI](#)
- Aprendizado profundo com OSS O driver NVIDIA suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

A AMI inclui o seguinte:

- AWS Serviço suportado: EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: x86
- Pilha NVIDIA CUDA12 6.6:
 - Caminho de instalação de CUDA, NCCL e cuDDN: /usr/local/cuda
 - CUDA padrão: 12,6
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12,6/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL compilada para 12.6:2.24.3
- Local dos testes NCCL:
 - all_reduce, all_gather e reduce_scatter: /usr/local/cuda-xx.x/efa/test
 - Para executar testes NCCL, o LD_LIBRARY_PATH já está atualizado com os caminhos necessários.
 - PATHs Os comuns já foram adicionados ao LD_LIBRARY_PATH:
 - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
 - LD_LIBRARY_PATH é atualizado com caminhos de versão CUDA
 - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- Instalador EFA: 1.38.0
- Nvidia GDRCopy: 2.4.1
- AWS OFI NCCL: 1.13.2-aws
 - AWS O OFI NCCL agora suporta várias versões do NCCL com uma única compilação
 - O caminho de instalação: /opt/amazon/of-nccl/. Path /opt/amazon/of-nccl/lib é adicionado ao LD_LIBRARY_PATH.
- Versão Python: 3.12
- Python: /opt/pytorch/bin/python
- Driver NVIDIA: 570.86.15
- AWS CLI v2 em /usr/bin/aws

- Tipo de volume do EBS: gp3
- NVMe Local do armazenamento de instâncias (em [EC2 instâncias compatíveis](#)):/opt/dlami/nvme
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-pytorch-2.6-
amazon-linux-2023/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI
GPU PyTorch 2.6.? (Amazon Linux 2023) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```

Avisos

PyTorch Depreciação do Anaconda Channel

[A partir da PyTorch versão 2.6, PyTorch o suporte para o Conda foi descontinuado \(veja o anúncio oficial\)](#). Como resultado, a versão PyTorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o PyTorch venv, use as instâncias source `opt/pytorch/bin/activate /p5/` `P5e`:

- DeviceIndex é exclusivo para cada um NetworkCard e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1. Abaixo está o exemplo do comando de inicialização da instância EC2 P5 usando awscli que aparece NetworkCardIndex do número 0-31 e DeviceIndex como 0 para a primeira interface e DeviceIndex como 1 para as demais 31 interfaces.

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
```

```
--image-id $AMI --key-name $KEYNAME \
--iam-instance-profile "Name=dlami-builder" \
--tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
--network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    ...
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
sudo dnf versionlock kernel*
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
sudo dnf versionlock delete kernel*
sudo dnf update -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-02-21

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6.0 (Amazon Linux 2023)
20250220

Adicionado

- Lançamento inicial do Deep Learning OSS Nvidia Driver AMI GPU 2.6 PyTorch para Amazon Linux 2023

- A partir da versão PyTorch 2.6, o Pytorch descontinuou o suporte para o Conda. Como resultado, o Pytorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o pytorch venv, use a fonte/opt/pytorch/bin/activate

AWS AMI GPU PyTorch 2.6 de aprendizado profundo (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI PyTorch GPU 2.6. \$ {PATCH-VERSION} (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: x86
- Python:/opt/pytorch/bin/python
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 570.86.15
- Pilha NVIDIA CUDA12 4.1:
 - Caminho de instalação de CUDA, NCCL e cuDDN:/-12.6/ usr/local/cuda
 - CUDA padrão: 12,6
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12,6/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib

- PATH para `ter/usr/local/cuda/bin:/usr/local/cuda/include/`
- Versão NCCL do sistema compilado presente em `/usr/local/cuda/`: 2.24.3
- PyTorch Versão NCCL compilada do ambiente PyTorch conda: 2.21.5
- Local dos testes NCCL:
 - `all_reduce`, `all_gather` e `reduce_scatter`: `/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test`
 - Para executar testes NCCL, o `LD_LIBRARY_PATH` já está atualizado com os caminhos necessários.
 - PATHs Os comuns já foram adicionados ao `LD_LIBRARY_PATH`:
 - `/opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib`
 - `LD_LIBRARY_PATH` é atualizado com caminhos de versão CUDA
 - `/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib`
- Instalador EFA: 1.38.0
- Nvidia GDRCopy: 2.4.1
- Motor Nvidia Transformer: v1.11.0
- AWS OFI NCCL: 1.13.2-aws
 - O caminho de instalação: `/opt/aws-ofi-nccl/` . Path `/opt/aws-ofi-nccl/lib` é adicionado ao `LD_LIBRARY_PATH`.
 - Nota: o PyTorch pacote também vem com o plug-in AWS OFI NCCL vinculado dinamicamente como um pacote conda e PyTorch usará esse `aws-ofi-nccl-dlc` pacote em vez do sistema OFI NCCL. AWS
- AWS CLI v2 como `aws2` e AWS CLI v1 como `aws`
- Tipo de volume do EBS: `gp3`
- Versão Python: 3.11
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é `us-east-1`):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-pytorch-2.6-
  ubuntu-22.04/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é `us-east-1`):

- Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI
GPU PyTorch 2.6.? (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```

Avisos

PyTorch Depreciação do Anaconda Channel

[A partir do PyTorch 2.6, o Pytorch descontinuou o suporte para o Conda \(veja o anúncio oficial\).](#)

Como resultado, o Pytorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o pytorch venv, use a fonte/opt/pytorch/bin/activate

Instâncias P5/P5e:

- DeviceIndex é exclusivo para cada um NetworkCard e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1. Abaixo está o exemplo do comando de inicialização da instância EC2 P5 usando awscli que aparece NetworkCardIndex do número 0-31 e DeviceIndex como 0 para a primeira interface e DeviceIndex como 1 para as demais 31 interfaces.

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
```

...

`"NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"`

Kernel

- A versão do kernel é fixada usando o comando:

```
echo linux-aws hold | sudo dpkg --set-selections
echo linux-headers-aws hold | sudo dpkg --set-selections
echo linux-image-aws hold | sudo dpkg --set-selections
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
echo linux-aws install | sudo dpkg --set-selections
echo linux-headers-aws install | sudo dpkg --set-selections
echo linux-image-aws install | sudo dpkg --set-selections
apt-get upgrade -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-02-21

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6.0 (Ubuntu 22.04) 20250220

Adicionado

- Lançamento inicial da série Deep Learning AMI GPU PyTorch 2.6 (Ubuntu 22.04). Incluindo um ambiente virtual Python pytorch (fonte/opt/pytorch/bin/activate), complementado com o driver NVIDIA R570, CUDA=12,6, cuDNN=9,7, NCCL=2,21,5 e EFA=1,38,0. PyTorch
- [A partir do PyTorch 2.6, o Pytorch descontinuou o suporte para o Conda \(veja o anúncio oficial\).](#) Como resultado, o Pytorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o pytorch venv, ative usando a fonte/opt/pytorch/bin/activate

AWS GPU AMI de aprendizado profundo PyTorch 2.5 (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

A AMI inclui o seguinte:

- AWS Serviço suportado: EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: x86
- Pilha NVIDIA CUDA12 4.4:
 - Caminho de instalação de CUDA, NCCL e cuDDN: /usr/local/cuda
 - CUDA padrão: 12,4
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12.4/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL compilada para 12.4:2.21.5
- Local dos testes NCCL:
 - all_reduce, all_gather e reduce_scatter: /usr/local/cuda-xx.x/efa/test
 - Para executar testes NCCL, o LD_LIBRARY_PATH já está atualizado com os caminhos necessários.
 - PATHs Os comuns já foram adicionados ao LD_LIBRARY_PATH:
 - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
 - LD_LIBRARY_PATH é atualizado com caminhos de versão CUDA

- Instalador EFA: 1.38.0
- Nvidia GDRCopy: 2.4.1
- AWS OFI NCCL: 1.13.2-aws
 - AWS O OFI NCCL agora suporta várias versões do NCCL com uma única compilação
 - O caminho de instalação: /opt/aws-ofi-nccl/. Path /opt/aws-ofi-nccl/lib é adicionado ao LD_LIBRARY_PATH.
 - Testa o caminho para o anel, message_transfer: /opt/aws-ofi-nccl/tests
- Versão Python: 3.11
- Python: /opt/conda/envs/pytorch/bin/python
- Driver NVIDIA: 560.35.03
- AWS CLI v2 em /usr/bin/aws
- Tipo de volume do EBS: gp3
- NVMe Local do armazenamento de instâncias (em [EC2 instâncias compatíveis](#)): /opt/dlami/nvme
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \
    --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-
pytorch-2.5-amazon-linux-2023/latest/ami-id \
    --query "Parameter.Value" \
    --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
    --owners amazon --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI
GPU PyTorch 2.5.? (Amazon Linux 2023) ????????' 'Name=state,Values=available' \
    --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
    --output text
```

Avisos

Instâncias P5/P5e:

- `DeviceIndex` é exclusivo para cada um `NetworkCard` e deve ser um número inteiro não negativo menor que o limite de ENIs per. `NetworkCard` Em P5, o número de ENIs per `NetworkCard` é 2, o que significa que os únicos valores válidos para `DeviceIndex` são 0 ou 1. Abaixo está o exemplo do comando de inicialização da instância EC2 P5 usando `awscli` que aparece `NetworkCardIndex` do número 0-31 e `DeviceIndex` como 0 para a primeira interface e `DeviceIndex` como 1 para as demais 31 interfaces.

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    ...
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
sudo dnf versionlock kernel*
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
sudo dnf versionlock delete kernel*
sudo dnf update -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Amazon Linux 2023)
20250216

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-08

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Amazon Linux 2023)
20250107

Adicionado

- Foi adicionado suporte para [instâncias G4dn](#).

Data de lançamento: 2024-11-21

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Amazon Linux 2023)
20241120

Adicionado

- Lançamento inicial do Deep Learning OSS Nvidia Driver AMI GPU 2.5 PyTorch para Amazon Linux 2023

Problemas conhecidos

- Esse DLAMI não oferece suporte às instâncias G4dn e EC2 G5 no momento. AWS está ciente de uma incompatibilidade que pode resultar em falhas de inicialização do CUDA, afetando as famílias de instâncias G4dn e G5 ao usar os drivers NVIDIA de código aberto junto com um kernel Linux versão 6.1 ou mais recente. Esse problema afeta distribuições Linux, como Amazon Linux 2023, Ubuntu 22.04 ou mais recente, ou SUSE Linux Enterprise Server 15 SP6 ou mais recente, entre outras.

AWS GPU AMI de aprendizado profundo PyTorch 2.5 (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI PyTorch GPU 2.5. \$ {PATCH_VERSION} (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: x86
- Python:/opt/conda/envs/pytorch/bin/python
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 550.144.03

- Pilha NVIDIA CUDA12 4.4:
 - Caminho de instalação de CUDA, NCCL e cuDDN: /usr/local/cuda-12.4/
 - CUDA padrão: 12.4
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12.4/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL do sistema compilado presente em/usr/local/cuda/: 2.21.5
 - PyTorch Versão NCCL compilada do ambiente PyTorch conda: 2.21.5
- Local dos testes NCCL:
 - all_reduce, all_gather e reduce_scatter: /usr/local/cuda-xx.x/efa/test
 - Para executar testes NCCL, o LD_LIBRARY_PATH já está atualizado com os caminhos necessários.
 - PATHs Os comuns já foram adicionados ao LD_LIBRARY_PATH:
 - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
 - LD_LIBRARY_PATH é atualizado com caminhos de versão CUDA
 - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
- Instalador EFA: 1.34.0
- Nvidia GDRCopy: 2.4.1
- Motor Nvidia Transformer: v1.11.0
- AWS OFI NCCL: 1.11.0-aws
 - O caminho de instalação: /opt/aws-ofi-nccl/. Path /opt/aws-ofi-nccl/lib é adicionado ao LD_LIBRARY_PATH.
 - Testa o caminho para o anel, message_transfer: /opt/aws-ofi-nccl/tests
 - Nota: o PyTorch pacote também vem com o plug-in AWS OFI NCCL vinculado dinamicamente como um pacote conda e PyTorch usará esse aws-ofi-nccl-dlc pacote em vez do sistema OFI NCCL. AWS
- AWS CLI v2 como aws2 e AWS CLI v1 como aws
- Tipo de volume do EBS: gp3

- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
- Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \
    --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-
pytorch-2.5-ubuntu-22.04/latest/ami-id \
    --query "Parameter.Value" \
    --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
- Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
    --owners amazon --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI
GPU PyTorch 2.5.? (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
    --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
    --output text
```

Avisos

Instâncias P5/P5e:

- DeviceIndex é exclusivo para cada um NetworkCard e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1. Abaixo está o exemplo do comando de inicialização da instância EC2 P5 usando awscli que aparece NetworkCardIndex do número 0-31 e DeviceIndex como 0 para a primeira interface e DeviceIndex como 1 para as demais 31 interfaces.

```
aws ec2 run-instances --region $REGION \
    --instance-type $INSTANCETYPE \
    --image-id $AMI --key-name $KEYNAME \
    --iam-instance-profile "Name=dlami-builder" \
    --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
    --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
```

```
"NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa" \
...
"NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Kernel

- A versão do kernel é fixada usando o comando:

```
echo linux-aws hold | sudo dpkg --set-selections
echo linux-headers-aws hold | sudo dpkg --set-selections
echo linux-image-aws hold | sudo dpkg --set-selections
```

- Recomendamos que os usuários evitem atualizar a versão do kernel (a menos que seja devido a um patch de segurança) para garantir a compatibilidade com os drivers instalados e as versões do pacote. Se os usuários ainda quiserem atualizar, eles podem executar os seguintes comandos para desafixar suas versões do kernel:

```
echo linux-aws install | sudo dpkg --set-selections
echo linux-headers-aws install | sudo dpkg --set-selections
echo linux-image-aws install | sudo dpkg --set-selections
apt-get upgrade -y
```

- Para cada nova versão do DLAMI, o kernel compatível mais recente disponível é usado.

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250216

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-21

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250119

Atualizado

- [O driver Nvidia foi atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025.](#)

Data de lançamento: 2024-11-21

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20241121

Adicionado

- Lançamento inicial da série Deep Learning AMI GPU PyTorch 2.4.1 (Ubuntu 22.04). Incluindo um ambiente conda pytorch complementado com NVIDIA Driver R550, CUDA=12.4.1, cuDNN=8.9.7, NCCL=2.21.5 e EFA=1.37.0. PyTorch

Fixo

- Devido a uma alteração no kernel do Ubuntu para solucionar defeitos na funcionalidade KASLR (Kernel Address Space Layout Randomization), as instâncias G4Dn/G5 não conseguem inicializar adequadamente o CUDA no driver OSS Nvidia. Para mitigar esse problema, esse DLAMI inclui uma funcionalidade que carrega dinamicamente o driver proprietário para instâncias G4Dn e G5. Aguarde um breve período de inicialização para esse carregamento para garantir que suas instâncias funcionem corretamente.
- Para verificar o status e a integridade desse serviço, você pode usar os seguintes comandos:

```
sudo systemctl is-active dynamic_driver_load.service
active
```


AWS GPU AMI de aprendizado profundo PyTorch 2.4 (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI PyTorch GPU 2.4. \$ {PATCH_VERSION} (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

A AMI inclui o seguinte:

- AWS Serviço suportado: EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: x86
- Python:/opt/conda/envs/pytorch/bin/python
- Controlador NVIDIA:
 - Driver OSS Nvidia: 550.144.03
- Pilha NVIDIA CUDA12 4.1:
 - Caminho de instalação de CUDA, NCCL e cuDDN:/-12.4/ usr/local/cuda
 - CUDA padrão: 12,4
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12.4/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH para ter/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL do sistema compilado presente em/usr/local/cuda/: 2.21.5
 - PyTorch Versão NCCL compilada do ambiente PyTorch conda: 2.20.5
- Local dos testes NCCL:

- `all_reduce`, `all_gather` e `reduce_scatter`: `./usr/local/cuda-xx.x/efa/test`
- Para executar testes NCCL, o `LD_LIBRARY_PATH` já está atualizado com os caminhos necessários.
 - PATHs Os comuns já foram adicionados ao `LD_LIBRARY_PATH`:
 - `/opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib`
 - `LD_LIBRARY_PATH` é atualizado com caminhos de versão CUDA
 - `/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib`
- Instalador EFA: 1.34.0
- Nvidia GDRCopy: 2.4.1
- Motor Nvidia Transformer: v1.11.0
- AWS OFI NCCL: 1.11.0-aws
 - O caminho de instalação: `/opt/aws-ofi-nccl/`. Path `/opt/aws-ofi-nccl/lib` é adicionado ao `LD_LIBRARY_PATH`.
 - Testa o caminho para o anel, `message_transfer:/opt/aws-ofi-nccl/tests`
 - Nota: o PyTorch pacote também vem com o plug-in AWS OFI NCCL vinculado dinamicamente como um pacote conda e também PyTorch usará esse `aws-ofi-nccl-dlc` pacote em vez do sistema OFI NCCL. AWS
- AWS CLI v2 como `aws2` e AWS CLI v1 como `aws`
- Tipo de volume do EBS: `gp3`
- Versão Python: 3.11
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é `us-east-1`):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \
    --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-
pytorch-2.4-ubuntu-22.04/latest/ami-id \
    --query "Parameter.Value" \
    --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é `us-east-1`):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
    --owners amazon \
```

```
--filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI GPU PyTorch
2.4.? (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
--output text
```

Avisos

Instâncias P5/P5e

- DeviceIndex é exclusivo para cada um NetworkCard e deve ser um número inteiro não negativo menor que o limite de ENIs per. NetworkCard Em P5, o número de ENIs per NetworkCard é 2, o que significa que os únicos valores válidos para DeviceIndex são 0 ou 1. Abaixo está o exemplo do comando de inicialização da instância EC2 P5 usando awscli que aparece NetworkCardIndex do número 0-31 e DeviceIndex como 0 para a primeira interface e DeviceIndex como 1 para as demais 31 interfaces.

```
aws ec2 run-instances --region $REGION \
  --instance-type $INSTANCETYPE \
  --image-id $AMI --key-name $KEYNAME \
  --iam-instance-profile "Name=dlami-builder" \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG}]" \
  --network-interfaces "NetworkCardIndex=0,DeviceIndex=0,Groups=$SG,SubnetId=
$SUBNET,InterfaceType=efa" \
    "NetworkCardIndex=1,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=2,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=3,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    "NetworkCardIndex=4,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
\
    ...
    "NetworkCardIndex=31,DeviceIndex=1,Groups=$SG,SubnetId=$SUBNET,InterfaceType=efa"
```

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20250216

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêineres, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Data de lançamento: 2025-01-21

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20250119

Atualizado

- [O driver Nvidia foi atualizado da versão 550.127.05 para 550.144.03 para abordar os CVEs presentes no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025.](#)

Data de lançamento: 2024-11-18

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20241116

Fixo

- Devido a uma alteração no kernel do Ubuntu para solucionar defeitos na funcionalidade KASLR (Kernel Address Space Layout Randomization), as instâncias G4Dn/G5 não conseguem inicializar adequadamente o CUDA no driver OSS Nvidia. Para mitigar esse problema, esse DLAMI inclui uma funcionalidade que carrega dinamicamente o driver proprietário para instâncias G4Dn e G5. Aguarde um breve período de inicialização para esse carregamento para garantir que suas instâncias funcionem corretamente.
- Para verificar o status e a integridade desse serviço, você pode usar os seguintes comandos:

```
sudo systemctl is-active dynamic_driver_load.service
active
```

Data de lançamento: 2024-10-16

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20241016

Adicionado

- [Foi adicionada a Nvidia TransformerEngine v1.11.0 para acelerar os modelos Transformer \(para obter mais detalhes, consulte transformer- .html\) https://docs.nvidia.com/deeplearning/engine/user-guide/index](https://docs.nvidia.com/deeplearning/engine/user-guide/index)

Data de lançamento: 2024-09-30

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20240929

Atualizado

- [Atualizou o Nvidia Container Toolkit da versão 1.16.1 para a 1.16.2, abordando a vulnerabilidade de segurança CVE-2024-0133.](#)

Data de lançamento: 2024-09-26

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20240925

Adicionado

- Lançamento inicial da série Deep Learning AMI GPU PyTorch 2.4.1 (Ubuntu 22.04). Incluindo um ambiente conda pytorch complementado com NVIDIA Driver R550, CUDA=12.4.1, cuDNN=8.9.7, NCCL=2.20.5 e EFA=1.34.0. PyTorch

ARM64 PyTorch Notas de lançamento do DLAMI

Abaixo estão as notas de lançamento de ARM64 PyTorch DLAMIs:

GPU

- [AWS ARM64 AMI GPU PyTorch 2.7 de aprendizado profundo \(Amazon Linux 2023\)](#)
- [AWS ARM64 AMI GPU PyTorch 2.7 de aprendizado profundo \(Ubuntu 22.04\)](#)
- [AWS ARM64 AMI GPU PyTorch 2.6 de aprendizado profundo \(Amazon Linux 2023\)](#)
- [AWS ARM64 AMI GPU PyTorch 2.6 de aprendizado profundo \(Ubuntu 22.04\)](#)

- [AWS GPU ARM64 AMI de aprendizado profundo PyTorch 2.5 \(Ubuntu 22.04\)](#)
- [AWS GPU ARM64 AMI de aprendizado profundo PyTorch 2.4 \(Ubuntu 22.04\)](#)

AWS Neuron

- Consulte o Guia do usuário do [Neuron DLAMI](#)

AWS ARM64 AMI OSS GPU PyTorch 2.7 de aprendizado profundo (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 AMI OSS Nvidia Driver GPU 2.7 de aprendizado profundo PyTorch (Amazon Linux 2023) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: ARM64
- Kernel Linux: 6.12
- Driver NVIDIA: 570.133.20
- Pilha NVIDIA CUDA 12.8:
 - Diretórios de instalação CUDA, NCCL e cuDDN: /usr/local/cuda
 - CUDA padrão: 12,8
 - PATH/usr/local/cudaaponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa
 - PATH para ter /usr/local/cuda/bin:/usr/local/cuda/include/

- AWS CLI v2 em/usr/local/bin/aws
- Tipo de volume do EBS: gp3
- Kit de ferramentas de contêiner Nvidia: 1.17.7
 - Comando de versão: nvidia-container-cli -V
- Docker: 25.0.8
- Python:/3.12 usr/bin/python
- Consulte o AMI-ID com o parâmetro SSM (a região de exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.7-
amazon-linux-2023/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters
'Name=name,Values=Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Amazon
Linux 2023) ????????' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
&CreationDate))[1].ImageId' --output text
```

Avisos

PyTorch Depreciação do Anaconda Channel

[A partir da PyTorch versão 2.6, PyTorch o suporte para o Conda foi descontinuado \(veja o anúncio oficial\).](#) Como resultado, a versão PyTorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o PyTorch venv, use a fonte/opt/pytorch/bin/activate

Data de lançamento: 2025-05-22

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU 2.7 PyTorch (Amazon Linux 2023) 20250521

Adicionado

- Lançamento inicial da série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Amazon Linux 2023). Incluindo um ambiente virtual Python pytorch (fonte/opt/pytorch/bin/activate) complementado com NVIDIA Driver R570, CUDA=12,8, cuDNN=9,10 e NCCL=2.26.2 PyTorch

AWS ARM64 AMI OSS GPU PyTorch 2.7 de aprendizado profundo (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 de aprendizado profundo (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: ARM64
- Kernel Linux: 6.8
- Driver NVIDIA: 570.133.20
- Pilha NVIDIA CUDA 12.8:
 - Diretórios de instalação CUDA, NCCL e cuDDN: /usr/local/cuda
 - CUDA padrão: 12.8
 - PATH/usr/local/cuda aponta para CUDA 12.8
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter /usr/local/cuda/lib64:/usr/local/cuda/lib:/usr/local/cuda/targets/sbsa
 - PATH para ter /usr/local/cuda/bin:/usr/local/cuda/include/
- AWS CLI v2 em /usr/local/bin/aws
- Tipo de volume do EBS: gp3
- Kit de ferramentas de contêiner Nvidia: 1.17.7
 - Comando de versão: nvidia-container-cli -V
- Docker: 28.2.2
- Python: /usr/bin/python3.12

- Consulte o AMI-ID com o parâmetro SSM (a região de exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \
    --name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.7-
ubuntu-22.04/latest/ami-id \
    --query "Parameter.Value" \
    --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters
'Name=name,Values=Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Ubuntu
22.04) ????????' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
&CreationDate))[:1].ImageId' --output text
```

Avisos

PyTorch Depreciação do Anaconda Channel

[A partir da PyTorch versão 2.6, PyTorch o suporte para o Conda foi descontinuado \(veja o anúncio oficial\).](#) Como resultado, a versão PyTorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o PyTorch venv, use a fonte/opt/pytorch/bin/activate

Data de lançamento: 2025-05-22

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Ubuntu 22.04) 20250521

Adicionado

- Lançamento inicial da série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Ubuntu 22.04). Incluindo um ambiente virtual Python pytorch (fonte/opt/pytorch/bin/activate) complementado com NVIDIA Driver R570, CUDA=12,8, cuDNN=9,10 e NCCL=2.26.2 PyTorch

AWS ARM64 AMI GPU PyTorch 2.6 de aprendizado profundo (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 AMI OSS Nvidia Driver PyTorch GPU 2.6 de aprendizado profundo. \$ {PATCH-VERSION} (Amazon Linux 2023) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g

A AMI inclui o seguinte:

- AWS Serviço suportado: EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: ARM64
- Python:/opt/pytorch/bin/python
- Versão Python: 3.12
- Controlador NVIDIA:
 - Driver OSS Nvidia: 570.86.15
- Pilha NVIDIA CUDA12 6.6:
 - Caminho de instalação de CUDA, NCCL e cuDDN:/-12.6/ usr/local/cuda
 - CUDA padrão: 12,6
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12,6/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter/64 usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL do sistema compilado presente em/usr/local/cuda/: 2.24.3
 - PyTorch Versão NCCL compilada do ambiente PyTorch conda: 2.21.5
- AWS CLI v2 como aws2 e AWS CLI v1 como aws
- Tipo de volume do EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \
```

```
--name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.6-
amazon-linux-2023/latest/ami-id \
--query "Parameter.Value" \
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 \
--owners amazon \
--filters 'Name=name,Values=Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch
2.6.? (Amazon Linux 2023) ????????' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
--output text
```

Avisos

PyTorch Depreciação do Anaconda Channel

[A partir da PyTorch versão 2.6, PyTorch o suporte para o Conda foi descontinuado \(veja o anúncio oficial\)](#). Como resultado, a versão PyTorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o PyTorch venv, use a fonte/opt/pytorch/bin/activate

Data de lançamento: 2025-02-21

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Amazon Linux 2023) 20250221

Adicionado

- Lançamento inicial da série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Amazon Linux 2023). Incluindo um ambiente virtual Python pytorch (fonte/opt/pytorch/bin/activate/), complementado com o driver NVIDIA R570, CUDA=12,6, cuDNN=9,7, NCCL = 2,21,5. PyTorch

AWS ARM64 AMI GPU PyTorch 2.6 de aprendizado profundo (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 AMI OSS NVIDIA Driver PyTorch GPU 2.6 de aprendizado profundo. \$ {PATCH-VERSION} (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: ARM64
- Python:/opt/pytorch/bin/python
- Versão Python: 3.12
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 570.86.15
- Pilha NVIDIA CUDA12 6.6:
 - Caminho de instalação de CUDA, NCCL e cuDDN:/-12.6/ usr/local/cuda
 - CUDA padrão: 12,6
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12,6/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter/64 usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL do sistema compilado presente em/usr/local/cuda/: 2.24.3
 - PyTorch Versão NCCL compilada do ambiente PyTorch venv: 2.21.5
- AWS CLI v2 como aws2 e AWS CLI v1 como aws
- Tipo de volume do EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região de exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \
```

```
--name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.6-ubuntu-22.04/latest/ami-id \
--query "Parameter.Value" \
--output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 \
--owners amazon --filters 'Name=name,Values=Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.? (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
--output text
```

Avisos

PyTorch Depreciação do Anaconda Channel

[A partir da PyTorch versão 2.6, PyTorch o suporte para o Conda foi descontinuado \(veja o anúncio oficial\)](#). Como resultado, a versão PyTorch 2.6 e versões posteriores passarão a usar ambientes virtuais Python. Para ativar o PyTorch venv, use a fonte/opt/pytorch/bin/activate

Data de lançamento: 2025-02-21

Nome da AMI: Deep Learning ARM64 AMI OSS NVIDIA Driver GPU PyTorch 2.6.0 (Ubuntu 22.04) 20250221

Adicionado

- Lançamento inicial da série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Ubuntu 22.04). Incluindo um ambiente virtual Python pytorch (fonte/opt/pytorch/bin/activate), complementado com o driver NVIDIA R570, CUDA=12,6, cuDNN=9,7, NCCL = 2,21,5. PyTorch

AWS GPU ARM64 AMI de aprendizado profundo PyTorch 2.5 (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 AMI OSS Nvidia Driver PyTorch GPU 2.5 de aprendizado profundo. \$ {PATCH-VERSION} (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: ARM64
- Python:/opt/conda/envs/pytorch/bin/python
- Versão Python: 3.11
- Controlador NVIDIA:
 - Driver OSS Nvidia: 550.144.03
- Pilha NVIDIA CUDA12 4.4:
 - Caminho de instalação de CUDA, NCCL e cuDDN:/-12.4/ usr/local/cuda
 - CUDA padrão: 12,4
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12.4/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter/64 usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
 - PATH para ter/usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL do sistema compilado presente em/usr/local/cuda/: 2.21.5
 - PyTorch Versão NCCL compilada do ambiente PyTorch conda: 2.21.5
- AWS CLI v2 como aws2 e AWS CLI v1 como aws
- Tipo de volume do EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \  
  --name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.5-  
ubuntu-22.04/latest/ami-id \  
  --query "Parameter.Value" \  
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon --filters \
    'Name=name,Values=Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.?
  (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04)
20250215

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêineres, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-21

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04)
20250117

Atualizado

- [O driver Nvidia foi atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025.](#)

Data de lançamento: 2024-11-22

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04) 20241122

Adicionado

- Lançamento inicial da série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04). Incluindo um ambiente conda pytorch complementado com o NVIDIA Driver R550, CUDA=12.4, cuDNN=8.9.7, NCCL=2.21.5. PyTorch

AWS GPU ARM64 AMI de aprendizado profundo PyTorch 2.4 (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- ARM64 AMI OSS Nvidia Driver PyTorch GPU 2.4 de aprendizado profundo. \$ {PATCH-VERSION} (Ubuntu 22.04) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- G5g

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: ARM64
- Python:/opt/conda/envs/pytorch/bin/python
- Versão Python: 3.11
- Controlador NVIDIA:

- Driver OSS Nvidia: 550.144.03
- Pilha NVIDIA CUDA12 4.1:
 - Caminho de instalação de CUDA, NCCL e cuDDN: /usr/local/cuda
 - CUDA padrão: 12,4
 - CAMINHO/usr/local/cuda points to /usr/local/cuda-12.4/
 - Variáveis de ambiente atualizadas abaixo:
 - LD_LIBRARY_PATH deve ter /usr/local/cuda/lib64:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib64:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
 - PATH para ter /usr/local/cuda/bin:/usr/local/cuda/include/
 - Versão NCCL do sistema compilado presente em /usr/local/cuda/: 2.21.5
 - PyTorch Versão NCCL compilada do ambiente PyTorch conda: 2.20.5
- AWS CLI v2 como aws2 e AWS CLI v1 como aws
- Tipo de volume do EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.4-
ubuntu-22.04/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon \
  --filters 'Name=name,Values=Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch
2.4.? (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \
  --output text
```

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04)
20250215

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - No Container Toolkit versão 1.17.4, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-21

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) 20250117

Atualizado

- [O driver Nvidia foi atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025.](#)

Data de lançamento: 2024-09-30

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) 20240927

Atualizado

- [Atualizou o Nvidia Container Toolkit da versão 1.16.1 para a 1.16.2, abordando a vulnerabilidade de segurança CVE-2024-0133.](#)

Data de lançamento: 2024-09-26

Nome da AMI: Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) 20240926

Adicionado

- Lançamento inicial da série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04). Incluindo um ambiente conda pytorch complementado com o NVIDIA Driver R550, CUDA=12,4, cuDNN=8.9.7, NCCL=2.20.5. PyTorch

TensorFlow DLAMIs

TensorFlow Notas de lançamento do DLAMI

- [Notas de lançamento do X86 TensorFlow DLAMI](#)

Notas de lançamento do X86 TensorFlow DLAMI

Abaixo estão as notas de lançamento do X86 TensorFlow DLAMI:

GPU

- [AWS GPU AMI TensorFlow 2.18 de aprendizado profundo \(Amazon Linux 2023\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.18 \(Ubuntu 22.04\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.17 \(Ubuntu 22.04\)](#)

AWS Neuron

- Consulte o Guia [DLAMI do Neurônio](#).

GPU AMI TensorFlow 2.18 de aprendizado profundo (Amazon Linux 2023)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI GPU TensorFlow 2.18 (Amazon Linux 2023) \$ {YYYY-MM-DD}

EC2 Instâncias suportadas

- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2023
- Arquitetura de computação: x86
- Python:/3.12 opt/tensorflow/bin/python
- TensorFlow versão: 2.18
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 560.35.03
- Pilha NVIDIA CUDA12 :
 - Caminho de instalação de CUDA, NCCL e cuDDN:/-12.5/ usr/local/cuda
- Instalador EFA: 1.37.0
- AWS CLI v2 como aws2 e AWS CLI v1 como aws
- Tipo de volume do EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \  
  --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-  
tensorflow-2.18-amazon-linux-2023/latest/ami-id \  
  --query "Parameter.Value" \  
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \  
  --owners amazon \  
  --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI GPU TensorFlow  
2.18 (Amazon Linux 2023) ????????' 'Name=state,Values=available' \  
  --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \  

```

```
--output text
```

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023)
20241215

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêiner, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2024-12-09

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023)
20241206

Adicionado

- Lançamento inicial da série Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023).
 - O software inclui o seguinte:
 - “nvidia-driver = 560,35,03”
 - “gerenciador de tecidos = 560,35,03”
 - “cuda = 12,5”

- “cudnn = 9.5.1”
- “efa = 1,37,0”
- “nccl=2.23,4”
- “aws-nccl-ofi-plugin=v1.13.0-aws”
- O ambiente virtual Tensorflow (fonte do comando de ativação/opt/tensorflow/bin/activate) inclui o seguinte:
 - “tensorflow = 2.18.0”

GPU AMI de aprendizado profundo TensorFlow 2.18 (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI GPU TensorFlow 2.18 (Ubuntu 22.04) \$ {AAAA-MM-DD}

EC2 Instâncias suportadas

- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en.

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: x86
- Python:/3.12 opt/tensorflow/bin/python
- TensorFlow versão: 2.18
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 550.144.03
- Pilha NVIDIA CUDA12 :
 - Caminho de instalação de CUDA, NCCL e cuDDN:/-12.5/ usr/local/cuda
- Instalador EFA: 1.37.0

- AWS CLI v2 como `aws2` e AWS CLI v1 como `aws`
- Tipo de volume do EBS: `gp3`
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é `us-east-1`):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \  
    --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-  
tensorflow-2.18-ubuntu-22.04/latest/ami-id \  
    --query "Parameter.Value" \  
    --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é `us-east-1`):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \  
    --owners amazon --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI  
GPU TensorFlow 2.18 (Ubuntu 22.04) ????????' 'Name=state,Values=available' \  
    --query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' \  
    --output text
```

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04)
20250215

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêineres, certifique-se de atualizar seu `LD_LIBRARY_PATH` para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-20

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04)
20250118

Atualizado

- [Driver Nvidia atualizado da versão 550.90.07 para 550.127.05 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025](#)

Data de lançamento: 2024-12-09

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04)
20241206

Adicionado

- Lançamento inicial da série Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04).
 - O software inclui o seguinte:
 - "nvidia-driver = 550.127.05"
 - "gerenciador de tecidos = 550,127,05"
 - "cuda = 12,5"
 - "cudnn = 9.5.1"
 - "efa = 1,37,0"
 - "nccl=2.23,4"
 - "aws-nccl-ofi-plugin=v1.13.0-aws"
 - O ambiente virtual Tensorflow (fonte do comando de ativação/opt/tensorflow/bin/activate) inclui o seguinte:
 - "tensorflow = 2.18.0"

Fixo

- Devido a uma alteração no kernel do Ubuntu para solucionar defeitos na funcionalidade KASLR (Kernel Address Space Layout Randomization), as instâncias G4Dn/G5 não conseguem inicializar adequadamente o CUDA no driver OSS Nvidia. Para mitigar esse problema, esse DLAMI inclui uma funcionalidade que carrega dinamicamente o driver proprietário para instâncias G4Dn e G5. Aguarde um breve período de inicialização para esse carregamento para garantir que suas instâncias funcionem corretamente.
- Para verificar o status e a integridade desse serviço, você pode usar os seguintes comandos:

```
sudo systemctl is-active dynamic_driver_load.service
active
```

GPU AMI de aprendizado profundo TensorFlow 2.17 (Ubuntu 22.04)

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Driver OSS Nvidia de aprendizado profundo AMI GPU TensorFlow 2.17 (Ubuntu 22.04) \$ {AAAA-MM-DD}

EC2 Instâncias suportadas

- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e. 5caneta.

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Ubuntu 22.04
- Arquitetura de computação: x86
- Python:/3.12 opt/tensorflow/bin/python
- TensorFlow versão: 2.17
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 550.144.03

- Pilha NVIDIA CUDA12 :
 - Caminho de instalação de CUDA, NCCL e cuDDN: /usr/local/cuda
- Instalador EFA: 1.34.0
- AWS CLI v2 como aws2 e AWS CLI v1 como aws
- Tipo de volume do EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região do exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --region us-east-1 \
  --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-
tensorflow-2.17-ubuntu-22.04/latest/ami-id \
  --query "Parameter.Value" \
  --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 \
  --owners amazon --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI
GPU TensorFlow 2.17 (Ubuntu 22.04) ????????' 'Name=state,Values=available' \
  --query 'reverse(sort_by(Images, &CreationDate))[1].ImageId' \
  --output text
```

Data de lançamento: 2025-02-17

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04)
20250215

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêineres, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas

bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial [Se](#) você usar uma camada de compatibilidade CUDA.

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-01-20

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04)
20250118

Atualizado

- [Driver Nvidia atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025](#)

Versão 2.17.1

Data de lançamento: 2024-11-18

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04)
20241115

Fixo

- Devido a uma alteração no kernel do Ubuntu para solucionar um defeito na funcionalidade KASLR (Kernel Address Space Layout Randomization), as instâncias G4Dn/G5 não conseguem inicializar adequadamente o CUDA no driver OSS Nvidia. Para mitigar esse problema, esse DLAMI inclui uma funcionalidade que carrega dinamicamente o driver proprietário para instâncias G4Dn e G5. Aguarde um breve período de inicialização para esse carregamento para garantir que suas instâncias funcionem corretamente.
- Para verificar o status e a integridade desse serviço, você pode usar os seguintes comandos:

```
sudo systemctl is-active dynamic_driver_load.service  
active
```

Versão 2.17.0

Data de lançamento: 2024-09-25

Nome da AMI: Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20240924

Adicionado

- Lançamento inicial da série Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04).
- O software inclui o seguinte:
 - “nvidia-driver = 550,90,07”
 - “gerenciador de tecidos = 550,90,07”
 - “cuda = 12,3”
 - “cudnn = 8,9,7”
 - “efa = 1,34,0”
 - “nccl=2.22.3”
 - “aws-nccl-ofi-plugin=v1.11.0-aws”
- O ambiente virtual Tensorflow (fonte do comando de ativação/opt/tensorflow/bin/activate) inclui o seguinte:
 - “tensorflow = 2.17.0”

Notas de lançamento do Multi-Framework DLAMIs

Tip

Se você usa apenas um framework de machine learning, é recomendado usar uma [DLAMI de framework único](#).

Notas de lançamento do Multi Framework DLAMI

- [Notas de lançamento do DLAMI Multi-Framework](#)

Notas de lançamento do DLAMI Multi-Framework

Abaixo estão as notas de lançamento do Multi-Framework X86 DLAMI:

GPU

- [AWS AMI de aprendizado profundo \(Amazon Linux 2\)](#)

AWS Neuron

- Consulte o Guia do usuário do [Neuron DLAMI](#)

AWS AMI de aprendizado profundo (Amazon Linux 2)

Tip

Os clientes que usam uma estrutura única gostam PyTorch ou TensorFlow são incentivados a usar a estrutura única DLAMIs mencionada [aqui](#).

Para obter ajuda para começar, consulte [Conceitos básicos da DLAMI](#).

Formato de nome da AMI

- Versão \$ {XX.X} do driver Nvidia Driver AMI (Amazon Linux 2) proprietário do Deep Learning
- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2), versão \$ {XX.X}

EC2 Instâncias suportadas

- Consulte [Alterações importantes no DLAMI](#).
- Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5
- O Deep Learning com driver proprietário da Nvidia suporta G3 (G3.16x não suportado), P3, P3dn

A AMI inclui o seguinte:

- AWS Serviço suportado: Amazon EC2
- Sistema operacional: Amazon Linux 2

- Arquitetura de computação: x86
- Estrutura de ambientes Conda e versões python:
 - AMI de driver OSS Nvidia de aprendizado profundo (Amazon Linux 2):
 - python3: Python 3.10
 - tensorflow2_p310:2.16, Python 3.10 TensorFlow
 - pytorch_p310:2.2, Python 3.10 PyTorch
 - AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2):
 - python3: Python 3.10
 - tensorflow2_p310:2.16, Python 3.10 TensorFlow
 - pytorch_p310:2.2, Python 3.10 PyTorch
- Controlador NVIDIA:
 - Controlador OSS Nvidia: 550.163.01
 - Driver proprietário da Nvidia: 550.163.01
- Pilha NVIDIA CUDA12 1.1-12.4:
 - Caminho de instalação de CUDA, NCCL e cuDDN: `usr/local/cuda`
 - CUDA padrão: 12.1
 - `PATH` para `usr/local/cuda` aponta para 1 CUDA12.
 - Variáveis de ambiente atualizadas abaixo:
 - `LD_LIBRARY_PATH` para `usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1:/usr/local/cuda-12.1/targets/x86_64-linux/lib`
 - `PATH` para `usr/local/cuda-12.1/bin:/usr/local/cuda-11.8/include/`
 - Para qualquer versão diferente do CUDA, atualize `LD_LIBRARY_PATH` adequadamente.
 - Versão NCCL compilada para CUDA 12.1-12.4: 2.22.3
 - Local dos testes NCCL:
 - `all_reduce`, `all_gather` e `reduce_scatter`: `usr/local/cuda-xx.x/efa/test`
 - Para executar testes NCCL, o `LD_LIBRARY_PATH` precisa ser aprovado com as atualizações abaixo.
 - `PATHs` Os comuns já foram adicionados ao `LD_LIBRARY_PATH`:
 - `/opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib`
 - Para qualquer versão diferente do CUDA, atualize `LD_LIBRARY_PATH` adequadamente.

- GDRCopy: 2,4
- AWS OFI NCCL: 1.13.2
 - Localização do sistema: /usr/local/cuda-xx.x/efa
 - Isso é adicionado para executar testes NCCL localizados em /usr/local/cuda-xx.x/efa/test
 - Além disso, o PyTorch pacote vem com o plug-in AWS OFI NCCL vinculado dinamicamente como um pacote conda e também PyTorch usará esse aws-ofi-nccl-dlc pacote em vez do sistema OFI NCCL. AWS
- Localização dos testes NCCL: /usr/local/cuda-xx.x/efa/test
- AWS CLI v2 em /usr/local/bin/aws2 e AWS CLI v1 em /usr/local/bin/aws
- Tipo de volume EBS: gp3
- Consulte o AMI-ID com o parâmetro SSM (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ssm get-parameter --name /aws/service/deeplearning/ami/x86_64/multi-framework-oss-nvidia-driver-amazon-linux-2/latest/ami-id --region us-east-1 --query "Parameter.Value" --output text
```

- Driver proprietário da Nvidia:

```
aws ssm get-parameter --name /aws/service/deeplearning/ami/x86_64/multi-framework-proprietary-nvidia-driver-amazon-linux-2/latest/ami-id --region us-east-1 --query "Parameter.Value" --output text
```

- Consulte o AMI-ID com AWSCLI (a região de exemplo é us-east-1):
 - Controlador OSS Nvidia:

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI (Amazon Linux 2) Version ??.' 'Name=state,Values=available' --query 'reverse(sort_by(Images, &CreationDate))[1].ImageId' --output text
```

- Driver proprietário da Nvidia:

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters 'Name=name,Values=Deep Learning Proprietary Nvidia Driver AMI (Amazon Linux 2)
```

```
Version ??.' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
&CreationDate))[:1].ImageId' --output text
```

Avisos

Atualizações do EFA de 1.37 para 1.38 (lançamento em 05/02/2025)

- O EFA agora inclui o plug-in AWS OFI NCCL, que agora pode ser encontrado em `/-ofi-nccl/.opt/amazon/ofi-nccl` rather than the original `/opt/aws`. Se estiver atualizando sua variável `LD_LIBRARY_PATH`, certifique-se de modificar a localização do OFI NCCL corretamente.

Remoção do ambiente Neuron Conda

- O driver Nvidia proprietário do Deep Learning AMIs lançado após 18 de julho de 2024 será enviado sem ambientes neuronconda para e. PyTorch TensorFlow Em vez disso, use o Neuron DLAMIs nas notas de [lançamento do DLAMI](#) para utilizar ambientes de neurônios.

Remoção do Audit Package

- Os DLAMI lançados entre 26 de março de 2024 (2024-03-26) e 12 de abril de 2024 (2024-04-12) foram enviados sem o pacote de auditoria. Se você precisar desse pacote específico para suas necessidades de registro e monitoramento, migre seus fluxos de trabalho para o DLAMI mais recente para consumir aqueles com o pacote de auditoria instalado.

Horovod

- O Horovod foi removido dos ambientes conda `pytorch_p310` e `tensorflow2_p310` atuais no DLAMI. Os clientes poderão instalar as bibliotecas do horovod seguindo as [diretrizes do horovod](#) e instalá-las em seus trabalhos de treinamento DLAMIs distribuídos.

Data de lançamento: 2025-04-22

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 81.2
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 81.2

Atualizado

- [Driver Nvidia atualizado da versão 550.144.03 para 550.163.01 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de abril de 2025](#)

Data de lançamento: 2025-02-17

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 80.6
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 80.4

Atualizado

- Kit de ferramentas de contêiner NVIDIA atualizado da versão 1.17.3 para a versão 1.17.4
 - Consulte a página de notas de lançamento aqui para obter mais informações: <https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4>
 - Na versão 1.17.4 do Container Toolkit, a montagem das bibliotecas de compatibilidade CUDA agora está desativada. [Para garantir a compatibilidade com várias versões do CUDA em fluxos de trabalho de contêineres, certifique-se de atualizar seu LD_LIBRARY_PATH para incluir suas bibliotecas de compatibilidade CUDA, conforme mostrado no tutorial “Se você usar uma camada de compatibilidade CUDA” aqui - -gpu-drivers.html# https://docs.aws.amazon.com/sagemaker/latest/dg/inference collapsible-cuda-compat](#)

Removido

- As bibliotecas de espaço do usuário cuobj e nvdiasm foram removidas fornecidas pelo kit de ferramentas [NVIDIA CUDA para abordar as CVEs presentes no boletim de segurança do kit de ferramentas NVIDIA](#) CUDA de 18 de fevereiro de 2025

Data de lançamento: 2025-02-05

Nomes da AMI

- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 80.2
- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 80.4

Atualizado

- Versão EFA atualizada de 1.37.0 para 1.38.0
 - O EFA agora inclui o plug-in AWS OFI NCCL, que agora pode ser encontrado em `/-ofi-nccl/.opt/amazon/ofi-nccl` rather than the original `/opt/aws`. Se estiver atualizando sua variável `LD_LIBRARY_PATH`, certifique-se de modificar a localização do OFI NCCL corretamente.

Data de lançamento: 2025-01-15

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 80.3
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 80.1

Atualizado

- [Driver Nvidia atualizado da versão 550.127.05 para 550.144.03 para o endereço CVEs presente no boletim de segurança do driver de vídeo GPU NVIDIA de janeiro de 2025](#)

Data de lançamento: 2024-12-09

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 80.1
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 79.9

Atualizado

- Kit de ferramentas de contêiner Nvidia atualizado da versão 1.17.0 para 1.17.3

Data de lançamento: 2024-11-11

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 79.9
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 79.7

Atualizado

- [Atualizou o Nvidia Container Toolkit da versão 1.16.2 para a 1.17.0, abordando a vulnerabilidade de segurança CVE-2024-0134.](#)

Data de lançamento: 2024-10-22

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 79.6
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 79.6

Atualizado

- [Driver Nvidia atualizado da versão 550.90.07 para 550.127.05 para endereço CVEs presente no Boletim de Segurança do NVIDIA GPU Display de outubro de 2024](#)

Data de lançamento: 2024-10-03

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 79.3
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 79.3

Atualizado

- [Atualizou o Nvidia Container Toolkit da versão 1.16.1 para a 1.16.2, abordando a vulnerabilidade de segurança CVE-2024-0133.](#)

Data de lançamento: 2024-07-18

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 78.6
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 78.7

Atualizado

- Os ambientes `aws_neuron_pytorch_p38` e `aws_neuron_tensorflow_p38` foram removidos da AMI de driver Nvidia proprietária de aprendizado profundo.
- O suporte à família de instâncias Inf1 foi removido da AMI proprietária do driver Nvidia de aprendizado profundo.

Data de lançamento: 2024-06-06

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 78.5
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 78.5

Atualizado

- Versão atualizada do driver Nvidia para 535.183.01 de 535.161.08

Data de lançamento: 2024-05-17

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 78.1
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 78.1

Atualizado

- O [torchserve](#) foi atualizado de v0.8.2 para v0.11.0 no ambiente [pytorch_p310](#).

Data de lançamento: 2024-05-07

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 78.0
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 78.0

Atualizado

- TensorFlow versão atualizada de 2.15 para 2.16 no ambiente tensorflow2_p310.
- Versão EFA atualizada da versão 1.30 para a versão 1.32
- Plugin AWS OFI NCCL atualizado da versão 1.7.4 para a versão 1.9.1
- [Kit de ferramentas de contêiner Nvidia atualizado da versão 1.13.5 para a versão 1.15.0](#)
 - NOTA: A versão 1.15.0 NÃO inclui os pacotes nvidia-docker2 nvidia-container-runtime e nvidia-docker2. É recomendável usar nvidia-container-toolkit pacotes diretamente seguindo os documentos do [kit de ferramentas de contêiner da Nvidia](#).

Adicionado

- Foi adicionada uma pilha CUDA12 3.0 com 3.3, NCCL CUDA12 2.21.5, cuDNN 8.9.7

Removido

- Foram removidas as pilhas de CUDA11 0,7, CUDA12 0,0 presentes em /-12,0 usr/local/cuda-11.7 and /usr/local/cuda
- [O pacote nvidia-docker2 e seu comando nvidia-docker foram removidos como parte da atualização do kit de ferramentas de contêiner Nvidia de 1.13.5 para 1.15.0, que NÃO inclui os pacotes nvidia-docker2. nvidia-container-runtime](#)

Data de lançamento: 2024-04-04

Nomes da AMI

- Aprendizado profundo OSS Nvidia Driver AMI (Amazon Linux 2) versão 77.0
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 77.0

Atualizado

- PyTorch versão atualizada de 2.1 para 2.2 no ambiente pytorch_p310.
- Para o driver OSS Nvidia DLAMIs, adicionou suporte às instâncias G6 e EC2 Gr6. Consulte a página de [seleção de EC2 instâncias](#) para obter mais informações.

Data de lançamento: 2024-03-29

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 76.8
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 76.9

Atualizado

- Driver Nvidia atualizado de 535.104.12 para 535.161.08 nos drivers Proprietário e OSS Nvidia. DLAMIs
- As novas instâncias suportadas para cada DLAMI são as seguintes:
 - O Deep Learning com driver proprietário da Nvidia suporta G3 (G3.16x não suportado), P3, P3dn, Inf1
 - Aprendizado profundo com OSS O driver Nvidia suporta G4dn, G5, P4d, P4de.

Removido

- Removido o suporte às EC2 instâncias G4dn, G5 e G3.16x do driver proprietário da Nvidia DLAMI.

Versão 76.8

Data de lançamento: 2024-03-20

Nomes da AMI

- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 76.8

Adicionado

- Foi adicionado [awscliv2](#) na AMI como/na AMI proprietária `usr/local/bin/aws2`, alongside `awscliv1` as `/usr/local/bin/aws` do driver Nvidia

Versão 76.7

Data de lançamento: 2024-03-20

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 76.7

Adicionado

- Foi adicionado [awscliv2](#) na AMI como /usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awsno OSS Nvidia Driver AMI
- O driver OSS Nvidia DLAMI atualizado com suporte para G4dn e G5, com base nele, o suporte atual é o seguinte:
 - A AMI de driver Nvidia proprietária da Deep Learning Base (Amazon Linux 2) suporta P3, P3dn, G3, G5, G4dn.
 - O Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) suporta G4dn, G5, P4, P5.
- Recomenda-se que o driver DLAMIs OSS Nvidia seja usado para G4dn, G5, P4, P5.

Versão 76.3

Data de lançamento: 2024-02-14

Atualizado

- Atualizado TensorFlow de 2.13.0 para 2.15.0
- EFA atualizado de 1.29.0 para 1.30.0
- AWS-OFI-NCCL atualizado de 1.7.3-aws para 1.7.4-aws
- Driver Nvidia atualizado para 535.104.12 na AMI de driver Nvidia proprietária de aprendizado profundo
- Driver Nvidia atualizado para 535.154.05 no Deep Learning OSS Nvidia Driver AMI

Versão 76.2

Data de lançamento: 2024-02-02

Nomes da AMI

- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 76.2
- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 76.4

Segurança

- [Versão atualizada do pacote runc para consumir o patch para CVE-2024-21626.](#)

Versão 76.1

Data de lançamento: 2023-12-27

Atualizado

- Atualizado PyTorch de 2.0.1 para 2.1.0

Versão 75.1

Data de lançamento: 2023-11-17

Consulte [Alterações importantes no DLAMI](#)

Nomes da AMI

- Driver AMI OSS Nvidia de aprendizado profundo (Amazon Linux 2) versão 75.1
- AMI de driver Nvidia proprietário de aprendizado profundo (Amazon Linux 2) versão 75.1

Adicionado

- AWS A AMI de aprendizado profundo (DLAMI) é dividida em dois grupos separados:
 - DLAMI que usa o driver proprietário da Nvidia (para suportar P3, P3dn, G3, G5, G4dn).
 - DLAMI que usa o driver Nvidia OSS para habilitar o EFA (para suportar P4, P5).
- Consulte o [anúncio público](#) para obter mais informações sobre a divisão do DLAMI.
- AWS As consultas cli acima estão nas [notas de lançamento](#) em bullet point Query AMI-ID com (a região de exemplo é us-east-1) AWSCLI

Atualizado

- EFA atualizado de 1.26.1 para 1.29.0
- GDRCopy atualizado de 2.3 para 2.4

Versão 74.4

Data de lançamento: 2023-10-27

Atualizado

- AWS Plugin OFI NCCL atualizado da versão 1.7.2 para a versão 1.7.3
- Diretórios CUDA 12.0-12.1 atualizados com NCCL versão 2.18.5
- CUDA12.1 atualizado como a versão padrão do CUDA
 - LD_LIBRARY_PATH atualizado para ter `/usr/local/cuda-12.1/targets/x86_64-linux/lib:/usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1` and PATH to have `/usr/local/cuda-12.1/bin`
 - Para clientes que desejam mudar para qualquer versão diferente do CUDA, defina as variáveis LD_LIBRARY_PATH e PATH adequadamente.
- Almofada atualizada da versão 9.4.0 para 10.1.0 para corrigir [SNYK-PYTHON-PILLOW-5918878](#) em todos os ambientes conda
- [opencv-python atualizado de 4.8.0.74 para 4.8.1.78 para corrigir SNYK-PYTHON-OPENCVPYTHON-5926695 em todos os ambientes conda](#)

Adicionado

- O Kernel Live Patching agora está habilitado. A aplicação dinâmica de patches permite que os clientes apliquem vulnerabilidades de segurança e correções de bugs críticos a um kernel Linux em execução, sem reinicializações ou interrupções na execução de aplicativos.
 - Observe que o suporte de patches ao vivo para o kernel 5.10.192 terminará em 30/11/23.
 - Para obter mais informações, consulte os AWS documentos oficiais aqui - <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/al2-live-patching.html>

Versão 74.0

Data de lançamento: 2023-07-19

Atualizado

- Atualizado TensorFlow de 2.12 para 2.13
 - O Horovod foi removido do ambiente conda nesta versão. Consulte o Aviso para obter detalhes sobre a instalação do horovod.

Versão 73.1

Data de lançamento: 2023-06-12

Atualizado

- Atualizado PyTorch de 2.0.0 para 2.0.1

Conceitos básicos da DLAMI

Este guia inclui dicas de como selecionar a DLAMI correta para você com a seleção de um tipo de instância que se adequa a seu caso de uso e a seu orçamento, e [Informações relacionadas sobre a DLAMI](#) que descreve configurações personalizadas que podem ser de seu interesse.

Se você está começando a usar AWS ou usar a Amazon EC2, comece com [AMI de aprendizado profundo com Conda](#) o. Se você está familiarizado com a Amazon EC2 e outros AWS serviços, como Amazon EMR, Amazon EFS ou Amazon S3, e está interessado em integrar esses serviços para projetos que precisam de treinamento distribuído ou inferência, [Informações relacionadas sobre a DLAMI](#) confira se um deles se adequa ao seu caso de uso.

Recomendamos que você confira [Escolher uma DLAMI](#) para ter uma ideia do tipo de instância que pode ser melhor para seu aplicativo.

Próxima etapa

[Escolher uma DLAMI](#)

Escolher uma DLAMI

Oferecemos uma variedade de opções de DLAMI, conforme mencionado nas notas de lançamento da [GPU DLAMI](#). Para ajudar você a selecionar a DLAMI correta para seu caso de uso, agrupamos as imagens pelo tipo de hardware ou pela funcionalidade para a qual elas foram desenvolvidas. Nossos agrupamentos de alto nível são:

- Tipo de DLAMI: base, estrutura única, estrutura múltipla (Conda DLAMI)
- Arquitetura de computação: Graviton baseado [em x86](#), [baseado em ARM64 AWS](#)
- Tipo de processador: [GPU](#), [CPU](#), [Inferentia](#), [Trainium](#)
- SDK: [CUDA](#), [Neurônio AWS](#)
- SISTEMA OPERACIONAL: Amazon Linux, Ubuntu

O restante dos tópicos deste guia ajudam você a obter mais informações e detalhes.

Tópicos

- [Instalações do CUDA e associações da estrutura](#)
- [AMI de deep learning base](#)

- [AMI de aprendizado profundo com Conda](#)
- [Opções de arquitetura de DLAMI](#)
- [Opções do sistema operacional da DLAMI](#)

A seguir

[AMI de aprendizado profundo com Conda](#)

Instalações do CUDA e associações da estrutura

O aprendizado profundo é de última geração, e cada estrutura oferece versões "estáveis". Essas versões estáveis podem não funcionar com a implementação e os recursos mais recentes do CUDA ou do cuDNN. O caso de uso e os recursos necessários podem ajudar você a escolher uma estrutura. Se você não tiver certeza, use a AMI de aprendizado profundo mais recente com o Conda. Ela tem binários pip oficiais para todos os frameworks com CUDA, usando a versão mais recente compatível com cada framework. Se você quiser as versões mais recentes e personalizar seu ambiente de aprendizado profundo, use a AMI base de deep learning.

Consulte o nosso guia em [Estabilidade e candidatos a lançamento](#) para obter mais orientações.

Escolher uma DLAMI com o CUDA

A [AMI de deep learning base](#) tem todas as séries de versões CUDA disponíveis.

A [AMI de aprendizado profundo com Conda](#) tem todas as séries de versões CUDA disponíveis.

Note

Não incluímos mais os ambientes MXNet CNTK, Caffe, Caffe2, Theano, Chainer ou Keras Conda no. AMIs de deep learning da AWS

Para obter números de versão específicos da estrutura, consulte [Notas de AMIs lançamento do Deep Learning](#)

Escolha esse tipo de DLAMI ou saiba mais sobre os DLAMIs diferentes com a opção Next Up.

Escolha uma das versões do CUDA e revise a lista completa das DLAMIs que têm essa versão no Apêndice, ou saiba mais sobre as diferentes DLAMIs com a opção Avançar.

A seguir

[AMI de deep learning base](#)

Related Topics

- Para obter instruções sobre como alternar entre as versões do CUDA, consulte o tutorial [Uso da AMI base de aprendizado profundo](#).

AMI de deep learning base

A AMI do deep learning base é como uma tela vazia para aprendizado profundo. Ela vem com tudo o que você precisa até o ponto da instalação de uma determinada estrutura e tem sua escolha de versões do CUDA.

Por que escolher a DLAMI base

Esse grupo de AMIs é útil para colaboradores de projeto que desejam usar um projeto de deep learning e criar o mais recente. É para alguém que deseja implementar seu próprio ambiente com a confiança de que o software NVIDIA mais recente está instalado e funcionando para que possa concentrar-se em escolher as estruturas e versões que deseja instalar.

Escolha esse tipo de DLAMI ou saiba mais sobre os DLAMIs diferentes com a opção Next Up.

A seguir

[DLAMI com o Conda](#)

Related Topics

- [Uso da AMI de deep learning base](#)

AMI de aprendizado profundo com Conda

O Conda DLAMI conda usa ambientes virtuais, eles estão presentes em várias estruturas ou em uma estrutura única. DLAMIs Esses ambientes são configurados para manter as instalações de estruturas diferentes separadas e simplificar a alternância entre estruturas. Isso é ótimo para aprendizado e testes com todas as estruturas que a DLAMI tem para oferecer. A maioria dos usuários acreditam que a nova AMI de deep learning com Conda é perfeita para eles.

Elas serão atualizadas frequentemente com as versões mais recentes das estruturas, e terão o software e os drivers de GPU mais recentes. Eles geralmente são chamados de “os” AMIs de deep learning da AWS na maioria dos documentos. Eles são DLAMIs compatíveis com os sistemas operacionais Ubuntu 20.04, Ubuntu 22.04, Amazon Linux 2 e Amazon Linux 2023. O suporte a sistemas operacionais depende do suporte do sistema operacional upstream.

Estabilidade e candidatos a lançamento

O Conda AMIs usa binários otimizados das versões formais mais recentes de cada estrutura. Versões "release candidate" e recursos experimentais não devem ser esperados. As otimizações dependem do suporte que a estrutura oferece para tecnologias de aceleração como a DNN MKL da Intel, que acelera o treinamento e a inferência em tipos de instância de CPU C4 e C5. Os binários também são compilados para suportar conjuntos de instruções avançados da Intel, incluindo, mas não se limitando a, AVX, AVX-2, SSE4 .1 e .2. SSE4 Essas operações de vetor de aceleração e de ponto de flutuação nas arquiteturas de Intel CPU. Além disso, para tipos de instância de GPU, o CUDA e o cuDNN são atualizados com qualquer versão compatível com o release oficial mais recente.

A AMI de aprendizado profundo com Conda instala automaticamente a versão mais otimizada da estrutura para sua EC2 instância Amazon na primeira ativação da estrutura. Para obter mais informações, consulte [Uso da AMI de aprendizado profundo com Conda](#).

Se você deseja realizar a instalação da origem usando opções de compilação personalizadas ou otimizadas, a de [AMI de deep learning bases](#) pode ser a melhor opção para você.

Defasagem do Python 2

A comunidade de código aberto Python encerrou oficialmente o suporte para Python 2 em 1 de janeiro de 2020. A PyTorch comunidade TensorFlow e anunciou que as versões TensorFlow 2.1 e PyTorch 1.4 são as últimas com suporte ao Python 2. As versões anteriores da DLAMI (v26, v25 etc.) que contêm ambientes Python 2 Conda continuarão disponíveis. No entanto, fornecemos atualizações para os ambientes Python 2 Conda em versões da DLAMI publicadas anteriormente somente se houver correções de segurança publicadas pela comunidade de código aberto para essas versões. As versões do DLAMI com as versões mais recentes dos frameworks PyTorch e não contêm TensorFlow os ambientes Python 2 Conda.

CUDA Support

Os números de versão específicos do CUDA podem ser encontrados nas [notas de versão da DLAMI da GPU](#).

A seguir

[Opções de arquitetura de DLAMI](#)

Related Topics

- Para ver um tutorial sobre como usar uma AMI de aprendizado profundo com Conda, consulte o tutorial do [Uso da AMI de aprendizado profundo com Conda](#).

Opções de arquitetura de DLAMI

As AMIs de deep learning da AWS são oferecidas com arquiteturas do [AWS Graviton2](#) baseadas em x86 ou Arm64.

Para obter informações sobre como começar a usar a ARM64 GPU DLAMI, consulte [O ARM64 DLAMI](#). Para obter mais detalhes sobre os tipos de instância disponíveis, consulte [Escolher o tipo de instância de DLAMI](#).

A seguir

[Opções do sistema operacional da DLAMI](#)

Opções do sistema operacional da DLAMI

DLAMIs são oferecidos nos seguintes sistemas operacionais.

- Amazon Linux 2
- Amazon Linux 2023
- Ubuntu 20.04
- Ubuntu 22.04

Versões mais antigas dos sistemas operacionais estão disponíveis como obsoletas DLAMIs. Para obter mais informações sobre a suspensão de uso da DLAMI, consulte [Descontinuação da DLAMI](#).

Antes de escolher uma DLAMI, avalie o tipo de instância que você precisa e identifique sua região da AWS .

A seguir

[Escolher o tipo de instância de DLAMI](#)

Escolher o tipo de instância de DLAMI

Em geral, considere o seguinte ao escolher um tipo de instância para uma DLAMI.

- Se você é novo no aprendizado profundo, uma instância com uma única GPU pode atender às suas necessidades.
- Caso se preocupe com o orçamento, é possível usar instâncias somente de CPU.
- Se você deseja otimizar o alto desempenho e a eficiência de custos para inferência de modelos de aprendizado profundo, pode usar instâncias com chips AWS Inferentia.
- Se você está procurando uma instância de GPU de alto desempenho com uma arquitetura de CPU baseada em Arm64, é possível usar o tipo de instância G5g.
- Se você estiver interessado em executar um modelo pré-treinado para inferência e previsões, você pode anexar um [Amazon Elastic Inference à sua instância da Amazon](#). EC2 O Amazon Elastic Inference fornece acesso a um acelerador com uma fração de uma GPU.
- Para serviços de inferência de alto volume, uma única instância de CPU com muita memória, ou um cluster dessas instâncias, pode ser uma solução melhor.
- Se estiver usando um modelo grande com muitos dados ou um tamanho de lote amplo, você precisará de uma instância maior com mais memória. Você também pode distribuir seu modelo para um cluster de GPUs. Usar uma instância com menos memória talvez seja a melhor solução se você diminuir o tamanho do lote. Isso pode afetar sua precisão e velocidade de treinamento.
- Se você estiver interessado em executar aplicativos de machine learning usando NVIDIA Collective Communications Library (NCCL) que exigem altos níveis de comunicação entre nós em escala, talvez você queira usar o [Elastic Fabric Adapter \(EFA\)](#).

Para obter mais detalhes sobre instâncias, consulte [de instância](#).

Os tópicos a seguir fornecem informações sobre as considerações relacionadas aos tipos de instância.

Important

O Deep Learning AMIs inclui drivers, software ou kits de ferramentas desenvolvidos, de propriedade ou fornecidos pela NVIDIA Corporation. Você concorda em usar esses drivers, software ou kits de ferramentas da NVIDIA somente em EC2 instâncias da Amazon que incluam hardware da NVIDIA.

Tópicos

- [Definição de preço para a DLAMI](#)
- [Disponibilidade de regiões da DLAMI](#)
- [Instâncias de GPU recomendadas](#)
- [Instâncias de CPU recomendadas](#)
- [Instâncias recomendadas do Inferentia](#)
- [Instâncias recomendadas do Trainium](#)

Definição de preço para a DLAMI

As estruturas de aprendizado profundo incluídas na DLAMI são gratuitas, e cada uma tem suas próprias licenças de código aberto. Embora o software incluído no DLAMI seja gratuito, você ainda precisa pagar pelo hardware subjacente da instância Amazon EC2 .

Alguns tipos de EC2 instância da Amazon são rotulados como gratuitos. É possível executar a DLAMI em uma dessas instâncias gratuitas. Isso significa que isso é totalmente gratuito quando você usa essa capacidade de instância. Se você precisar de uma instância mais poderosa com mais núcleos de CPU, mais espaço em disco, mais RAM ou uma ou mais GPUs, precisará de uma instância que não esteja na classe de instância de nível gratuito.

Para obter mais informações sobre opções e preços de instâncias, consulte [EC2 Preços da Amazon](#).

Disponibilidade de regiões da DLAMI

Cada região oferece suporte a uma variedade diferente de tipos de instância e, geralmente, um tipo de instância tem um custo ligeiramente diferente em regiões diferentes. DLAMIs não estão disponíveis em todas as regiões, mas é possível copiar DLAMIs para a região de sua escolha. Para obter mais informações, consulte [Copiar uma AMI](#). Observe a lista de seleções de regiões e escolha a que esteja próxima de você ou de seus clientes. Se você planeja usar mais de uma DLAMI e potencialmente criar um cluster, use a mesma região para todos os nós do cluster.

Para obter mais informações sobre regiões, acesse [Amazon EC2 Service Endpoints](#).

A seguir

[Instâncias de GPU recomendadas](#)

Instâncias de GPU recomendadas

Recomendamos uma instância de GPU para a maioria dos objetivos de aprendizado profundo. O treinamento de novos modelos é mais rápido em uma instância de GPU do que em uma instância de CPU. Você pode escalar sublinearmente quando tiver instâncias com várias GPUs ou se usar treinamento distribuído em várias instâncias com GPUs.

Os tipos de instância a seguir são compatíveis com a DLAMI. Para obter informações sobre as opções de tipo de instância de GPU e seus usos, consulte [instância](#) e selecione Computação acelerada.

Note

O tamanho do modelo deve ser um fator ao selecionar uma instância. Se o modelo exceder a RAM disponível de uma instância, escolha outro tipo de instância com memória suficiente para a aplicação.

- As [instâncias Amazon EC2 P6-B200](#) têm até 8 NVIDIA Blackwell B200. GPUs
- As [instâncias Amazon EC2 GB2 P6-00](#) têm até 4 NVIDIA Blackwell 00. GB2 GPUs
- [As instâncias Amazon EC2 P5e](#) têm até 8 NVIDIA Tesla H200. GPUs
- [As instâncias Amazon EC2 P5](#) têm até 8 NVIDIA Tesla H100. GPUs
- [As instâncias EC2 P4 da Amazon](#) têm até 8 NVIDIA Tesla A100. GPUs
- [As instâncias EC2 P3 da Amazon](#) têm até 8 NVIDIA Tesla V100. GPUs
- [As instâncias Amazon EC2 G3](#) têm até 4 NVIDIA Tesla M60. GPUs
- [As instâncias Amazon EC2 G4](#) têm até 4 NVIDIA T4. GPUs
- [As instâncias Amazon EC2 G5](#) têm até 8 NVIDIA A10G. GPUs
- [As instâncias Amazon EC2 G6](#) têm até 8 NVIDIA L4. GPUs
- [As instâncias Amazon EC2 G6e](#) têm até 8 NVIDIA L40S Tensor Core. GPUs
- [As instâncias Amazon EC2 G5g têm processadores Graviton2 baseados em ARM64 AWS](#) .

As instâncias da DLAMI oferecem ferramentas para monitorar e otimizar seus processos da GPU. Para obter mais informações sobre o monitoramento dos processos da GPU, consulte [Monitoramento e otimização de GPU](#).

Para ver tutoriais específicos sobre como trabalhar com instâncias G5g, consulte [O ARM64 DLAMI](#).

A seguir

[Instâncias de CPU recomendadas](#)

Instâncias de CPU recomendadas

Independentemente de você ter um orçamento, estar aprendendo sobre o deep learning ou apenas querer executar um serviço de previsão, você tem muitas opções acessíveis na categoria de CPU. Alguns frameworks usam a biblioteca MKL DNN da Intel, que acelera o treinamento e a inferência em tipos de instância de CPU C5 (não disponíveis em todas as regiões). Para obter informações sobre os tipos de instância de CPU, consulte [de instância](#) e selecione Otimizado para computação.

Note

O tamanho do modelo deve ser um fator ao selecionar uma instância. Se o modelo exceder a RAM disponível de uma instância, escolha outro tipo de instância com memória suficiente para a aplicação.

- [As instâncias Amazon EC2 C5](#) têm até 72 Intel v. CPUs As instâncias C5 se destacam em modelagem científica, processamento em lote, análise distribuída, computação de alto desempenho (HPC) e inferência de aprendizado profundo e de máquina.

A seguir

[Instâncias recomendadas do Inferentia](#)

Instâncias recomendadas do Inferentia

AWS As instâncias de inferência são projetadas para fornecer alto desempenho e economia para cargas de trabalho de inferência de modelos de aprendizado profundo. Especificamente, os tipos de instância Inf2 usam chips AWS Inferentia e o [SDK AWS Neuron](#), que é integrado a estruturas populares de aprendizado de máquina, como e. TensorFlow PyTorch

Os clientes podem usar instâncias Inf2 para executar aplicativos de inferência de machine learning em grande escala, como pesquisa, mecanismos de recomendação, visão computacional, reconhecimento de fala, processamento de linguagem natural, personalização e detecção de fraudes, com o menor custo na nuvem.

Note

O tamanho do modelo deve ser um fator ao selecionar uma instância. Se o modelo exceder a RAM disponível de uma instância, escolha outro tipo de instância com memória suficiente para a aplicação.

- [As instâncias Amazon EC2 Inf2](#) têm até 16 chips AWS Inferentia e 100 Gbps de taxa de transferência de rede.

Para obter mais informações sobre como começar a usar a AWS inferência DLAMIs, consulte [O chip de AWS inferência com DLAMI](#).

A seguir

[Instâncias recomendadas do Trainium](#)

Instâncias recomendadas do Trainium

AWS As instâncias Trainium são projetadas para fornecer alto desempenho e economia para cargas de trabalho de inferência de modelos de aprendizado profundo. Especificamente, os tipos de instância Trn1 usam chips AWS Trainium e o [SDK AWS Neuron](#), que é integrado a estruturas populares de aprendizado de máquina, como TensorFlow PyTorch

Os clientes podem usar instâncias Trn1 para executar aplicativos de inferência de machine learning em grande escala, como pesquisa, mecanismos de recomendação, visão computacional, reconhecimento de fala, processamento de linguagem natural, personalização e detecção de fraudes, com o menor custo na nuvem.

Note

O tamanho do modelo deve ser um fator ao selecionar uma instância. Se o modelo exceder a RAM disponível de uma instância, escolha outro tipo de instância com memória suficiente para a aplicação.

- [As instâncias Amazon EC2 Trn1](#) têm até 16 chips AWS Trainium e 100 Gbps de taxa de transferência de rede.

Configurar uma instância de DLAMI

Depois de [escolher uma DLAMI](#) e [escolher um tipo de instância da Amazon Elastic Compute Cloud \(EC2Amazon\)](#) que você deseja usar, você estará pronto para configurar sua nova instância de DLAMI.

Se você ainda não escolheu uma DLAMI EC2 e um tipo de instância, consulte. [Conceitos básicos da DLAMI](#)

Tópicos

- [Encontrar o ID de uma DLAMI](#)
- [Inicializar uma instância de DLAMI](#)
- [Conectar-se a uma instância de DLAMI](#)
- [Configurar um servidor de cadernos Jupyter em uma instância de DLAMI](#)
- [Limpar uma instância de DLAMI](#)

Encontrar o ID de uma DLAMI

Cada DLAMI tem um identificador exclusivo (ID). Ao iniciar uma instância de DLAMI usando o console da EC2 Amazon, você pode, opcionalmente, usar o ID da DLAMI para pesquisar a DLAMI que deseja usar. Quando você executa uma instância DLAMI usando AWS Command Line Interface o AWS CLI(), esse ID é obrigatório.

Você pode encontrar o ID do DLAMI de sua escolha usando AWS CLI um comando para EC2 Amazon ou Parameter Store, um recurso de. AWS Systems Manager Para obter instruções sobre como instalar e configurar o AWS CLI, consulte [Introdução ao AWS CLI](#) no Guia do AWS Command Line Interface usuário.

Using Parameter Store

Como encontrar o ID de uma DLAMI usando ssm get-parameter

No [ssm get-parameter](#) comando a seguir, para a --name opção, o formato do nome do parâmetro é `/aws/service/deeplearning/ami/$architecture/$ami_type/latest/ami-id`. Nesse formato de nome, *architecture* pode ser `x86_64` ou `arm64`. Especifique o *ami_type* usando o nome DLAMI e removendo as palavras-chave “deep”, “learning” e “ami”. O nome da AMI pode ser encontrado em [Notas de AMIs lançamento do Deep Learning](#).

⚠ Important

Para usar esse comando, o principal AWS Identity and Access Management (IAM) que você usa deve ter a `ssm:GetParameter` permissão. Para ter mais informações sobre as entidades principais do IAM, consulte a seção [Recursos adicionais](#) de Perfis do IAM no Guia do usuário do IAM.

- ```
aws ssm get-parameter --name /aws/service/deeplearning/ami/x86_64/base-oss-
nvidia-driver-ubuntu-22.04/latest/ami-id \\
--region us-east-1 --query "Parameter.Value" --output text
```

A saída deve ser semelhante à seguinte:

```
ami-09ee1a996ac214ce7
```

**💡 Tip**

Para alguns frameworks de DLAMI atualmente compatíveis, você pode encontrar exemplos de comandos `ssm get-parameter` mais específicos em [Notas de AMIs lançamento do Deep Learning](#). Escolha o link para as notas de lançamento da DLAMI escolhida, depois encontre o respectivo ID query nas notas de lançamento.

## Using Amazon EC2 CLI

Como encontrar o ID de uma DLAMI usando `ec2 describe-images`

No comando [ec2 describe-images](#) a seguir, para o valor do filtro `Name=name`, insira o nome da DLAMI. Você pode especificar uma versão de lançamento para determinado framework ou obter a versão mais recente substituindo o número da versão por um ponto de interrogação (?).

- ```
aws ec2 describe-images --region us-east-1 --owners amazon \\  
--filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu  
22.04) ????????' 'Name=state,Values=available' \\  
--query 'reverse(sort_by(Images, &CreationDate))[1].ImageId' --output text
```

A saída deve ser semelhante à seguinte:

```
ami-09ee1a996ac214ce7
```

i Tip

Para conferir um exemplo de comando `ec2 describe-images` específico para a DLAMI escolhida, consulte [Notas de AMIs lançamento do Deep Learning](#). Escolha o link para as notas de lançamento da DLAMI escolhida, depois encontre o respectivo ID query nas notas de lançamento.

Próxima etapa

Inicializar uma instância de DLAMI

Inicializar uma instância de DLAMI

Depois de [encontrar o ID](#) da DLAMI que você quer usar para inicializar uma instância de DLAMI, você estará com tudo pronto para inicializar a instância. Para iniciá-lo, você pode usar o EC2 console da Amazon ou o AWS Command Line Interface (AWS CLI).

i Note

Para esta demonstração, podemos fazer referências específicas à AMI de aprendizado profundo básica para GPU com driver OSS da NVIDIA (Ubuntu 22.04). Mesmo que selecione outra DLAMI, você deve ser capaz de seguir este guia.

EC2 console

i Note

Para acelerar as aplicações de machine learning e de computação de alta performance (HPC), você pode inicializar a instância de DLAMI com um Elastic Fabric Adapter (EFA). Para conferir instruções específicas, consulte [Executando uma AMIs de deep learning da AWS instância com o EFA](#).

1. Abra o [EC2 console](#).
2. Anote sua atual Região da AWS na navegação superior. Se essa não for a região desejada, altere essa opção antes de continuar. Para obter mais informações, consulte os [endpoints EC2 de serviço da Amazon](#) no Referência geral da Amazon Web Services.
3. Escolha Executar instância.
4. Insira um nome para a instância e selecione a DLAMI adequada.
 - a. Encontre um DLAMI existente em AMIs Meu ou escolha Início rápido.
 - b. Pesquise por ID da DLAMI. Navegue pelas opções e selecione a escolhida.
5. Escolha um tipo de instância. Você pode encontrar as famílias de instâncias recomendadas para uma DLAMI em [Notas de AMIs lançamento do Deep Learning](#). Para encontrar recomendações gerais sobre os tipos de instância de DLAMI, consulte [Escolher o tipo de instância de DLAMI](#).
6. Escolha Executar instância.

AWS CLI

- Para usar o AWS CLI, você deve ter o ID do DLAMI que deseja usar, Região da AWS o tipo de instância EC2 e as informações do token de segurança. Em seguida, você pode iniciar a instância usando o [ec2 run-instances](#) AWS CLI comando.

Para obter instruções sobre como instalar e configurar o AWS CLI, consulte [Introdução ao AWS CLI](#) no Guia do AWS Command Line Interface usuário. Para obter mais informações, incluindo exemplos de comandos, consulte [Iniciar, listar e fechar EC2 instâncias da Amazon para AWS CLI](#) o.

Depois de iniciar sua instância usando o EC2 console da Amazon ou AWS CLI, aguarde até que a instância esteja pronta. Isso normalmente leva apenas alguns minutos. Você pode verificar o status da instância no [EC2 console da Amazon](#). Para obter mais informações, consulte [Verificações de status para EC2 instâncias da Amazon](#) no Guia EC2 do usuário da Amazon.

Próxima etapa

[Conectar-se a uma instância de DLAMI](#)

Conectar-se a uma instância de DLAMI

Depois que você [inicializar uma instância de DLAMI](#) e ela estiver em execução, poderá se conectar a ela de um cliente (Windows, macOS ou Linux) usando SSH. Para obter instruções, consulte [Conecte-se à sua instância Linux usando SSH](#) no Guia do EC2 usuário da Amazon.

Mantenha uma cópia do comando de login SSH à mão caso queira configurar um servidor de cadernos Jupyter depois de fazer login. Para se conectar à página da web do Jupyter, use uma variação desse comando.

Próxima etapa

[Configurar um servidor de cadernos Jupyter em uma instância de DLAMI](#)

Configurar um servidor de cadernos Jupyter em uma instância de DLAMI

Com um servidor de cadernos Jupyter, você pode criar e executar cadernos Jupyter por meio da instância de DLAMI. Com os notebooks Jupyter, você pode realizar experimentos de aprendizado de máquina (ML) para treinamento e inferência enquanto usa a AWS infraestrutura e acessa pacotes incorporados ao DLAMI. Para ter mais informações sobre cadernos Jupyter, consulte [The Jupyter Notebook](#) no site da documentação do usuário do Jupyter.

Para configurar um servidor de cadernos Jupyter, você deve:

- Configurar o servidor de cadernos Jupyter na instância de DLAMI.
- Configurar o cliente para se conectar ao servidor de cadernos Jupyter. Fornecemos instruções de configuração para Windows, macOS e clientes Linux.
- Testar a configuração fazendo login no servidor de cadernos Jupyter.

Para concluir essas etapas, siga as instruções nos tópicos a seguir. Depois de configurar um servidor Jupyter Notebook, você pode executar os exemplos de tutoriais de notebook fornecidos no DLAMIs. Para obter mais informações, consulte [Executar os tutoriais do notebook Jupyter](#).

Tópicos

- [Proteger o servidor de cadernos Jupyter em uma instância de DLAMI](#)

- [Iniciar o servidor de cadernos Jupyter em uma instância de DLAMI](#)
- [Conectar um cliente ao servidor de cadernos Jupyter em uma instância de DLAMI](#)
- [Fazer login no servidor de cadernos Jupyter em uma instância de DLAMI](#)

Proteger o servidor de cadernos Jupyter em uma instância de DLAMI

Para manter um servidor de cadernos Jupyter seguro, recomendamos configurar uma senha e criar um certificado SSL para o servidor. Para configurar uma senha e SSL, primeiro [conecte-se à instância de DLAMI](#) e siga estas instruções.

Como proteger o servidor de cadernos Jupyter

1. O Jupyter fornece um utilitário de senha. Execute o comando a seguir e insira sua senha preferencial no prompt.

```
$ jupyter notebook password
```

O resultado será semelhante ao seguinte:

```
Enter password:
Verify password:
[NotebookPasswordApp] Wrote hashed password to /home/ubuntu/.jupyter/
jupyter_notebook_config.json
```

2. Crie um certificado SSL autoassinado. Siga os prompts para preencher sua localidade, conforme julgar necessário. É necessário inserir . se deseja deixar um prompt em branco. Suas respostas não afetarão a funcionalidade do certificado.

```
$ cd ~
$ mkdir ssl
$ cd ssl
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout mykey.key -out
mycert.pem
```

Note

Talvez você queira criar um certificado SSL regular assinado por terceiros que não faz com que o navegador forneça um aviso de segurança. Esse processo é muito mais complexo.

Para ter mais informações, consulte [Securing a notebook server](#) na documentação do usuário do caderno Jupyter.

Próxima etapa

[Iniciar o servidor de cadernos Jupyter em uma instância de DLAMI](#)

Iniciar o servidor de cadernos Jupyter em uma instância de DLAMI

Depois de [proteger o servidor de cadernos Jupyter com uma senha e SSL](#), você pode iniciar o servidor. Faça login na instância de DLAMI e execute o comando a seguir que usa o certificado SSL que você criou anteriormente.

```
$ jupyter notebook --certfile=~/.ssl/mycert.pem --keyfile ~/.ssl/mykey.key
```

Com o servidor iniciado, agora você pode se conectar a ele por meio de um túnel SSH em seu computador cliente. Quando o servidor for executado, você verá alguma saída do Jupyter confirmando que o servidor está em execução. Nesse ponto, ignore o texto informando que você pode acessar o servidor por meio de um URL de host local, pois isso não funcionará até criar o túnel.

Note

O Jupyter resolverá a troca de ambientes para você ao alternar as estruturas usando a interface da web do Jupyter. Para obter mais informações, consulte [Alternar ambientes com o Jupyter](#).

Próxima etapa

[Conectar um cliente ao servidor de cadernos Jupyter em uma instância de DLAMI](#)

Conectar um cliente ao servidor de cadernos Jupyter em uma instância de DLAMI

Depois de [iniciar o servidor de cadernos Jupyter na instância de DLAMI](#), configure um cliente Windows, macOS ou Linux para se conectar ao servidor. Ao se conectar, você pode criar e acessar

cadernos Jupyter no servidor em sua área de trabalho e executar o código de aprendizado profundo no servidor.

Pré-requisitos

Você deve ter as seguintes informações, que são necessárias para configurar um túnel SSH:

- O nome DNS público da sua EC2 instância da Amazon. Para obter mais informações, consulte os [tipos de nome de host de EC2 instância da Amazon](#) no Guia do EC2 usuário da Amazon.
- O par de chaves do arquivo de chave privada. Para obter mais informações sobre como acessar seu par de chaves, consulte os [pares de EC2 chaves da Amazon e as EC2 instâncias](#) da Amazon no Guia EC2 do usuário da Amazon.

Conectar-se de um cliente Windows, macOS ou Linux

Para se conectar à instância de DLAMI por meio de um cliente Windows, macOS ou Linux, siga as instruções do sistema operacional do seu cliente.

Windows

Como se conectar à instância de DLAMI por meio de um cliente Windows usando SSH

1. Use um cliente SSH para Windows, como PuTTY. Para obter instruções, consulte [Conecte-se à sua instância Linux usando PuTTY no Guia EC2](#) do usuário da Amazon. Para outras opções de conexão SSH, consulte [Conectar-se à instância do Linux usando SSH](#).
2. (Opcional) Crie um túnel SSH para um servidor do Jupyter em execução. Instale o Git Bash no cliente Windows e siga as instruções de conexão para clientes macOS e Linux.

macOS or Linux

Como se conectar à instância de DLAMI por meio de um cliente macOS ou Linux usando SSH

1. Abra um terminal.
2. Execute o comando a seguir para encaminhar todas as solicitações na porta local 8888 para a porta 8888 em sua instância remota da Amazon EC2 . Atualize o comando substituindo a localização da sua chave para acessar a EC2 instância da Amazon e o nome DNS público da sua EC2 instância da Amazon. Observe que, para uma AMI do Amazon Linux, o nome do usuário `ec2-user` em vez de `ubuntu`.

```
$ ssh -i ~/mykeypair.pem -N -f -L 8888:localhost:8888 ubuntu@ec2-###-##-###-###.compute-1.amazonaws.com
```

Esse comando abre um túnel entre seu cliente e a EC2 instância remota da Amazon que está executando o servidor Jupyter Notebook.

Próxima etapa

[Fazer login no servidor de cadernos Jupyter em uma instância de DLAMI](#)

Fazer login no servidor de cadernos Jupyter em uma instância de DLAMI

Depois de [conectar o cliente ao servidor de cadernos Jupyter na instância de DLAMI](#), você pode fazer login no servidor.

Como fazer login no servidor pelo navegador

1. Na barra de endereço do navegador, insira o seguinte URL ou clique neste link: <https://localhost:8888>
2. Com um certificado SSL autoassinado, o navegador avisará você para que não continue com o acesso ao site.



Your connection is not private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

☐ Help improve Safe Browsing by sending some [system information and page content](#) to Google.
[Privacy policy](#)



Back to safety

Como você mesmo configurou isso, é seguro continuar. O navegador receberá um botão "avançado", "mostrar detalhes" ou algo semelhante.



Your connection is not private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

☐ Help improve Safe Browsing by sending some [system information and page content](#) to Google.
[Privacy policy](#)

Hide advanced

Back to safety

This server could not prove that it is **localhost**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to localhost \(unsafe\)](#)

Clique nele e depois clique no link "prosseguir para localhost". Se a conexão for bem-sucedida, você verá a página da web do servidor de cadernos Jupyter. Nesse momento, será solicitada a senha definida anteriormente.

Agora você tem acesso ao servidor de cadernos Jupyter em execução na instância de DLAMI. Você pode criar novos cadernos ou executar os [Tutoriais](#) fornecidos.

Limpar uma instância de DLAMI

Quando você não precisar mais da sua instância DLAMI, poderá interrompê-la ou encerrá-la na EC2 Amazon para evitar cobranças inesperadas.

Se você interromper uma instância, poderá mantê-la ativa e iniciá-la mais tarde, quando quiser usá-la novamente. Suas configurações, arquivos e outras informações não voláteis são armazenados em um volume no Amazon Simple Storage Service (Amazon S3). Enquanto a instância está parada, você incorre em cobranças do S3 pela retenção do volume, mas não incorre em cobranças por recursos computacionais. Quando você iniciar a instância novamente, ela montará esse volume de armazenamento com os dados.

Se você encerrar uma instância, ela será perdida e você não poderá iniciá-la novamente.

Obviamente, uma instância encerrada não incorrerá em mais nenhuma cobrança pelos recursos de computação. No entanto, seus dados continuarão residindo no Amazon S3 e você poderá continuar incorrendo em cobranças do S3. Para evitar todas as cobranças adicionais relacionadas à instância encerrada, você também deve excluir o volume de armazenamento no Amazon S3. Para obter instruções, consulte [Encerrar EC2 instâncias da Amazon](#) no Guia do EC2 usuário da Amazon.

Para obter mais informações sobre os estados das EC2 instâncias da Amazon, como stopped e terminated, consulte [as mudanças no estado da EC2 instância](#) da Amazon no Guia EC2 do usuário da Amazon.

Utilizar uma DLAMI

Tópicos

- [Uso da AMI de aprendizado profundo com Conda](#)
- [Uso da AMI base de aprendizado profundo](#)
- [Executar os tutoriais do notebook Jupyter](#)
- [Tutoriais](#)

As seções a seguir descrevem como a AMI de deep learning com Conda pode ser usada para alternar ambientes, executar o código de exemplo para cada uma das estruturas e executar o Jupyter para que você possa testar diferentes tutoriais de notebook.

Uso da AMI de aprendizado profundo com Conda

Tópicos

- [Introdução à AMI de aprendizado profundo com Conda](#)
- [Faça login na DLAMI](#)
- [Inicie o TensorFlow meio ambiente](#)
- [Mude para o PyTorch ambiente Python 3](#)
- [Remoção de ambientes](#)

Introdução à AMI de aprendizado profundo com Conda

O Conda é um sistema de gerenciamento de pacotes e de ambientes de código aberto que é executado no Windows, macOS e Linux. O Conda instala, executa e atualiza pacotes e suas dependências rapidamente. O Conda cria, salva, carrega e alterna facilmente entre ambientes em seu computador local.

A AMI de deep learning com Conda foi configurada para que você alterne facilmente entre ambientes de aprendizado profundo. As instruções a seguir orientam você na execução de alguns comandos básicos com conda. Elas também ajudam você a verificar se a importação básica da estrutura está funcionando e se você pode executar algumas operações simples com a estrutura. Em seguida, você pode continuar com tutoriais mais completos fornecidos com a DLAMI ou os exemplos de estruturas encontrados em cada site de projeto das estruturas.

Faça login na DLAMI

Após fazer login em seu servidor, você verá uma "mensagem do dia" (MOTD) do servidor que descreve vários comandos do Conda que você pode usar para alternar entre as diferentes estruturas de aprendizado profundo. Abaixo está um exemplo de MOTD. Seu MOTD específico pode variar conforme novas versões da DLAMI são lançadas.

```
=====
AMI Name: Deep Learning OSS Nvidia Driver AMI (Amazon Linux 2) Version 77
Supported EC2 instances: G4dn, G5, G6, Gr6, P4d, P4de, P5
  * To activate pre-built tensorflow environment, run: 'source activate
tensorflow2_p310'
  * To activate pre-built pytorch environment, run: 'source activate
pytorch_p310'
  * To activate pre-built python3 environment, run: 'source activate python3'

NVIDIA driver version: 535.161.08

CUDA versions available: cuda-11.7 cuda-11.8 cuda-12.0 cuda-12.1 cuda-12.2

Default CUDA version is 12.1

Release notes: https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-
release-notes.html
AWS Deep Learning AMI Homepage: https://aws.amazon.com/machine-learning/amis/
Developer Guide and Release Notes: https://docs.aws.amazon.com/dlami/latest/
devguide/what-is-dlami.html
Support: https://forums.aws.amazon.com/forum.jspa?forumID=263
For a fully managed experience, check out Amazon SageMaker at https://
aws.amazon.com/sagemaker
=====
```

Inicie o TensorFlow meio ambiente

Note

Quando iniciar seu primeiro ambiente Conda, aguarde enquanto ele carrega. A AMI de aprendizado profundo com Conda instala automaticamente a versão mais otimizada da estrutura para sua EC2 instância na primeira ativação da estrutura. Não deve haver atrasos subsequentes.

1. Ative o ambiente TensorFlow virtual para Python 3.

```
$ source activate tensorflow2_p310
```

2. Inicie o terminal iPython.

```
(tensorflow2_p310)$ ipython
```

3. Execute um TensorFlow programa rápido.

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Você deve ver "Hello, Tensorflow!"

A seguir

[Executar os tutoriais do notebook Jupyter](#)

Mude para o PyTorch ambiente Python 3

Se você ainda estiver no console do iPython, use `quit()` e, em seguida, prepare-se para alternar os ambientes.

- Ative o ambiente PyTorch virtual para Python 3.

```
$ source activate pytorch_p310
```

Teste alguns PyTorch códigos

Para testar sua instalação, use o Python para escrever um PyTorch código que cria e imprime uma matriz.

1. Inicie o terminal iPython.

```
(pytorch_p310)$ ipython
```

2. Importar PyTorch.

```
import torch
```

Você pode ver uma mensagem de aviso sobre um pacote de terceiros. Você pode ignorá-lo.

3. Crie uma matriz 5x3 com elementos inicializados de modo aleatório. Imprima a matriz.

```
x = torch.rand(5, 3)
print(x)
```

Verifique o resultado.

```
tensor([[0.3105, 0.5983, 0.5410],
        [0.0234, 0.0934, 0.0371],
        [0.9740, 0.1439, 0.3107],
        [0.6461, 0.9035, 0.5715],
        [0.4401, 0.7990, 0.8913]])
```

Remoção de ambientes

Se ficar sem espaço na DLAMI, você poderá optar por desinstalar pacotes do Conda que não estiver usando:

```
conda env list
conda env remove --name <env_name>
```

Uso da AMI base de aprendizado profundo

Uso da AMI base de aprendizado profundo

A AMI Base é fornecida com uma plataforma base de drivers de GPU e bibliotecas de aceleração para implantar seu próprio ambiente personalizado de aprendizado profundo. Por padrão, a AMI é configurada com qualquer versão do ambiente do NVIDIA CUDA. Você também pode alternar entre diferentes versões do CUDA. Consulte as seguintes instruções para saber como fazer isso.

Configurar as versões do CUDA

É possível verificar a versão do CUDA ao executar o programa `nvcc` da NVIDIA.

```
nvcc --version
```

Você pode selecionar e verificar uma determinada versão do CUDA com o comando bash a seguir:

```
sudo rm /usr/local/cuda
sudo ln -s /usr/local/cuda-12.0 /usr/local/cuda
```

Para obter mais informações, consulte as [notas de versão da DLAMI base](#).

Executar os tutoriais do notebook Jupyter

Tutoriais e exemplos são fornecidos com cada origem dos projetos de aprendizado profundo e, na maioria dos casos, eles são executados em qualquer DLAMI. Se você escolher a [AMI de aprendizado profundo com Conda](#), obterá o benefício adicional de alguns tutoriais escolhidos a dedo já configurados e prontos para serem testados.

Important

Para executar os tutoriais instalados no caderno Jupyter na DLAMI, você precisará [Configurar um servidor de cadernos Jupyter em uma instância de DLAMI](#).

Assim que o servidor do Jupyter estiver em execução, você poderá executar os tutoriais por meio de seu navegador da web. Se estiver executando a AMI de deep learning com Conda ou tiver configurado ambientes do Python, será possível alternar os kernels do Python na interface do caderno Jupyter. Selecione o kernel apropriado antes de tentar executar um tutorial específico à estrutura. Exemplos adicionais são fornecidos para os usuários da AMI de deep learning com Conda.

Note

Muitos tutoriais exigem módulos adicionais do Python que podem não estar configurados na DLAMI. Se você receber um erro, como "xyz module not found", faça login na DLAMI, ative o ambiente conforme descrito acima e, em seguida, instale os módulos necessários.

 Tip

Os tutoriais e exemplos de aprendizado profundo geralmente dependem de um ou mais GPUs. Se seu tipo de instância não tiver uma GPU, talvez seja necessário alterar um pouco o código de exemplo para que ele seja executado.

Navegar pelos tutoriais instalados

Depois de fazer login no servidor do Jupyter e poder ver o diretório dos tutoriais (somente na AMI de deep learning com Conda), serão apresentadas pastas de tutoriais para cada nome de estrutura. Se você não vir uma estrutura listada, os tutoriais não estarão disponíveis para essa estrutura em sua DLAMI atual. Clique no nome da estrutura para ver os tutoriais listados e, em seguida, clique em um tutorial para iniciá-lo.

Na primeira vez que executa um caderno na AMI de deep learning com Conda, você precisará informar qual ambiente deseja usar. Você poderá selecionar de uma lista. Cada ambiente é denominado de acordo com este padrão:

Environment (conda_framework_python-version)

Por exemplo, você pode ver Environment (conda_mxnet_p36), o que significa que o ambiente tem um MXNet Python 3. A outra variação disso seria Environment (conda_mxnet_p27), o que significa que o ambiente tem um MXNet Python 2.

 Tip

Se você está preocupado sobre qual versão do CUDA está ativa, uma maneira de ver isso é no MOTD, quando você faz login pela primeira vez na DLAMI.

Alternar ambientes com o Jupyter

Se você decidir experimentar um tutorial para uma estrutura diferente, certifique-se de verificar o kernel em execução. Essas informações podem ser vistas no canto superior direito da interface do Jupyter, logo abaixo do botão de logout. Você pode alterar o kernel em qualquer notebook clicando no item de menu Kernel do Jupyter, em seguida, em Change Kernel e, em seguida, clicando no ambiente que atende ao notebook que você está executando.

Neste ponto, você precisará executar novamente todas as células porque uma alteração no kernel apagará o estado de qualquer coisa que tenha executado anteriormente.

Tip

Alternar entre as estruturas pode ser divertido e instrutivo, no entanto, você pode esgotar a memória. Se você começar a receber erros, examine a janela de terminal que tem o servidor do Jupyter em execução. Há mensagens úteis e registros de erros aqui, e você pode ver um out-of-memory erro. Para corrigir isso, você pode ir para a página inicial do servidor do Jupyter, clicar na guia Running e, em seguida, em Shutdown para cada um dos tutoriais que provavelmente ainda estão em execução em segundo plano e consumindo toda a sua memória.

Tutoriais

Veja a seguir os tutoriais sobre como usar o software da AMI de deep learning com Conda.

Tópicos

- [Ativar estruturas](#)
- [Treinamento distribuído usando o Elastic Fabric Adapter](#)
- [Monitoramento e otimização de GPU](#)
- [O chip de AWS inferência com DLAMI](#)
- [O ARM64 DLAMI](#)
- [Inferência](#)
- [Fornecimento de modelos](#)

Ativar estruturas

Veja a seguir as estruturas de deep learning instaladas na AMI de deep learning com Conda. Clique em uma estrutura para saber como ativá-la.

Tópicos

- [PyTorch](#)
- [TensorFlow 2](#)

PyTorch

Ativando PyTorch

Quando um pacote Conda estável de uma estrutura é lançada, ele é testado e pré-instalado na DLAMI. Se desejar executar as mais recentes compilações noturnas não testadas, você pode [PyTorchInstall's Nightly Build \(experimental\)](#) manualmente.

Para ativar a estrutura instalada no momento, siga estas instruções sobre a AMI de deep learning com Conda.

Para o PyTorch Python 3 com CUDA e MKL-DNN, execute este comando:

```
$ source activate pytorch_p310
```

Inicie o terminal iPython.

```
(pytorch_p310)$ ipython
```

Execute um PyTorch programa rápido.

```
import torch
x = torch.rand(5, 3)
print(x)
print(x.size())
y = torch.rand(5, 3)
print(torch.add(x, y))
```

Você deve ver a matriz aleatória inicial impressa, seu tamanho e a adição de outra matriz aleatória.

PyTorchInstall's Nightly Build (experimental)

Como instalar a PyTorch partir de uma compilação noturna

Você pode instalar a PyTorch versão mais recente em um ou em ambos os ambientes PyTorch Conda em sua AMI de aprendizado profundo com o Conda.

- (Opção para Python 3) - Ative o ambiente Python 3: PyTorch

```
$ source activate pytorch_p310
```


2. As etapas restantes assumem que você está usando o ambiente do `pytorch_p310`. Remova o atualmente instalado PyTorch:

```
(pytorch_p310)$ pip uninstall torch
```

3.
 - (Opção para instâncias de GPU) - Instale a versão noturna mais recente do com CUDA.0: PyTorch

```
(pytorch_p310)$ pip install torch_nightly -f https://download.pytorch.org/whl/nightly/cu100/torch_nightly.html
```

- (Opção para instâncias de CPU) - Instale a versão noturna mais recente de PyTorch for instâncias sem: GPUs

```
(pytorch_p310)$ pip install torch_nightly -f https://download.pytorch.org/whl/nightly/cpu/torch_nightly.html
```

4. Para verificar se você instalou com sucesso a última compilação noturna, inicie o IPython terminal e verifique a versão do. PyTorch

```
(pytorch_p310)$ ipython
```

```
import torch
print (torch.__version__)
```

A saída deve imprimir algo semelhante a `1.0.0.dev20180922`

5. Para verificar se a compilação PyTorch noturna funciona bem com o exemplo do MNIST, você pode executar um script de teste no repositório de exemplos PyTorch da:

```
(pytorch_p310)$ cd ~
(pytorch_p310)$ git clone https://github.com/pytorch/examples.git pytorch_examples
(pytorch_p310)$ cd pytorch_examples/mnist
(pytorch_p310)$ python main.py || exit 1
```

Mais tutoriais

Para mais tutoriais e exemplos, consulte os documentos oficiais, a [PyTorch documentação](#) e o site da estrutura. [PyTorch](#)

TensorFlow 2

Este tutorial mostra como ativar TensorFlow 2 em uma instância executando a AMI de aprendizado profundo com o Conda (DLAMI no Conda) e executar um programa 2. TensorFlow

Quando um pacote Conda estável de uma estrutura é lançada, ele é testado e pré-instalado na DLAMI.

Ativando 2 TensorFlow

Para rodar TensorFlow no DLAMI com Conda

1. Para ativar TensorFlow 2, abra uma instância do Amazon Elastic Compute Cloud (Amazon EC2) do DLAMI com o Conda.
2. Para TensorFlow 2 e Keras 2 em Python 3 com CUDA 10.1 e MKL-DNN, execute este comando:

```
$ source activate tensorflow2_p310
```

3. Inicie o terminal iPython:

```
(tensorflow2_p310)$ ipython
```

4. Execute um programa TensorFlow 2 para verificar se ele está funcionando corretamente:

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
tf.print(hello)
```

Hello, TensorFlow! deve aparecer na tela.

Mais tutoriais

Para ver mais tutoriais e exemplos, consulte a TensorFlow documentação da [API TensorFlow Python](#) ou acesse o site. [TensorFlow](#)

Treinamento distribuído usando o Elastic Fabric Adapter

O [Elastic Fabric Adapter \(EFA\)](#) é um dispositivo de rede que é possível anexar à instância da DLAMI para acelerar as aplicações de Computação de Alta Performance (HPC). O EFA permite

que você alcance o desempenho do aplicativo de um cluster de HPC local, com a escalabilidade, a flexibilidade e a elasticidade fornecidas pela nuvem. AWS

Os tópicos a seguir mostram como começar a usar o EFA com a DLAMI.

 Note

Escolha sua DLAMI nesta [lista de DLAMI de GPU base](#)

Tópicos

- [Executando uma AMIs de deep learning da AWS instância com o EFA](#)
- [Uso do EFA na DLAMI](#)

Executando uma AMIs de deep learning da AWS instância com o EFA

A DLAMI base mais recente está pronta para usar o EFA e vem com os drivers, os módulos do kernel, libfabric, openmpi e o [plug-in OFI NCCL](#) necessários para instâncias de GPU.

Você pode encontrar as versões do CUDA compatíveis de uma DLAMI base nas [notas de versão](#).

Nota:

- Ao executar um aplicativo NCCL usando `mpirun` no EFA, será necessário especificar o caminho completo para as instalações do EFA com suporte como:

```
/opt/amazon/openmpi/bin/mpirun <command>
```

- Para habilitar seu aplicativo para usar o EFA, adicione `FI_PROVIDER="efa"` ao comando `mpirun` como mostrado em [Uso do EFA na DLAMI](#).

Tópicos

- [Preparar um grupo de segurança habilitado para o EFA](#)
- [Executar sua instância](#)
- [Verificar anexo do EFA](#)

Preparar um grupo de segurança habilitado para o EFA

O EFA requer um grupo de segurança que permita todo o tráfego recebido do grupo de segurança e enviado para ele. Para ter mais informações, consulte a [Documentação do EFA](#).

1. Abra o EC2 console da Amazon em <https://console.aws.amazon.com/ec2/>.
2. No painel de navegação, escolha Security Groups (Grupos de segurança) e, em seguida, Create Security Group (Criar grupo de segurança).
3. Na janela Security group, faça o seguinte:
 - Em Security group name (Nome do grupo de segurança), insira um nome descritivo para o grupo de segurança, como EFA-enabled security group.
 - (Opcional) Em Description (Descrição), insira uma breve descrição do grupo de segurança.
 - Em VPC, selecione a VPC na qual você pretende executar suas instâncias habilitadas para EFA.
 - Escolha Criar.
4. Selecione o grupo de segurança que você criou e, na guia Description (Descrição), copie o Group ID (ID do grupo).
5. Nas guias Entrada e Saída, faça o seguinte:
 - Selecione Edit.
 - Para Type (Tipo), escolha All traffic (Todo o tráfego).
 - Em Source, escolha Custom.
 - Cole o ID do grupo de segurança que você copiou no campo.
 - Escolha Salvar.
6. Habilite o tráfego de entrada fazendo referência a [Autorizar tráfego de entrada para as instâncias do Linux](#). Se ignorar esta etapa, você não poderá se comunicar com sua instância da DLAMI.

Executar sua instância

Atualmente, o EFA no AMIs de deep learning da AWS é compatível com os seguintes tipos de instância e sistemas operacionais:

- P3dn: Amazon Linux 2, Ubuntu 20.04
- P4d, P4de: Amazon Linux 2, Amazon Linux 2023, Ubuntu 20.04, Ubuntu 22.04

- P5, P5e, P5en: Amazon Linux 2, Amazon Linux 2023, Ubuntu 20.04, Ubuntu 22.04

A seção a seguir mostra como iniciar uma instância da DLAMI habilitada para EFA. Para obter mais informações sobre executar uma instância habilitada para EFA, consulte [Executar instâncias habilitadas para EFA em um grupo com posicionamento em cluster](#).

1. Abra o EC2 console da Amazon em <https://console.aws.amazon.com/ec2/>.
2. Escolha Executar instância.
3. Na página Escolha uma AMI, selecione uma DLAMI compatível encontrada na [página de notas de lançamento de DLAMIs](#).
4. Na página Escolher um tipo de instância, selecione um dos seguintes tipos de instância com suporte e escolha Próximo: Configurar os detalhes da instância. Consulte este link para acessar a lista de instâncias compatíveis: [Conceitos básicos do EFA e MPI](#).
5. Na página Configurar detalhes da instância, faça o seguinte:
 - Em Number of instances (Número de instâncias), insira o número de instâncias habilitadas para EFA que você deseja executar.
 - Em Rede e Sub-rede, selecione a VPC e a sub-rede na qual executar as instâncias.
 - [Opcional] Em Grupo de posicionamento, selecione Adicionar instância ao grupo de posicionamento. Para obter o melhor desempenho, execute as instâncias dentro de um placement group.
 - [Opcional] Em Nome do grupo de posicionamento, selecione Adicionar a um novo grupo de posicionamento, insira um nome descritivo para o grupo de posicionamento e, em Estratégia de grupo de posicionamento, selecione cluster.
 - Habilite o “Elastic Fabric Adapter” nesta página. Se esta opção estiver desabilitada, altere a sub-rede para uma que ofereça suporte ao tipo de instância selecionado.
 - Na seção Interfaces de rede, para dispositivo eth0, escolha Nova interface de rede. Opcionalmente, você pode especificar um IPv4 endereço principal e um ou mais IPv4 endereços secundários. Se você estiver iniciando a instância em uma sub-rede que tenha um bloco IPv6 CIDR associado, você pode especificar opcionalmente um IPv6 endereço primário e um ou mais endereços secundários. IPv6
 - Escolha Next: Add Storage.
6. Na página Add Storage (Adicionar armazenamento), especifique os volumes para anexar às instâncias em associação aos volumes especificados pela AMI (como o volume do dispositivo raiz) e escolha Next: Add Tags (Próximo: adicionar tags).

7. Na página Adicionar tags, especifique tags para as instâncias, como nome amigável, e selecione Próximo: Configurar grupo de segurança.
8. Na página Configurar grupo de segurança, em Atribuir um grupo de segurança, escolha Selecionar um grupo de segurança existente e selecione o grupo de segurança que você criou anteriormente.
9. Escolha revisar e iniciar.
10. Na página Revisar execução da instância, reveja as configurações, e escolha Executar para escolher um par de chaves e executar a instâncias.

Verificar anexo do EFA

No console

Depois de iniciar a instância, verifique os detalhes da instância no AWS console. Para fazer isso, selecione a instância no EC2 console e veja a guia Descrição no painel inferior da página. Encontre o parâmetro "Network Interfaces: eth0" e clique em eth0 e um pop-up será aberto. O "Elastic Fabric Adapter" deve estar habilitado.

Se o EFA não estiver habilitado, você poderá corrigir isso da seguinte forma:

- Encerrar a EC2 instância e iniciar uma nova com as mesmas etapas. Verifique se o EFA está associado.
- Associe o EFA a uma instância existente.
 1. No EC2 console, vá para Interfaces de rede.
 2. Clique em "Criar uma interface de rede".
 3. Selecione a mesma sub-rede na qual está sua instância.
 4. Habilite o "Elastic Fabric Adapter" e clique em Criar.
 5. Volte para a guia EC2 Instâncias e selecione sua instância.
 6. Acesse Ações: estado da instância e interrompa a instância antes de anexar o EFA.
 7. Em "Ações", selecione "Rede: Anexar Interface de Rede".
 8. Selecione a interface que você acabou de criar e clique em associar.
 9. Reinicie sua instância.

Na instância

O script de teste a seguir já está presente na DLAMI. Execute-o para garantir que os módulos do kernel sejam carregados corretamente.

```
$ fi_info -p efa
```

O resultado deve ser semelhante ao seguinte:

```
provider: efa
  fabric: EFA-fe80::e5:56ff:fe34:56a8
  domain: efa_0-rdm
  version: 2.0
  type: FI_EP_RDM
  protocol: FI_PROTO_EFA
provider: efa
  fabric: EFA-fe80::e5:56ff:fe34:56a8
  domain: efa_0-dgrm
  version: 2.0
  type: FI_EP_DGRAM
  protocol: FI_PROTO_EFA
provider: efa;ofi_rxd
  fabric: EFA-fe80::e5:56ff:fe34:56a8
  domain: efa_0-dgrm
  version: 1.0
  type: FI_EP_RDM
  protocol: FI_PROTO_RXD
```

Verificar a configuração do grupo de segurança

O script de teste a seguir já está presente na DLAMI. Execute-o para garantir que o grupo de segurança criado esteja configurado corretamente.

```
$ cd /opt/amazon/efa/test/
$ ./efa_test.sh
```

O resultado deve ser semelhante ao seguinte:

```
Starting server...
Starting client...
```

bytes	#sent	#ack	total	time	MB/sec	usec/xfer	Mxfers/sec
64	10	=10	1.2k	0.02s	0.06	1123.55	0.00
256	10	=10	5k	0.00s	17.66	14.50	0.07
1k	10	=10	20k	0.00s	67.81	15.10	0.07
4k	10	=10	80k	0.00s	237.45	17.25	0.06
64k	10	=10	1.2m	0.00s	921.10	71.15	0.01
1m	10	=10	20m	0.01s	2122.41	494.05	0.00

Se ele parar de responder ou não for concluído, certifique-se de que seu grupo de segurança tenha as inbound/outbound regras corretas.

Uso do EFA na DLAMI

A seção a seguir descreve como usar o EFA para executar aplicativos de vários nós na AMIs de deep learning da AWS.

Como executar aplicativos de vários nós com o EFA

Para executar uma aplicação em um cluster de nós, é necessária a configuração a seguir.

Tópicos

- [Permitir SSH sem senha](#)
- [Criar arquivo de hosts](#)
- [Testes NCCL](#)

Permitir SSH sem senha

Selecione um nó no cluster como o nó líder. Os nós restantes são referidos como nós membros.

1. No nó líder, gere o par de chaves RSA.

```
ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
```

2. Altere as permissões da chave privada no nó líder.

```
chmod 600 ~/.ssh/id_rsa
```

3. Copie a chave pública ~/.ssh/id_rsa.pub e a anexe a ~/.ssh/authorized_keys dos nós membros no cluster.

4. Agora, você poderá fazer login diretamente nos nós membros a partir do nó líder usando o ip privado.

```
ssh <member private ip>
```

5. Desative a strictHostKey verificação e habilite o encaminhamento de agentes no nó principal adicionando o seguinte ao arquivo ~/.ssh/config no nó líder:

```
Host *  
    ForwardAgent yes  
Host *  
    StrictHostKeyChecking no
```

6. Nas instâncias do Amazon Linux 2, execute o seguinte comando no nó líder para fornecer as permissões corretas ao arquivo de configuração:

```
chmod 600 ~/.ssh/config
```

Criar arquivo de hosts

No nó líder, crie um arquivo de hosts para identificar os nós no cluster. O arquivo de hosts deve ter uma entrada para cada nó no cluster. Crie um arquivo ~/.hosts e adicione cada nó usando o ip privado da seguinte forma:

```
localhost slots=8  
<private ip of node 1> slots=8  
<private ip of node 2> slots=8
```

Testes NCCL

Note

Esses testes foram executados usando o EFA versão 1.38.0 e o OFI NCCL Plugin 1.13.2.

Abaixo está listado um subconjunto de testes NCCL fornecidos pela NVIDIA para testar a funcionalidade e o desempenho em vários nós de computação.

Instâncias suportadas: P3dn, P4, P5, P5e, P5en

Testes de desempenho

Teste de desempenho de vários nós do NCCL em P4d.24xlarge

Para verificar o desempenho do NCCL com o EFA, execute o teste de desempenho do NCCL padrão que está disponível no [NCCL-Tests Repo](#). A DLAMI vem com esse teste já criado para CUDA XX.X. Também é possível executar um script próprio com o EFA.

Ao criar seu próprio script, consulte as seguintes orientações:

- Use o caminho completo para mpirun como mostrado no exemplo ao executar aplicativos NCCL com o EFA.
- Altere os parâmetros np e N com base no número de instâncias e GPUs no seu cluster.
- Adicione o sinalizador NCCL_DEBUG=INFO e verifique se os logs indicam o uso do EFA como “O provedor selecionado é EFA”.
- Defina o local do log de treinamento a ser analisado para validação.

```
TRAINING_LOG="testEFA_$(date +%N%).log"
```

Use o comando `watch nvidia-smi` em qualquer um dos nós membros para monitorar o uso da GPU. Os comandos `watch nvidia-smi` a seguir são para o CUDA versão xx.x e dependem do sistema operacional da instância. Você pode executar os comandos para qualquer versão CUDA disponível em sua EC2 instância da Amazon substituindo a versão CUDA no script.

- Amazon Linux 2, Amazon Linux 2023:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib64:/opt/amazon/openmpi/
lib64:$LD_LIBRARY_PATH \
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
none \
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
100 | tee ${TRAINING_LOG}
```

- Ubuntu 20.04, Ubuntu 22.04:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
```

```
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib:/opt/amazon/openmpi/
lib:$LD_LIBRARY_PATH \
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
none \
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
100 | tee ${TRAINING_LOG}
```

A saída será semelhante a:

```
# nThread 1 nGpus 1 minBytes 8 maxBytes 1073741824 step: 2(factor) warmup iters: 5
iters: 100 agg iters: 1 validation: 1 graph: 0
#
# Using devices
# Rank 0 Group 0 Pid 33378 on ip-172-31-42-25 device 0 [0x10] NVIDIA A100-
SXM4-40GB
# Rank 1 Group 0 Pid 33379 on ip-172-31-42-25 device 1 [0x10] NVIDIA A100-
SXM4-40GB
# Rank 2 Group 0 Pid 33380 on ip-172-31-42-25 device 2 [0x20] NVIDIA A100-
SXM4-40GB
# Rank 3 Group 0 Pid 33381 on ip-172-31-42-25 device 3 [0x20] NVIDIA A100-
SXM4-40GB
# Rank 4 Group 0 Pid 33382 on ip-172-31-42-25 device 4 [0x90] NVIDIA A100-
SXM4-40GB
# Rank 5 Group 0 Pid 33383 on ip-172-31-42-25 device 5 [0x90] NVIDIA A100-
SXM4-40GB
# Rank 6 Group 0 Pid 33384 on ip-172-31-42-25 device 6 [0xa0] NVIDIA A100-
SXM4-40GB
# Rank 7 Group 0 Pid 33385 on ip-172-31-42-25 device 7 [0xa0] NVIDIA A100-
SXM4-40GB
# Rank 8 Group 0 Pid 30378 on ip-172-31-43-8 device 0 [0x10] NVIDIA A100-SXM4-40GB
# Rank 9 Group 0 Pid 30379 on ip-172-31-43-8 device 1 [0x10] NVIDIA A100-SXM4-40GB
# Rank 10 Group 0 Pid 30380 on ip-172-31-43-8 device 2 [0x20] NVIDIA A100-SXM4-40GB
# Rank 11 Group 0 Pid 30381 on ip-172-31-43-8 device 3 [0x20] NVIDIA A100-SXM4-40GB
# Rank 12 Group 0 Pid 30382 on ip-172-31-43-8 device 4 [0x90] NVIDIA A100-SXM4-40GB
# Rank 13 Group 0 Pid 30383 on ip-172-31-43-8 device 5 [0x90] NVIDIA A100-SXM4-40GB
# Rank 14 Group 0 Pid 30384 on ip-172-31-43-8 device 6 [0xa0] NVIDIA A100-SXM4-40GB
# Rank 15 Group 0 Pid 30385 on ip-172-31-43-8 device 7 [0xa0] NVIDIA A100-SXM4-40GB
ip-172-31-42-25:33385:33385 [7] NCCL INFO cudaDriverVersion 12060
ip-172-31-43-8:30383:30383 [5] NCCL INFO Bootstrap : Using ens32:172.31.43.8
ip-172-31-43-8:30383:30383 [5] NCCL INFO NCCL version 2.23.4+cuda12.5
```

```

...
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.13.2-aws
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Using Libfabric version 1.22
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Using CUDA driver version 12060 with
runtime 12050
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
variable to 1
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting NCCL_NVLSTREE_MAX_CHUNKSIZE
to 512KiB
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting NCCL_NVLS_CHUNKSIZE to 512KiB
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Running on p4d.24xlarge platform,
Setting NCCL_TOPO_FILE environment variable to /opt/amazon/of-nccl/share/aws-ofi-
nccl/xml/p4d-24x1-topo.xml

```

```

...
-----some output truncated-----

```

#	in-place				out-of-place					
#	size	count	type	redop	root	time	algbw	busbw	#wrong	
#	time	algbw	busbw	#wrong						
#	(us)	(B)	(elements)			(us)	(GB/s)	(GB/s)		
		8	2	float	sum	-1	180.3	0.00	0.00	0
179.3	0.00	0.00	0							
		16	4	float	sum	-1	178.1	0.00	0.00	0
177.6	0.00	0.00	0							
		32	8	float	sum	-1	178.5	0.00	0.00	0
177.9	0.00	0.00	0							
		64	16	float	sum	-1	178.8	0.00	0.00	0
178.7	0.00	0.00	0							
		128	32	float	sum	-1	178.2	0.00	0.00	0
177.8	0.00	0.00	0							
		256	64	float	sum	-1	178.6	0.00	0.00	0
178.8	0.00	0.00	0							
		512	128	float	sum	-1	177.2	0.00	0.01	0
177.1	0.00	0.01	0							
		1024	256	float	sum	-1	179.2	0.01	0.01	0
179.3	0.01	0.01	0							
		2048	512	float	sum	-1	181.3	0.01	0.02	0
181.2	0.01	0.02	0							
		4096	1024	float	sum	-1	184.2	0.02	0.04	0
183.9	0.02	0.04	0							

```

      8192      2048      float      sum      -1      191.2      0.04      0.08      0
190.6    0.04    0.08      0
      16384     4096      float      sum      -1      202.5      0.08      0.15      0
202.3    0.08    0.15      0
      32768     8192      float      sum      -1      233.0      0.14      0.26      0
232.1    0.14    0.26      0
      65536    16384      float      sum      -1      238.6      0.27      0.51      0
235.1    0.28    0.52      0
      131072   32768      float      sum      -1      237.2      0.55      1.04      0
236.8    0.55    1.04      0
      262144   65536      float      sum      -1      248.3      1.06      1.98      0
247.0    1.06    1.99      0
      524288   131072     float      sum      -1      309.2      1.70      3.18      0
307.7    1.70    3.20      0
      1048576  262144      float      sum      -1      408.7      2.57      4.81      0
404.3    2.59    4.86      0
      2097152  524288      float      sum      -1      613.5      3.42      6.41      0
607.9    3.45    6.47      0
      4194304  1048576     float      sum      -1      924.5      4.54      8.51      0
914.8    4.58    8.60      0
      8388608  2097152     float      sum      -1      1059.5     7.92     14.85     0
1054.3    7.96    14.92     0
      16777216 4194304     float      sum      -1      1269.9     13.21    24.77     0
1272.0    13.19   24.73     0
      33554432 8388608     float      sum      -1      1642.7     20.43    38.30     0
1636.7    20.50   38.44     0
      67108864 16777216    float      sum      -1      2446.7     27.43    51.43     0
2445.8    27.44   51.45     0
      134217728 33554432    float      sum      -1      4143.6     32.39    60.73     0
4142.4    32.40   60.75     0
      268435456 67108864    float      sum      -1      7351.9     36.51    68.46     0
7346.7    36.54   68.51     0
      536870912 134217728   float      sum      -1      13717     39.14    73.39     0
13703     39.18   73.46     0
      1073741824 268435456   float      sum      -1      26416     40.65    76.21     0
26420     40.64   76.20     0
...
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 15.5514

```

Testes de validação

Para validar se os testes do EFA retornaram um resultado válido, use os seguintes testes:

- Obtenha o tipo de instância usando os metadados da EC2 instância:

```
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCE_TYPE=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/instance-type)
```

- Execute a [Testes de desempenho](#)
- Defina os seguintes parâmetros:

```
CUDA_VERSION
CUDA_RUNTIME_VERSION
NCCL_VERSION
```

- Valide os resultados conforme mostrado:

```
RETURN_VAL=`echo $?`
if [ ${RETURN_VAL} -eq 0 ]; then

    # [0] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.13.2-aws
    # [0] NCCL INFO NET/OFI Using CUDA driver version 12060 with runtime 12010

    # cudaDriverVersion 12060 --> This is max supported cuda version by nvidia
    driver
    # NCCL version 2.23.4+cuda12.5 --> This is NCCL version compiled with cuda
    version

    # Validation of logs
    grep "NET/OFI Configuring AWS-specific options" ${TRAINING_LOG} || { echo "AWS-
specific options text not found"; exit 1; }
    grep "busbw" ${TRAINING_LOG} || { echo "busbw text not found"; exit 1; }
    grep "Avg bus bandwidth " ${TRAINING_LOG} || { echo "Avg bus bandwidth text not
found"; exit 1; }
    grep "NCCL version $NCCL_VERSION" ${TRAINING_LOG} || { echo "Text not found: NCCL
version $NCCL_VERSION"; exit 1; }
    if [[ ${INSTANCE_TYPE} == "p4d.24xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Text not found: NET/
Libfabric/0/GDRDMA"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Selected Provider is efa text not found"; exit 1; }
        elif [[ ${INSTANCE_TYPE} == "p4de.24xlarge" ]]; then
            grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
```

```

    grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p5.48xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
        elif [[ ${INSTANCE_TYPE} == "p5e.48xlarge" ]]; then
            grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
            grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
            elif [[ ${INSTANCE_TYPE} == "p5en.48xlarge" ]]; then
                grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
                grep "NET/OFI Selected Provider is efa (found 16 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
                elif [[ ${INSTANCE_TYPE} == "p3dn.24xlarge" ]]; then
                    grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Selected Provider is efa text not found"; exit 1; }
                fi
            echo "***** check_efa_nccl_all_reduce passed for cuda
version ${CUDA_VERSION} *****"
        else
            echo "***** check_efa_nccl_all_reduce failed for cuda
version ${CUDA_VERSION} *****"
        fi
    fi

```

- Para acessar os dados de referência, podemos analisar a linha final da saída da tabela do teste all_reduce de vários nós:

```

benchmark=$(sudo cat ${TRAINING_LOG} | grep '1073741824' | tail -n1 | awk -F " "
'{{print $12}}' | sed 's/ //' | sed 's/ 5e-07//')
if [[ -z "${benchmark}" ]]; then
    echo "benchmark variable is empty"
    exit 1
fi

echo "Benchmark throughput: ${benchmark}"

```

Monitoramento e otimização de GPU

A seção a seguir orientará você durante a otimização de GPU e opções de monitoramento.

Esta seção é organizada como um fluxo de trabalho típico com monitoramento, supervisão, pré-processamento e treinamento.

- [Monitoramento](#)
 - [Monitor GPUs com CloudWatch](#)
- [Otimização](#)
 - [Pré-processamento](#)
 - [Treinamento](#)

Monitoramento

A DLAMI vem pré-instalada com várias ferramentas de monitoramento de GPU. Este guia menciona também ferramentas que estão disponíveis para download e instalação.

- [Monitor GPUs com CloudWatch](#) - um utilitário pré-instalado que reporta estatísticas de uso da GPU para a Amazon. CloudWatch
- [CLI nvidia-smi](#) - um utilitário para monitorar a utilização geral de computação e memória de GPU. Isso está pré-instalado no seu AMIs de deep learning da AWS (DLAMI).
- [Biblioteca NVML C](#) - uma API baseada em C para acessar diretamente funções de monitoramento e gerenciamento de GPU. Isso é usado pela CLI nvidia-smi nos bastidores e é pré-instalado na DLAMI. Também tem associações Python e Perl para facilitar o desenvolvimento nessas linguagens. O utilitário gpumon.py pré-instalado em seu DLAMI usa o pacote pynvml do. [nvidia-ml-py](#)
- [NVIDIA DCGM](#) - uma ferramenta de gerenciamento de cluster. Visite a página do desenvolvedor para saber como instalar e configurar essa ferramenta.

Tip

Confira o blog do desenvolvedor de NVIDIA para obter as informações mais recentes sobre como usar as ferramentas do CUDA instaladas na DLAMI:

- [Monitorando TensorCore a utilização usando o Nsight IDE e o nvprof.](#)

Monitor GPUs com CloudWatch

Ao usar a DLAMI com uma GPU, talvez você descubra que está procurando maneiras de controlar o uso durante o treinamento ou a inferência. Isso pode ser útil para otimizar o pipeline de dados e ajustar sua rede de aprendizado profundo.

Há duas maneiras de configurar as métricas da GPU com CloudWatch:

- [Configurar métricas com o AWS CloudWatch agente \(recomendado\)](#)
- [Configurar métricas com o script `gpumon.py` pré-instalado](#)

Configurar métricas com o AWS CloudWatch agente (recomendado)

Integre seu DLAMI com [o agente CloudWatch unificado](#) para configurar métricas de GPU e monitorar a utilização de coprocessos de GPU em instâncias aceleradas da Amazon. EC2

Há quatro maneiras de configurar [métricas de GPU](#) com a DLAMI:

- [Configurar métricas de GPU mínimas](#)
- [Configurar métricas de GPU parciais](#)
- [Configurar todas as métricas de GPU disponíveis](#)
- [Configurar métricas de GPU personalizadas](#)

Para obter mais informações sobre atualizações e patches de segurança, consulte [Patches de segurança para o agente AWS CloudWatch](#)

Pré-requisitos

Para começar, você deve configurar as permissões do IAM da EC2 instância Amazon que permitam que sua instância envie métricas para CloudWatch. Para ver etapas detalhadas, consulte [Criar funções e usuários do IAM para uso com o CloudWatch agente](#).

Configurar métricas de GPU mínimas

Configure métricas mínimas de GPU usando o serviço `dlami-cloudwatch-agent@minimal systemd`. Esse serviço configura as seguintes métricas:

- `utilization_gpu`
- `utilization_memory`

Você pode encontrar o serviço `systemd` para métricas mínimas de GPU pré-configuradas no seguinte local:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-minimal.json
```

Habilite e inicie o serviço `systemd` com os seguintes comandos:

```
sudo systemctl enable dlami-cloudwatch-agent@minimal
sudo systemctl start dlami-cloudwatch-agent@minimal
```

Configurar métricas de GPU parciais

Configure métricas de GPU parciais usando o serviço `dlami-cloudwatch-agent@partial` `systemd`. Esse serviço configura as seguintes métricas:

- `utilization_gpu`
- `utilization_memory`
- `memory_total`
- `memory_used`
- `memory_free`

Você pode encontrar o serviço `systemd` para métricas parciais de GPU pré-configuradas no seguinte local:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-partial.json
```

Habilite e inicie o serviço `systemd` com os seguintes comandos:

```
sudo systemctl enable dlami-cloudwatch-agent@partial
sudo systemctl start dlami-cloudwatch-agent@partial
```

Configurar todas as métricas de GPU disponíveis

Configure todas as métricas de GPU disponíveis usando o serviço `dlami-cloudwatch-agent@all` `systemd`. Esse serviço configura as seguintes métricas:

- `utilization_gpu`
- `utilization_memory`

- memory_total
- memory_used
- memory_free
- temperature_gpu
- power_draw
- fan_speed
- pcie_link_gen_current
- pcie_link_width_current
- encoder_stats_session_count
- encoder_stats_average_fps
- encoder_stats_average_latency
- clocks_current_graphics
- clocks_current_sm
- clocks_current_memory
- clocks_current_video

Você pode encontrar o serviço `systemd` para todas as métricas disponíveis de GPU pré-configuradas no seguinte local:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-all.json
```

Habilite e inicie o serviço `systemd` com os seguintes comandos:

```
sudo systemctl enable dlami-cloudwatch-agent@all  
sudo systemctl start dlami-cloudwatch-agent@all
```

Configurar métricas de GPU personalizadas

Se as métricas pré-configuradas não atenderem aos seus requisitos, você poderá criar um arquivo personalizado de configuração do CloudWatch agente.

Criar um arquivo de configuração personalizada

Para criar um arquivo de configuração personalizado, consulte as etapas detalhadas em [Criar ou editar manualmente o arquivo de configuração do CloudWatch agente](#).

Neste exemplo, suponha que a definição do esquema esteja localizada em `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`.

Configurar métricas com seu arquivo personalizado

Execute o comando a seguir para configurar o CloudWatch agente de acordo com seu arquivo personalizado:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
-a fetch-config -m ec2 -s -c \
file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
```

Patches de segurança para o agente AWS CloudWatch

Os recém-lançados DLAMIs são configurados com os patches de segurança de AWS CloudWatch agentes mais recentes disponíveis. Consulte as seções a seguir para atualizar a DLAMI atual com os patches de segurança mais recentes, dependendo do sistema operacional escolhido.

Amazon Linux 2

Use yum para obter os patches de segurança de AWS CloudWatch agentes mais recentes para um Amazon Linux 2 DLAMI.

```
sudo yum update
```

Ubuntu

Para obter os patches de AWS CloudWatch segurança mais recentes para um DLAMI com Ubuntu, é necessário reinstalar o agente usando um link de download AWS CloudWatch do Amazon S3.

```
wget https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/
amazon-cloudwatch-agent.deb
```

Para obter mais informações sobre como instalar o AWS CloudWatch agente usando os links de download do Amazon S3, consulte [Instalando e executando o CloudWatch agente em seus servidores](#).

Configurar métricas com o script **gpumon.py** pré-instalado

Um utilitário chamado gpumon.py é pré-instalado na DLAMI. Ele se integra CloudWatch e oferece suporte ao monitoramento do uso por GPU: memória da GPU, temperatura da GPU e potência da

GPU. O script envia periodicamente os dados monitorados para CloudWatch o. Você pode configurar o nível de granularidade dos dados enviados CloudWatch alterando algumas configurações no script. Antes de iniciar o script, no entanto, você precisará configurar CloudWatch para receber as métricas.

Como configurar e executar o monitoramento de GPU com CloudWatch

1. Crie um usuário do IAM ou modifique um existente para ter uma política para publicar a métrica CloudWatch. Se você criar um novo usuário, anote as credenciais, pois elas serão necessárias na próxima etapa.

A política do IAM a ser pesquisada é “cloudwatch:PutMetricData”. A política que é adicionada é a seguinte:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Tip

Para obter mais informações sobre como criar um usuário do IAM e adicionar políticas para CloudWatch, consulte a [CloudWatch documentação](#).

2. Em sua DLAMI, execute [AWS configure](#) e especifique as credenciais de usuário do IAM.

```
$ aws configure
```

3. Talvez você precise fazer algumas modificações no utilitário gpumon antes de executá-lo. Você pode encontrar o utilitário gpumon e o README no seguinte local definido no seguinte bloco de código. Para obter mais informações sobre o script gpumon.py, consulte a [localização do script no Amazon S3](#).

```
Folder: ~/tools/GPUCloudWatchMonitor
Files:  ~/tools/GPUCloudWatchMonitor/gpumon.py
        ~/tools/GPUCloudWatchMonitor/README
```

Opções:

- Altere a região no gpumon.py se sua instância NÃO estiver em us-east-1.
 - Altere outros parâmetros, como o período do relatório CloudWatch namespace ou o período do relatório, com `store_reso`.
4. No momento, o script oferece suporte apenas ao Python 3. Ative o ambiente do Python 3 da estrutura de trabalho preferencial ou ative o ambiente geral do Python 3 da DLAMI.

```
$ source activate python3
```

5. Execute o utilitário gpumon em segundo plano.

```
(python3)$ python gpumon.py &
```

6. Abra seu navegador para <https://console.aws.amazon.com/cloudwatch/> e, depois, selecione a métrica. Ele terá um namespace ". DeepLearningTrain

 Tip

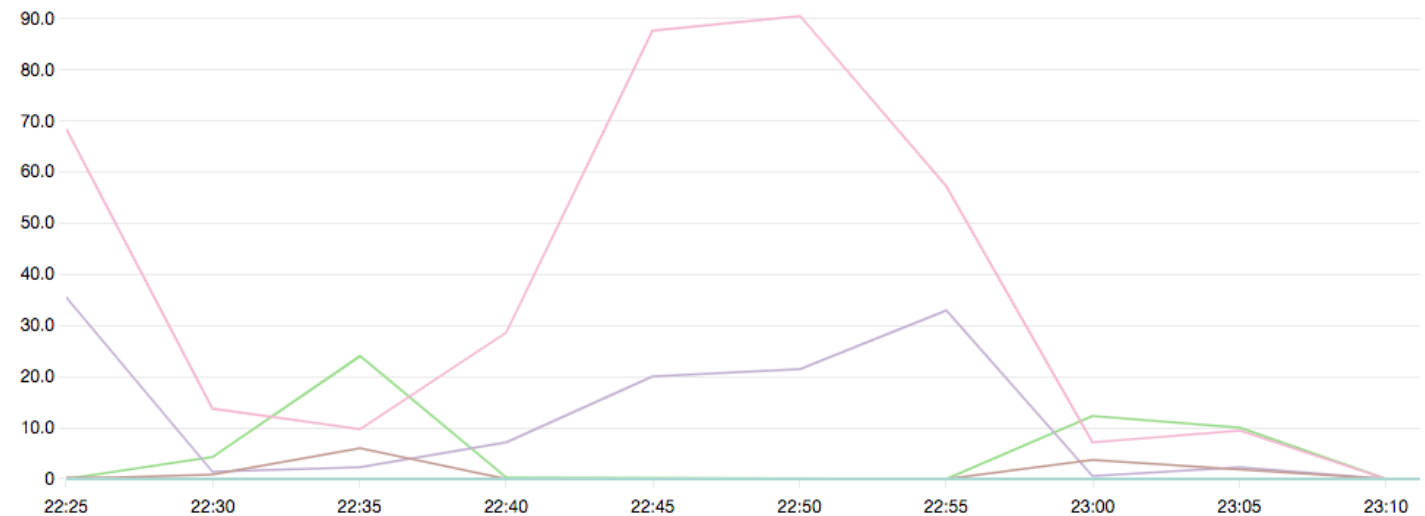
Você pode alterar o namespace modificando o gpumon.py. Você também pode modificar o intervalo de relatório ajustando `store_reso`.

Veja a seguir um exemplo de CloudWatch gráfico relatando uma execução do gpumon.py monitorando um trabalho de treinamento na instância p2.8xlarge.

GPU usage, Memory usage

1h 3h 12h 1d 3d 1w custom

Various units



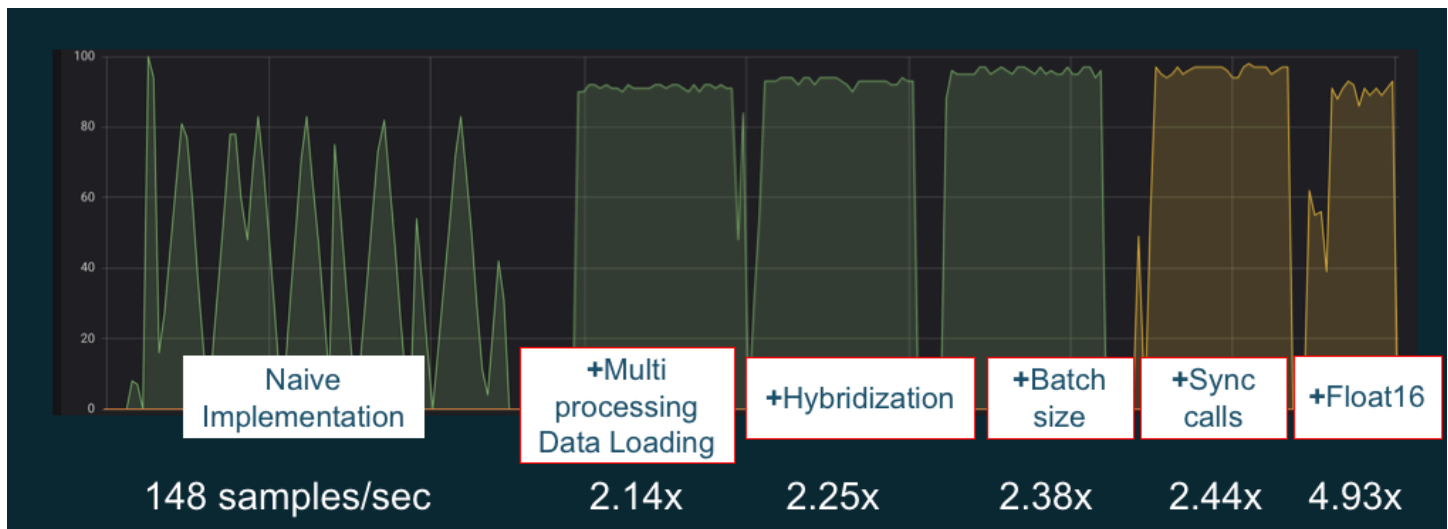
Você pode estar interessado nesses outros tópicos sobre monitoramento e otimização de GPU:

- [Monitoramento](#)
 - [Monitor GPUs com CloudWatch](#)
- [Otimização](#)
 - [Pré-processamento](#)
 - [Treinamento](#)

Otimização

Para aproveitar ao máximo GPUs, você pode otimizar seu pipeline de dados e ajustar sua rede de aprendizado profundo. Como o gráfico a seguir descreve, uma implementação simples ou básica de uma rede neural pode usar a GPU de maneira inconsistente e não aproveitar o potencial máximo. Ao otimizar o pré-processamento e o carregamento de dados, você pode reduzir o gargalo da CPU para a GPU. Você pode ajustar a rede neural em si usando hibridização (quando compatível com a estrutura), ajustando o tamanho do lote e sincronizando as chamadas. Você também pode usar treinamento de precisão múltipla (float16 ou int8) na maioria das estruturas, o que pode causar um efeito enorme na melhoria da taxa de transferência.

A tabela a seguir mostra os ganhos de desempenho cumulativos ao aplicar otimizações diferentes. Seus resultados dependerão dos dados que você está processando e da rede que está otimizando.



Exemplos de otimização de desempenho de GPUs. Fonte do gráfico: [Truques de desempenho com MXNet Gluon](#)

Os seguintes guias apresentam opções que funcionarão com a DLAMI e ajudarão a aumentar o desempenho da GPU.

Tópicos

- [Pré-processamento](#)
- [Treinamento](#)

Pré-processamento

O pré-processamento de dados durante transformações ou aumento muitas vezes pode ser um processo vinculado à CPU e isso pode ser o gargalo em todo o seu pipeline. Estruturas têm operadores integrados para processamento de imagens, mas a DALI (Data Augmentation Library) demonstra um melhor desempenho em relação às opções integradas das estruturas.

- NVIDIA Data Augmentation Library (DALI): a DALI minimiza o aumento dos dados para a GPU. Não é pré-instalada na DLAMI, mas você pode ter acesso instalando-a ou carregando um contêiner de estrutura compatível na sua DLAMI ou em outra instância do Amazon Elastic Compute Cloud. Consulte a [página do projeto DALI](#) no site do NVIDIA para obter detalhes. Para ver um exemplo de caso de uso e baixar amostras de código, consulte a amostra de desempenho do [treinamento SageMaker de pré-processamento](#).
- nvJPEG: uma biblioteca de decodificadores JPEG acelerados para GPU para programadores de C. Ela oferece suporte à decodificação de imagens ou lotes únicos, bem como operações

de transformação subsequentes que são comuns em aprendizado profundo. A nvJPEG vem integrada à DALI, ou você pode fazer download da [página nvjpeg do site da NVIDIA](#) e usá-la separadamente.

Você pode estar interessado nesses outros tópicos sobre monitoramento e otimização de GPU:

- [Monitoramento](#)
 - [Monitor GPUs com CloudWatch](#)
- [Otimização](#)
 - [Pré-processamento](#)
 - [Treinamento](#)

Treinamento

Com treinamento de precisão mista, você pode implantar redes maiores com a mesma quantidade de memória ou reduzir o uso de memória em comparação com sua rede de precisão única ou dupla, e você verá aumentos de desempenho de computação. Você também pode obter o benefício de transferências de dados menores e mais rápidas, um fator importante em um treinamento distribuído em vários nós. Para aproveitar o treinamento de precisão mista, você precisa ajustar a conversão de dados e a escalabilidade de perdas. Veja a seguir os guias que descrevem como fazer isso para as estruturas que oferecem suporte à precisão mista.

- [SDK de aprendizado profundo da NVIDIA](#) - documentos no site da NVIDIA descrevendo a implementação de precisão mista para, e. MXNet PyTorch TensorFlow

Tip

Verifique se o site oferece a estrutura de sua preferência e procure por "precisão mista" ou "fp16" para encontrar as técnicas de otimização mais recentes. Veja a seguir alguns guias sobre precisão mista que podem ser úteis:

- [Treinamento de precisão mista com TensorFlow \(vídeo\)](#) - no blog da NVIDIA.
- [Treinamento de precisão mista usando float16 com MXNet](#) - um artigo de perguntas frequentes no site. MXNet

- [NVIDIA Apex: uma ferramenta para treinamento fácil de precisão mista com PyTorch](#) - um artigo de blog no site da NVIDIA.

Você pode estar interessado nesses outros tópicos sobre monitoramento e otimização de GPU:

- [Monitoramento](#)
 - [Monitor GPUs com CloudWatch](#)
- [Otimização](#)
 - [Pré-processamento](#)
 - [Treinamento](#)

O chip de AWS inferência com DLAMI

AWS O Inferentia é um chip de aprendizado de máquina personalizado projetado por AWS ele que você pode usar para previsões de inferência de alto desempenho. Para usar o chip, configure uma instância Amazon Elastic Compute Cloud e use o kit de desenvolvimento de software (SDK) AWS Neuron para invocar o chip Inferentia. Para fornecer aos clientes a melhor experiência no Inferentia, o Neuron foi integrado à AMIs de deep learning da AWS (DLAMI).

Os tópicos a seguir mostram como começar a usar o Inferentia com a DLAMI.

Conteúdo

- [Lançamento de uma instância DLAMI com Neuron AWS](#)
- [Usando o DLAMI com Neuron AWS](#)

Lançamento de uma instância DLAMI com Neuron AWS

O DLAMI mais recente está pronto para uso AWS com o Inferentia e vem com AWS o pacote Neuron API. Para iniciar uma instância da DLAMI, consulte [Iniciar e configurar uma DLAMI](#). Depois de ter um DLAMI, use as etapas aqui para garantir que AWS seu chip de inferência AWS e os recursos do Neuron estejam ativos.

Conteúdo

- [Verifique a instância](#)
- [Identificação de AWS dispositivos de inferência](#)

- [Exibir o uso de recursos](#)
- [Como usar o Monitor do Neuron](#)
- [Atualização do software Neuron](#)

Verifique a instância

Antes de usar a instância, verifique se ela está corretamente definida e configurada com o Neuron.

Identificação de AWS dispositivos de inferência

Para identificar o número de dispositivos do Inferentia na sua instância, use o seguinte comando:

```
neuron-ls
```

Se a instância tiver dispositivos do Inferentia conectados a ela, a saída será semelhante à seguinte:

```
+-----+-----+-----+-----+-----+
| NEURON | NEURON | NEURON | CONNECTED | PCI      |
| DEVICE | CORES  | MEMORY | DEVICES   | BDF      |
+-----+-----+-----+-----+-----+
| 0      | 4      | 8 GB   | 1         | 0000:00:1c.0 |
| 1      | 4      | 8 GB   | 2, 0      | 0000:00:1d.0 |
| 2      | 4      | 8 GB   | 3, 1      | 0000:00:1e.0 |
| 3      | 4      | 8 GB   | 2         | 0000:00:1f.0 |
+-----+-----+-----+-----+-----+
```

O resultado fornecido é obtido de uma instância Inf1.6xlarge e inclui as seguintes colunas:

- **DISPOSITIVO NEURONAL:** O ID lógico atribuído ao NeuronDevice. Esse ID é usado ao configurar vários tempos de execução para usar diferentes. NeuronDevices
- **NÚCLEOS DE NEURÔNIOS:** O número de NeuronCores presentes no NeuronDevice.
- **MEMÓRIA NEURONAL:** A quantidade de memória DRAM no. NeuronDevice
- **DISPOSITIVOS CONECTADOS:** Outros NeuronDevices conectados ao NeuronDevice.
- **PCI BDF:** O ID da função de dispositivo de barramento PCI (BDF) do. NeuronDevice

Exibir o uso de recursos

Visualize informações úteis sobre a NeuronCore utilização da vCPU, o uso da memória, os modelos carregados e os aplicativos Neuron com o comando. `neuron-top` O lançamento `neuron-top` sem

argumentos mostrará os dados de todos os aplicativos de aprendizado de máquina que utilizam NeuronCores.

```
neuron-top
```

Quando um aplicativo está usando quatro NeuronCores, a saída deve ser semelhante à imagem a seguir:



Para obter mais informações sobre recursos para monitorar e otimizar aplicações de inferência que usam como base o Neuron, consulte [Ferramentas do Neuron](#).

Como usar o Monitor do Neuron

O Monitor do Neuron coleta métricas dos runtimes do Neuron em execução no sistema e transmite os dados coletados para stdout no formato JSON. Elas são organizadas em grupos de métricas que você configura fornecendo um arquivo de configuração. Para obter mais informações sobre o Monitor do Neuron, consulte o [Guia do usuário do monitor do Neuron](#).

Atualização do software Neuron

[Para obter informações sobre como atualizar o software Neuron SDK no DLAMI, consulte o Guia de configuração do Neuron. AWS](#)

Próxima etapa

[Usando o DLAMI com Neuron AWS](#)

Usando o DLAMI com Neuron AWS

Um fluxo de trabalho típico com o AWS Neuron SDK é compilar um modelo de aprendizado de máquina previamente treinado em um servidor de compilação. Depois disso, distribua os artefatos para as instâncias Inf1 para execução. AMIs de deep learning da AWS (DLAMI) vem pré-instalado com tudo o que você precisa para compilar e executar inferência em uma instância Inf1 que usa Inferentia.

As seções a seguir descrevem como usar a DLAMI com o Inferentia.

Conteúdo

- [Usando TensorFlow -Neuron e o compilador Neuron AWS](#)
- [Usando AWS Neuron Serving TensorFlow](#)
- [Usando MXNet -Neuron e o compilador Neuron AWS](#)
- [Usando o MXNet -Neuron Model Serving](#)
- [Usando PyTorch -Neuron e o compilador Neuron AWS](#)

Usando TensorFlow -Neuron e o compilador Neuron AWS

Este tutorial mostra como usar o compilador AWS Neuron para compilar o modelo Keras ResNet -50 e exportá-lo como um modelo salvo em formato. SavedModel Esse formato é um formato típico de TensorFlow modelo intercambiável. Você também aprenderá a executar a inferência em uma instância do Inf1 com exemplo de entrada.

Para obter mais informações sobre o SDK do Neuron, consulte a [Documentação do SDK do AWS Neuron](#).

Conteúdo

- [Pré-requisitos](#)

- [Ative o ambiente Conda](#)
- [Compilação ResNet50](#)
- [ResNet50 Inferência](#)

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em [Lançamento de uma instância DLAMI com Neuron AWS](#). Também é necessário conhecer a aprendizagem profunda e o uso da DLAMI.

Ative o ambiente Conda

Ative o ambiente TensorFlow -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_tensorflow_p36
```

Para sair do ambiente Conda atual, execute o seguinte comando:

```
source deactivate
```

Compilação ResNet50

Crie um script Python chamado **tensorflow_compile_resnet50.py** com o seguinte conteúdo. Esse script Python compila o modelo Keras ResNet 50 e o exporta como um modelo salvo.

```
import os
import time
import shutil
import tensorflow as tf
import tensorflow.neuron as tfn
import tensorflow.compat.v1.keras as keras
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input

# Create a workspace
WORKSPACE = './ws_resnet50'
```

```

os.makedirs(WORKSPACE, exist_ok=True)

# Prepare export directory (old one removed)
model_dir = os.path.join(WORKSPACE, 'resnet50')
compiled_model_dir = os.path.join(WORKSPACE, 'resnet50_neuron')
shutil.rmtree(model_dir, ignore_errors=True)
shutil.rmtree(compiled_model_dir, ignore_errors=True)

# Instantiate Keras ResNet50 model
keras.backend.set_learning_phase(0)
model = ResNet50(weights='imagenet')

# Export SavedModel
tf.saved_model.simple_save(
    session          = keras.backend.get_session(),
    export_dir       = model_dir,
    inputs           = {'input': model.inputs[0]},
    outputs          = {'output': model.outputs[0]})

# Compile using Neuron
tf.nn.saved_model.compile(model_dir, compiled_model_dir)

# Prepare SavedModel for uploading to Inf1 instance
shutil.make_archive(compiled_model_dir, 'zip', WORKSPACE, 'resnet50_neuron')

```

Compile o modelo usando o seguinte comando:

```
python tensorflow_compile_resnet50.py
```

O processo de compilação leva alguns minutos. Quando concluído, sua saída será semelhante a:

```

...
INFO:tensorflow:fusing subgraph neuron_op_d6f098c01c780733 with neuron-cc
INFO:tensorflow:Number of operations in TensorFlow session: 4638
INFO:tensorflow:Number of operations after tf.neuron optimizations: 556
INFO:tensorflow:Number of operations placed on Neuron runtime: 554
INFO:tensorflow:Successfully converted ./ws_resnet50/resnet50 to ./ws_resnet50/
resnet50_neuron
...

```

Após a compilação, o modelo salvo será compactado em **ws_resnet50/resnet50_neuron.zip**. Descompacte o modelo e faça download da imagem de exemplo para a inferência, usando os seguintes comandos:

```
unzip ws_resnet50/resnet50_neuron.zip -d .
curl -O https://raw.githubusercontent.com/aws-labs/mxnet-model-server/master/docs/
images/kitten_small.jpg
```

ResNet50 Inferência

Crie um script Python chamado **tensorflow_infer_resnet50.py** com o seguinte conteúdo. Esse script executa a inferência no modelo obtido por download usando um modelo de inferência previamente compilado.

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import resnet50

# Create input from image
img_sgl = image.load_img('kitten_small.jpg', target_size=(224, 224))
img_arr = image.img_to_array(img_sgl)
img_arr2 = np.expand_dims(img_arr, axis=0)
img_arr3 = resnet50.preprocess_input(img_arr2)
# Load model
COMPILED_MODEL_DIR = './ws_resnet50/resnet50_neuron/'
predictor_inferentia = tf.contrib.predictor.from_saved_model(COMPILED_MODEL_DIR)
# Run inference
model_feed_dict={'input': img_arr3}
infa_rslts = predictor_inferentia(model_feed_dict);
# Display results
print(resnet50.decode_predictions(infa_rslts["output"], top=5)[0])
```

Execute a inferência no modelo usando o seguinte comando:

```
python tensorflow_infer_resnet50.py
```

A saída será semelhante a:


```
...  
[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',  
 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',  
 'snow_leopard', 0.009290541)]
```

Próxima etapa

[Usando AWS Neuron Serving TensorFlow](#)

Usando AWS Neuron Serving TensorFlow

Este tutorial mostra como construir um gráfico e adicionar uma etapa de compilação do AWS Neuron antes de exportar o modelo salvo para uso com o Serving. TensorFlow TensorFlow Serving é um sistema de atendimento que permite ampliar a inferência em uma rede. O Neuron TensorFlow Serving usa a mesma API do TensorFlow Serving normal. A única diferença é que um modelo salvo deve ser compilado para AWS Inferência e o ponto de entrada é um binário diferente chamado `tensorflow_model_server_neuron`. O binário é encontrado em `/usr/local/bin/tensorflow_model_server_neuron` e é pré-instalado na DLAMI.

Para obter mais informações sobre o SDK do Neuron, consulte a [Documentação do SDK do AWS Neuron](#).

Conteúdo

- [Pré-requisitos](#)
- [Ative o ambiente Conda](#)
- [Compile e exporte o modelo salvo](#)
- [Fornecer o modelo salvo](#)
- [Gerar solicitações de inferência para o modelo de servidor](#)

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em [Lançamento de uma instância DLAMI com Neuron AWS](#). Também é necessário conhecer a aprendizagem profunda e o uso da DLAMI.

Ative o ambiente Conda

Ative o ambiente TensorFlow -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_tensorflow_p36
```

Se você precisar sair do ambiente Conda atual, execute:

```
source deactivate
```

Compile e exporte o modelo salvo

Crie um script Python chamado `tensorflow-model-server-compile.py` com o conteúdo a seguir. Ele constrói um gráfico e o compila usando o Neuron. Depois, exporta o gráfico compilado como modelo salvo.

```
import tensorflow as tf
import tensorflow.neuron
import os

tf.keras.backend.set_learning_phase(0)
model = tf.keras.applications.ResNet50(weights='imagenet')
sess = tf.keras.backend.get_session()
inputs = {'input': model.inputs[0]}
outputs = {'output': model.outputs[0]}

# save the model using tf.saved_model.simple_save
model_dir = "./resnet50/1"
tf.saved_model.simple_save(sess, model_dir, inputs, outputs)

# compile the model for Inferentia
neuron_model_dir = os.path.join(os.path.expanduser('~'), 'resnet50_inf1', '1')
tf.neuron.saved_model.compile(model_dir, neuron_model_dir, batch_size=1)
```

Compile o modelo usando o seguinte comando:

```
python tensorflow-model-server-compile.py
```

A saída será semelhante a:

```
...
INFO:tensorflow:fusing subgraph neuron_op_d6f098c01c780733 with neuron-cc
INFO:tensorflow:Number of operations in TensorFlow session: 4638
INFO:tensorflow:Number of operations after tf.neuron optimizations: 556
INFO:tensorflow:Number of operations placed on Neuron runtime: 554
INFO:tensorflow:Successfully converted ./resnet50/1 to /home/ubuntu/resnet50_inf1/1
```

Fornecer o modelo salvo

Depois que o modelo foi compilado, você pode usar o seguinte comando para fornecer o modelo salvo com o binário `tensorflow_model_server_neuron`:

```
tensorflow_model_server_neuron --model_name=resnet50_inf1 \
  --model_base_path=$HOME/resnet50_inf1/ --port=8500 &
```

A saída será semelhante a: O modelo compilado é preparado na DRAM do dispositivo do Inferentia, pelo servidor, para preparar para a inferência.

```
...
2019-11-22 01:20:32.075856: I external/org_tensorflow/tensorflow/cc/saved_model/
loader.cc:311] SavedModel load for tags { serve }; Status: success. Took 40764
microseconds.
2019-11-22 01:20:32.075888: I tensorflow_serving/servables/tensorflow/
saved_model_warmup.cc:105] No warmup data file found at /home/ubuntu/resnet50_inf1/1/
assets.extra/tf_serving_warmup_requests
2019-11-22 01:20:32.075950: I tensorflow_serving/core/loader_harness.cc:87]
Successfully loaded servable version {name: resnet50_inf1 version: 1}
2019-11-22 01:20:32.077859: I tensorflow_serving/model_servers/
server.cc:353] Running gRPC ModelServer at 0.0.0.0:8500 ...
```

Gerar solicitações de inferência para o modelo de servidor

Crie um script Python chamado `tensorflow-model-server-infer.py` com o conteúdo a seguir. Esse script executa a inferência via gRPC, que é um framework de serviço.

```
import numpy as np
import grpc
import tensorflow as tf
```

```

from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc
from tensorflow.keras.applications.resnet50 import decode_predictions

if __name__ == '__main__':
    channel = grpc.insecure_channel('localhost:8500')
    stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
    img_file = tf.keras.utils.get_file(
        "./kitten_small.jpg",
        "https://raw.githubusercontent.com/aws-labs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
    img = image.load_img(img_file, target_size=(224, 224))
    img_array = preprocess_input(image.img_to_array(img)[None, ...])
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'resnet50_inf1'
    request.inputs['input'].CopyFrom(
        tf.contrib.util.make_tensor_proto(img_array, shape=img_array.shape))
    result = stub.Predict(request)
    prediction = tf.make_ndarray(result.outputs['output'])
    print(decode_predictions(prediction))

```

Execute a inferência no modelo usando gRPC com o seguinte comando:

```
python tensorflow-model-server-infer.py
```

A saída será semelhante a:

```

[(['n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
'snow_leopard', 0.009290541)]]

```

Usando MXNet -Neuron e o compilador Neuron AWS

A API de compilação MXNet -Neuron fornece um método para compilar um gráfico de modelo que você pode executar em um dispositivo Inferentia. AWS

Neste exemplo, você usa a API para compilar um modelo ResNet -50 e usá-lo para executar inferência.

Para obter mais informações sobre o SDK do Neuron, consulte a [Documentação do SDK do AWS Neuron](#).

Conteúdo

- [Pré-requisitos](#)
- [Ative o ambiente Conda](#)
- [Compilação ResNet50](#)
- [ResNet50 Inferência](#)

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em [Lançamento de uma instância DLAMI com Neuron AWS](#). Também é necessário conhecer a aprendizagem profunda e o uso da DLAMI.

Ative o ambiente Conda

Ative o ambiente MXNet -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_mxnet_p36
```

Para sair do ambiente Conda atual, execute:

```
source deactivate
```

Compilação ResNet50

Crie um script Python chamado **mxnet_compile_resnet50.py** com o conteúdo a seguir. Esse script usa a API Python de compilação MXNet -Neuron para compilar um modelo -50. ResNet

```
import mxnet as mx
import numpy as np

print("downloading...")
path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
```

```
print("download finished.")

sym, args, aux = mx.model.load_checkpoint('resnet-50', 0)

print("compile for inferentia using neuron... this will take a few minutes...")
inputs = { "data" : mx.nd.ones([1,3,224,224], name='data', dtype='float32') }

sym, args, aux = mx.contrib.neuron.compile(sym, args, aux, inputs)

print("save compiled model...")
mx.model.save_checkpoint("compiled_resnet50", 0, sym, args, aux)
```

Compile o modelo usando o seguinte comando:

```
python mxnet_compile_resnet50.py
```

A compilação demora alguns minutos. Quando ela terminar, os seguintes arquivos estarão no diretório atual:

```
resnet-50-0000.params
resnet-50-symbol.json
compiled_resnet50-0000.params
compiled_resnet50-symbol.json
```

ResNet50 Inferência

Crie um script Python chamado **mxnet_infer_resnet50.py** com o conteúdo a seguir. Esse script faz download de uma imagem de amostra e a usa para executar a inferência com o modelo compilado.

```
import mxnet as mx
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'synset.txt')

fname = mx.test_utils.download('https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/images/kitten_small.jpg')
img = mx.image.imread(fname)
```

```
# convert into format (batch, RGB, width, height)
img = mx.image.imresize(img, 224, 224)
# resize
img = img.transpose((2, 0, 1))
# Channel first
img = img.expand_dims(axis=0)
# batchify
img = img.astype(dtype='float32')

sym, args, aux = mx.model.load_checkpoint('compiled_resnet50', 0)
softmax = mx.nd.random_normal(shape=(1,))
args['softmax_label'] = softmax
args['data'] = img
# Inferentia context
ctx = mx.neuron()

exe = sym.bind(ctx=ctx, args=args, aux_states=aux, grad_req='null')
with open('synset.txt', 'r') as f:
    labels = [l.rstrip() for l in f]

exe.forward(data=img)
prob = exe.outputs[0].asnumpy()
# print the top-5
prob = np.squeeze(prob)
a = np.argsort(prob)[::-1]
for i in a[0:5]:
    print('probability=%f, class=%s' %(prob[i], labels[i]))
```

Execute a inferência com o modelo compilado usando o seguinte comando:

```
python mxnet_infer_resnet50.py
```

A saída será semelhante a:

```
probability=0.642454, class=n02123045 tabby, tabby cat
probability=0.189407, class=n02123159 tiger cat
probability=0.100798, class=n02124075 Egyptian cat
probability=0.030649, class=n02127052 lynx, catamount
probability=0.016278, class=n02129604 tiger, Panthera tigris
```

Próxima etapa

[Usando o MXNet -Neuron Model Serving](#)

Usando o MXNet -Neuron Model Serving

Neste tutorial, você aprende a usar um MXNet modelo pré-treinado para realizar a classificação de imagens em tempo real com o Multi Model Server (MMS). O MMS é uma easy-to-use ferramenta flexível para servir modelos de aprendizado profundo que são treinados usando qualquer estrutura de aprendizado de máquina ou aprendizado profundo. Este tutorial inclui uma etapa de compilação usando o AWS Neuron e uma implementação do MMS usando. MXNet

Para obter mais informações sobre o SDK do Neuron, consulte a [Documentação do SDK do AWS Neuron](#).

Conteúdo

- [Pré-requisitos](#)
- [Ative o ambiente Conda](#)
- [Faça download do código de exemplo](#)
- [Compile o modelo.](#)
- [Execute a inferência](#)

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em [Lançamento de uma instância DLAMI com Neuron AWS](#). Também é necessário conhecer a aprendizagem profunda e o uso da DLAMI.

Ative o ambiente Conda

Ative o ambiente MXNet -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_mxnet_p36
```

Para sair do ambiente Conda atual, execute:

```
source deactivate
```


Faça download do código de exemplo

Para executar o exemplo, faça download do código de exemplo usando os seguintes comandos:

```
git clone https://github.com/aws-labs/multi-model-server
cd multi-model-server/examples/mxnet_vision
```

Compile o modelo.

Crie um script Python chamado `multi-model-server-compile.py` com o conteúdo a seguir. Esse script compila o modelo ResNet 50 para o alvo do dispositivo Inferentia.

```
import mxnet as mx
from mxnet.contrib import neuron
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
mx.test_utils.download(path+'synset.txt')

nn_name = "resnet-50"

#Load a model
sym, args, auxs = mx.model.load_checkpoint(nn_name, 0)

#Define compilation parameters# - input shape and dtype
inputs = {'data' : mx.nd.zeros([1,3,224,224], dtype='float32')}

# compile graph to inferentia target
csym, cargs, cauxs = neuron.compile(sym, args, auxs, inputs)

# save compiled model
mx.model.save_checkpoint(nn_name + "_compiled", 0, csym, cargs, cauxs)
```

Para compilar o modelo, use o seguinte comando:

```
python multi-model-server-compile.py
```

A saída será semelhante a:

```
...
```

```
[21:18:40] src/nnvm/legacy_json_util.cc:209: Loading symbol saved by previous version
v0.8.0. Attempting to upgrade...
[21:18:40] src/nnvm/legacy_json_util.cc:217: Symbol successfully upgraded!
[21:19:00] src/operator/subgraph/build_subgraph.cc:698: start to execute partition
graph.
[21:19:00] src/nnvm/legacy_json_util.cc:209: Loading symbol saved by previous version
v0.8.0. Attempting to upgrade...
[21:19:00] src/nnvm/legacy_json_util.cc:217: Symbol successfully upgraded!
```

Crie um arquivo chamado `signature.json` com o seguinte conteúdo para configurar o nome e a forma de entrada:

```
{
  "inputs": [
    {
      "data_name": "data",
      "data_shape": [
        1,
        3,
        224,
        224
      ]
    }
  ]
}
```

Faça download do arquivo `synset.txt` usando o seguinte comando: Esse arquivo é uma lista de nomes para classes de ImageNet predição.

```
curl -O https://s3.amazonaws.com/model-server/model_archive_1.0/examples/
squeezenet_v1.1/synset.txt
```

Crie uma classe de serviço personalizada seguindo o modelo na pasta `model_server_template`. Copie o modelo para o diretório de trabalho atual usando o seguinte comando:

```
cp -r ../model_service_template/* .
```

Edite o módulo `mxnet_model_service.py` para substituir o contexto `mx.cpu()` pelo contexto `mx.neuron()`, da seguinte forma. Você também precisa comentar a cópia de dados desnecessária `model_input` porque MXNet-Neuron não suporta o `NDArray` e o `Gluon` APIs

```
...
self.mxnet_ctx = mx.neuron() if gpu_id is None else mx.gpu(gpu_id)
...
#model_input = [item.as_in_context(self.mxnet_ctx) for item in model_input]
```

Empacote o modelo com arquivador de modelos, usando os seguintes comandos:

```
cd ~/multi-model-server/examples
model-archiver --force --model-name resnet-50_compiled --model-path mxnet_vision --
handler mxnet_vision_service:handle
```

Execute a inferência

Inicie o Multi Model Server e carregue o modelo que usa a RESTful API usando os comandos a seguir. Certifique-se de que o neuron-rtd está sendo executado com as configurações padrão.

```
cd ~/multi-model-server/
multi-model-server --start --model-store examples > /dev/null # Pipe to log file if you
  want to keep a log of MMS
curl -v -X POST "http://localhost:8081/models?
  initial_workers=1&max_workers=4&synchronous=true&url=resnet-50_compiled.mar"
sleep 10 # allow sufficient time to load model
```

Execute a inferência usando uma imagem de exemplo com os seguintes comandos:

```
curl -O https://raw.githubusercontent.com/aws-labs/multi-model-server/master/docs/
  images/kitten_small.jpg
curl -X POST http://127.0.0.1:8080/predictions/resnet-50_compiled -T kitten_small.jpg
```

A saída será semelhante a:

```
[
  {
    "probability": 0.6388034820556641,
    "class": "n02123045 tabby, tabby cat"
  },
  {
    "probability": 0.16900072991847992,
    "class": "n02123159 tiger cat"
  },
  {
```

```

    "probability": 0.12221276015043259,
    "class": "n02124075 Egyptian cat"
  },
  {
    "probability": 0.028706775978207588,
    "class": "n02127052 lynx, catamount"
  },
  {
    "probability": 0.01915954425930977,
    "class": "n02129604 tiger, Panthera tigris"
  }
]

```

Para limpar após o teste, emita um comando delete por meio da RESTful API e interrompa o servidor de modelos usando os seguintes comandos:

```

curl -X DELETE http://127.0.0.1:8081/models/resnet-50_compiled

multi-model-server --stop

```

A seguinte saída deverá ser mostrada:

```

{
  "status": "Model \"resnet-50_compiled\" unregistered"
}
Model server stopped.
Found 1 models and 1 NCGs.
Unloading 10001 (MODEL_STATUS_STARTED) :: success
Destroying NCG 1 :: success

```

Usando PyTorch -Neuron e o compilador Neuron AWS

A API de compilação PyTorch -Neuron fornece um método para compilar um gráfico de modelo que você pode executar em um dispositivo Inferentia. AWS

Um modelo treinado deve ser compilado para um destino Inferentia antes que ele possa ser implantado em instâncias Inf1. O tutorial a seguir compila o modelo torchvision ResNet 50 e o exporta como um módulo salvo. TorchScript Esse modelo é, assim sendo, usado para executar inferência.

Por conveniência, este tutorial usa uma instância Inf1 para compilação e inferência. Na prática, você pode compilar o modelo usando outro tipo de instância, como a família de instâncias c5. Depois,

você deve implantar o modelo compilado no servidor de inferência Inf1. Para obter mais informações, consulte a documentação do [AWS Neuron PyTorch SDK](#).

Conteúdo

- [Pré-requisitos](#)
- [Ative o ambiente Conda](#)
- [Compilação ResNet50](#)
- [ResNet50 Inferência](#)

Pré-requisitos

Antes de usar este tutorial, você precisa ter concluído os passos da configuração em [Lançamento de uma instância DLAMI com Neuron AWS](#). Também é necessário conhecer a aprendizagem profunda e o uso da DLAMI.

Ative o ambiente Conda

Ative o ambiente PyTorch -Neuron conda usando o seguinte comando:

```
source activate aws_neuron_pytorch_p36
```

Para sair do ambiente Conda atual, execute:

```
source deactivate
```

Compilação ResNet50

Crie um script Python chamado **pytorch_trace_resnet50.py** com o conteúdo a seguir. Esse script usa a API Python de compilação PyTorch -Neuron para compilar um modelo -50. ResNet

Note

Há uma dependência entre as versões do torchvision e do pacote torch que você deve conhecer ao compilar os modelos do torchvision. Essas regras de dependência podem ser gerenciadas por meio do pip. Torchvision==0.6.1 corresponde à versão torch==1.5.1, enquanto torchvision==0.8.2 corresponde à versão torch==1.7.1.

```
import torch
import numpy as np
import os
import torch_neuron
from torchvision import models

image = torch.zeros([1, 3, 224, 224], dtype=torch.float32)

## Load a pretrained ResNet50 model
model = models.resnet50(pretrained=True)

## Tell the model we are using it for evaluation (not training)
model.eval()
model_neuron = torch.neuron.trace(model, example_inputs=[image])

## Export to saved model
model_neuron.save("resnet50_neuron.pt")
```

Execute o script de compilação.

```
python pytorch_trace_resnet50.py
```

A compilação demora alguns minutos. Quando terminar, o modelo compilado será salvo como `resnet50_neuron.pt` no diretório local.

ResNet50 Inferência

Crie um script Python chamado **pytorch_infer_resnet50.py** com o conteúdo a seguir. Esse script faz download de uma imagem de amostra e a usa para executar a inferência com o modelo compilado.

```
import os
import time
import torch
import torch_neuron
import json
import numpy as np

from urllib import request

from torchvision import models, transforms, datasets
```

```
## Create an image directory containing a small kitten
os.makedirs("./torch_neuron_test/images", exist_ok=True)
request.urlretrieve("https://raw.githubusercontent.com/aws-labs/mxnet-model-server/
master/docs/images/kitten_small.jpg",
                    "./torch_neuron_test/images/kitten_small.jpg")

## Fetch labels to output the top classifications
request.urlretrieve("https://s3.amazonaws.com/deep-learning-models/image-models/
imagenet_class_index.json", "imagenet_class_index.json")
idx2label = []

with open("imagenet_class_index.json", "r") as read_file:
    class_idx = json.load(read_file)
    idx2label = [class_idx[str(k)][1] for k in range(len(class_idx))]

## Import a sample image and normalize it into a tensor
normalize = transforms.Normalize(
    mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225])

eval_dataset = datasets.ImageFolder(
    os.path.dirname("./torch_neuron_test/"),
    transforms.Compose([
        transforms.Resize([224, 224]),
        transforms.ToTensor(),
        normalize,
    ])
)

image, _ = eval_dataset[0]
image = torch.tensor(image.numpy()[np.newaxis, ...])

## Load model
model_neuron = torch.jit.load( 'resnet50_neuron.pt' )

## Predict
results = model_neuron( image )

# Get the top 5 results
top5_idx = results[0].sort()[1][-5:]

# Lookup and print the top 5 labels
```

```
top5_labels = [idx2label[idx] for idx in top5_idx]

print("Top 5 labels:\n {}".format(top5_labels) )
```

Execute a inferência com o modelo compilado usando o seguinte comando:

```
python pytorch_infer_resnet50.py
```

A saída será semelhante a:

```
Top 5 labels:
['tiger', 'lynx', 'tiger_cat', 'Egyptian_cat', 'tabby']
```

O ARM64 DLAMI

AWS ARM64 As GPUs DLAMIs são projetadas para fornecer alto desempenho e economia para cargas de trabalho de aprendizado profundo. Especificamente, o tipo de instância G5g apresenta o [processador AWS Graviton2](#) baseado em ARM64, que foi desenvolvido do zero AWS e otimizado para a forma como os clientes executam suas cargas de trabalho na nuvem. AWS ARM64 As GPUs DLAMIs são pré-configuradas com Docker, NVIDIA Docker, NVIDIA Driver, CUDA, cuDNN, NCCL, além de estruturas populares de aprendizado de máquina, como e. TensorFlow PyTorch

Com o tipo de instância G5g, você pode aproveitar os benefícios de preço e desempenho do Graviton2 para implantar modelos de aprendizado profundo acelerados por GPU a um custo significativamente menor em comparação com instâncias que usam como base x86 com aceleração de GPU.

Selecione um ARM64 DLAMI

Execute uma [instância G5g](#) com o ARM64 DLAMI de sua escolha.

Para step-by-step obter instruções sobre como iniciar uma DLAMI, [consulte Iniciando e configurando uma DLAMI](#).

Para obter uma lista das mais recentes ARM64 DLAMIs, consulte as [notas de lançamento do DLAMI](#).

Comece agora

Os tópicos a seguir mostram como começar a usar o ARM64 DLAMI.

Conteúdo

- [Usando a ARM64 GPU DLAMI PyTorch](#)

Usando a ARM64 GPU DLAMI PyTorch

O AMIs de deep learning da AWS está pronto para uso com processador Arm64 baseado em processador e vem GPUs otimizado para. PyTorch O ARM64 GPU PyTorch DLAMI inclui um ambiente Python [PyTorch](#) pré-configurado [TorchVision](#) com, [TorchServe](#) e para casos de uso de treinamento e inferência de aprendizado profundo.

Conteúdo

- [Verifique o PyTorch ambiente Python](#)
- [Execute uma amostra de treinamento com PyTorch](#)
- [Execute uma amostra de inferência com PyTorch](#)

Verifique o PyTorch ambiente Python

Conecte-se à sua instância G5g e ative o ambiente básico do Conda com o seguinte comando:

```
source activate base
```

Seu prompt de comando deve indicar que você está trabalhando no ambiente básico do Conda, que contém PyTorch TorchVision, e outras bibliotecas.

```
(base) $
```

Verifique os caminhos de ferramentas padrão do PyTorch ambiente:

```
(base) $ which python
(base) $ which pip
(base) $ which conda
(base) $ which mamba
>>> import torch, torchvision
>>> torch.__version__
>>> torchvision.__version__
>>> v = torch.autograd.Variable(torch.randn(10, 3, 224, 224))
>>> v = torch.autograd.Variable(torch.randn(10, 3, 224, 224)).cuda()
>>> assert isinstance(v, torch.Tensor)
```

Execute uma amostra de treinamento com PyTorch

Execute uma amostra de trabalho de treinamento do MNIST:

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

O resultado deve ser semelhante ao seguinte:

```
...
Train Epoch: 14 [56320/60000 (94%)]    Loss: 0.021424
Train Epoch: 14 [56960/60000 (95%)]    Loss: 0.023695
Train Epoch: 14 [57600/60000 (96%)]    Loss: 0.001973
Train Epoch: 14 [58240/60000 (97%)]    Loss: 0.007121
Train Epoch: 14 [58880/60000 (98%)]    Loss: 0.003717
Train Epoch: 14 [59520/60000 (99%)]    Loss: 0.001729
Test set: Average loss: 0.0275, Accuracy: 9916/10000 (99%)
```

Execute uma amostra de inferência com PyTorch

Use os comandos a seguir para baixar um modelo densenet161 pré-treinado e executar a inferência usando: TorchServe

```
# Set up TorchServe
cd $HOME
git clone https://github.com/pytorch/serve.git
mkdir -p serve/model_store
cd serve

# Download a pre-trained densenet161 model
wget https://download.pytorch.org/models/densenet161-8d451a50.pth >/dev/null

# Save the model using torch-model-archiver
torch-model-archiver --model-name densenet161 \
  --version 1.0 \
  --model-file examples/image_classifier/densenet_161/model.py \
  --serialized-file densenet161-8d451a50.pth \
  --handler image_classifier \
  --extra-files examples/image_classifier/index_to_name.json \
  --export-path model_store

# Start the model server
```

```
torchserve --start --no-config-snapshots \  
  --model-store model_store \  
  --models densenet161=densenet161.mar &> torchserve.log  
  
# Wait for the model server to start  
sleep 30  
  
# Run a prediction request  
curl http://127.0.0.1:8080/predictions/densenet161 -T examples/image_classifier/  
kitten.jpg
```

O resultado deve ser semelhante ao seguinte:

```
{  
  "tiger_cat": 0.4693363308906555,  
  "tabby": 0.4633873701095581,  
  "Egyptian_cat": 0.06456123292446136,  
  "lynx": 0.0012828150065615773,  
  "plastic_bag": 0.00023322898778133094  
}
```

Use os comandos a seguir para cancelar o registro do modelo densenet161 e parar o servidor:

```
curl -X DELETE http://localhost:8081/models/densenet161/1.0  
torchserve --stop
```

O resultado deve ser semelhante ao seguinte:

```
{  
  "status": "Model \"densenet161\" unregistered"  
}  
TorchServe has stopped.
```

Inferência

Nesta seção você encontra tutoriais sobre como executar inferências usando as estruturas e ferramentas da DLAMI.

Ferramentas de inferência

- [TensorFlow Servindo](#)

Fornecimento de modelos

Veja a seguir as opções de fornecimento de modelos instaladas na AMI de deep learning com Conda. Clique em uma das opções para saber como usá-la.

Tópicos

- [TensorFlow Servindo](#)
- [TorchServe](#)

TensorFlow Servindo

TensorFlow O [Serving](#) é um sistema de atendimento flexível e de alto desempenho para modelos de aprendizado de máquina.

tensorflow-serving-api É pré-instalado com DLAMI de estrutura única. Para usar o serviço tensorflow, primeiro ative o TensorFlow ambiente.

```
$ source /opt/tensorflow/bin/activate
```

Em seguida, use um editor de texto de sua preferência para criar um script com o conteúdo a seguir. Chame-o de `test_train_mnist.py`. Esse script é referenciado no [TensorFlow Tutorial](#), que treinará e avaliará um modelo de aprendizado de máquina de rede neural que classifica imagens.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
```

```
metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Agora execute o script passando o local do servidor, a porta e o nome do arquivo da fotografia do husky como parâmetros.

```
$ /opt/tensorflow/bin/python3 test_train_mnist.py
```

Aguarde. Este script pode demorar um pouco antes de fornecer resultados. Quando o treinamento estiver concluído, você verá o seguinte:

```
I0000 00:00:1739482012.389276    4284 device_compiler.h:188] Compiled cluster using
XLA! This line is logged at most once for the lifetime of the process.
1875/1875 [=====] - 24s 2ms/step - loss: 0.2973 - accuracy:
0.9134
Epoch 2/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.1422 - accuracy:
0.9582
Epoch 3/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.1076 - accuracy:
0.9687
Epoch 4/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.0872 - accuracy:
0.9731
Epoch 5/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0731 - accuracy:
0.9771
313/313 [=====] - 0s 1ms/step - loss: 0.0749 - accuracy:
0.9780
```

Outros recursos e exemplos

Se você estiver interessado em saber mais sobre o TensorFlow Serving, confira o [TensorFlow site](#).

TorchServe

TorchServe é uma ferramenta flexível para servir modelos de aprendizado profundo que foram exportados do PyTorch. TorchServe vem pré-instalado com a AMI de aprendizado profundo com Conda.

Para obter mais informações sobre o uso TorchServe, consulte [Model Server for PyTorch Documentation](#).

Tópicos

Ofereça um modelo de classificação de imagens em TorchServe

Este tutorial mostra como servir um modelo de classificação de imagens com TorchServe. Ele usa um modelo DenseNet -161 fornecido pela PyTorch. Quando o servidor está em execução, ele escuta as solicitações de previsão. Quando você carrega uma imagem, neste caso, uma imagem de um gatinho, o servidor retorna uma estimativa das cinco principais classes correspondentes das classes em que o modelo foi treinado.

Para fornecer um exemplo de modelo de classificação de imagens em TorchServe

1. Conecte-se a uma instância do Amazon Elastic Compute Cloud (Amazon EC2) com o Deep Learning AMI com o Conda v34 ou posterior.
2. Ative o ambiente `pytorch_p310`.

```
source activate pytorch_p310
```

3. Clone o TorchServe repositório e crie um diretório para armazenar seus modelos.

```
git clone https://github.com/pytorch/serve.git
mkdir model_store
```

4. Arquive o modelo usando o arquivador de modelos. O `extra-files` parâmetro usa um arquivo do TorchServe repositório, portanto, atualize o caminho, se necessário. Para obter mais informações sobre o arquivador de modelos, consulte [Torch Model archiver](#) for TorchServe.

```
wget https://download.pytorch.org/models/densenet161-8d451a50.pth
torch-model-archiver --model-name densenet161 --version 1.0 --model-file ./
serve/examples/image_classifier/densenet_161/model.py --serialized-file
densenet161-8d451a50.pth --export-path model_store --extra-files ./serve/examples/
image_classifier/index_to_name.json --handler image_classifier
```

5. Execute TorchServe para iniciar um endpoint. A adição de `> /dev/null` silencia a saída do log.

```
torchserve --start --ncs --model-store model_store --models densenet161.mar > /dev/null
```

6. Baixe uma imagem de um gatinho e envie-a para o endpoint de TorchServe previsão:

```
curl -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg  
curl http://127.0.0.1:8080/predictions/densenet161 -T kitten.jpg
```

O endpoint de previsão retorna uma previsão em JSON semelhante às cinco principais previsões a seguir, em que a imagem tem uma probabilidade de 47% de conter um gato egípcio, seguida por uma chance de 46% de ter um gato malhado.

```
{  
  "tiger_cat": 0.46933576464653015,  
  "tabby": 0.463387668132782,  
  "Egyptian_cat": 0.0645613968372345,  
  "lynx": 0.0012828196631744504,  
  "plastic_bag": 0.00023323058849200606  
}
```

7. Ao terminar o teste, interrompa o servidor.

```
torchserve --stop
```

Outros exemplos

TorchServe tem vários exemplos que você pode executar em sua instância DLAMI. Você pode visualizá-los na [página de exemplos TorchServe do repositório do projeto](#).

Mais informações

Para obter mais TorchServe documentação, incluindo como configurar TorchServe o Docker e os TorchServe recursos mais recentes, consulte [a página do TorchServe projeto](#) em GitHub.

Como atualizar a DLAMI

Aqui você encontrará informações sobre como atualizar a DLAMI e dicas sobre como atualizar o software na DLAMI.

Sempre mantenha o sistema operacional e outros softwares instalados atualizados executando patches e atualizações assim que forem disponibilizados.

Se você está usando o Amazon Linux ou Ubuntu, ao efetuar login na DLAMI, você será notificado se houver atualizações disponíveis e verá as instruções para atualização. Para obter mais informações sobre a manutenção do Amazon Linux, consulte [Atualizar o software de instância](#). Para instâncias do Ubuntu, consulte a [Documentação oficial do Ubuntu](#).

No Windows, verifique o Windows Update regularmente para instalar atualizações de software e de segurança. Se você preferir, as atualizações podem ser instaladas automaticamente.

Important

[Para obter informações sobre as vulnerabilidades Meltdown e Spectre e como corrigir seu sistema operacional para resolvê-las, consulte o Boletim de Segurança -2018-013. AWS](#)

Tópicos

- [Atualização para a nova versão da DLAMI](#)
- [Dicas para as atualizações de software](#)
- [Receber notificações sobre novas atualizações](#)

Atualização para a nova versão da DLAMI

As imagens do sistema da DLAMI são atualizadas regularmente para aproveitar as novas versões da estrutura de aprendizado profundo, CUDA e outras atualizações de softwares e ajustes de desempenho. Se estiver usando uma DLAMI há um certo tempo e deseja aproveitar uma atualização, você precisaria executar uma nova instância. Também seria necessário transferir manualmente todos os conjuntos de dados, pontos de verificação ou outros dados importantes. Em vez disso, você pode usar o Amazon EBS para reter seus dados e associá-los a uma nova DLAMI. Dessa forma, você pode fazer atualizações com frequência e, ao mesmo tempo, minimizar o tempo necessário para transferir os dados.

Note

Ao conectar e mover volumes do Amazon EBS entre eles DLAMIs, você deve ter o volume DLAMIs e o novo na mesma zona de disponibilidade.

1. Use a Amazon EC2console para criar um novo volume do Amazon EBS. Para obter instruções detalhadas, consulte [Criação de um volume do Amazon EBS](#).
2. Associe o volume do Amazon EBS recém-criado à sua DLAMI existente. Para obter instruções detalhadas, consulte [Associação de um volume do Amazon EBS](#).
3. Transfira seus dados, como conjuntos de dados, pontos de verificação e arquivos de configuração.
4. Execute uma DLAMI. Para obter instruções detalhadas, consulte [Configurar uma instância de DLAMI](#).
5. Desassocie o volume do Amazon EBS da sua antiga DLAMI. Para obter instruções detalhadas, consulte [Desassociação de um volume do Amazon EBS](#).
6. Associe o volume do Amazon EBS à sua nova DLAMI. Siga as instruções da etapa 2 para associar o volume.
7. Após verificar se seus dados estão disponíveis na nova DLAMI, interrompa e encerre a antiga. Para obter instruções detalhadas para limpeza, consulte [Limpar uma instância de DLAMI](#).

Dicas para as atualizações de software

Periodicamente, você pode atualizar manualmente o software em sua DLAMI. É recomendável usar `pip` para atualizar pacotes Python. Você também deve usar `pip` para atualizar pacotes em um ambiente Conda na AMI de deep learning com Conda. Consulte o site do software ou do framework específico para obter instruções de atualização e instalação.

Note

Não podemos garantir que uma atualização de pacote será bem-sucedida. A tentativa de atualizar um pacote em um ambiente com dependências incompatíveis pode resultar em uma falha. Nesse caso, você deve entrar em contato com o responsável pela biblioteca para conferir se é possível atualizar as dependências do pacote. Como alternativa, é possível tentar modificar o ambiente para permitir a atualização. No entanto, essa modificação

provavelmente significará remover ou atualizar os pacotes existentes, ou seja, não poderemos mais garantir a estabilidade desse ambiente.

AMIs de deep learning da AWS Ele vem com muitos ambientes Conda e muitos pacotes pré-instalados. Devido ao número de pacotes pré-instalados, é difícil encontrar um conjunto de pacotes com garantia de compatibilidade. Você pode ver um aviso “O ambiente é inconsistente, verifique o plano do pacote com cuidado”. A DLAMI garante que todos os ambientes fornecidos por ela estejam corretos, mas não pode garantir que nenhum pacote instalado pelo usuário funcione corretamente.

Receber notificações sobre novas atualizações

Note

AWS O Deep Learning AMIs tem uma cadência semanal de lançamentos de patches de segurança. As notificações de lançamento serão enviadas para esses patches de segurança incrementais, embora eles possam não estar incluídos nas notas oficiais de lançamento.

Você pode receber notificações sempre que uma nova DLAMI for lançada. As notificações são publicadas com o [Amazon SNS](#) usando o tópico a seguir.

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

As mensagens são publicadas aqui quando uma nova DLAMI é publicada. A versão, os metadados e o ID regional da AMI serão incluídos na mensagem.

Essas mensagens podem ser recebidas usando vários métodos diferentes. Recomendamos que você use o método a seguir.

1. Abra o console do [Amazon SNS](#).
2. Na barra de navegação, altere a AWS Região para Oeste dos EUA (Oregon), se necessário. Você deve selecionar a região em que a notificação do SNS que está assinando foi criada.
3. No painel de navegação, escolha Assinaturas, Criar assinatura.
4. Na caixa de diálogo Create subscription, faça o seguinte:
 - a. Para ARN do tópico, copie e cole o seguinte nome do recurso da Amazon (ARN):
arn:aws:sns:us-west-2:767397762724:dlami-updates

- b. Para Protocolo, escolha um entre [Amazon SQS, AWS Lambda, Email, Email-json]
 - c. Para Endpoint, insira o endereço de e-mail ou nome do recurso da Amazon (ARN) do recurso que você usará para receber as notificações.
 - d. Selecione Create subscription.
5. Você receberá um e-mail de confirmação com o assunto Notificação da AWS : confirmação de assinatura. Abra o e-mail e escolha Confirm subscription para concluir a assinatura.

Segurança em AMIs de deep learning da AWS

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como segurança da nuvem e segurança na nuvem:

- **Segurança da nuvem** — AWS é responsável por proteger a infraestrutura que é executada Serviços da AWS no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de [AWS](#) de . Para saber mais sobre os programas de conformidade aplicáveis AMIs de deep learning da AWS, consulte [AWS Serviços no escopo do programa de conformidade AWS](#) .
- **Segurança na nuvem** — Sua responsabilidade é determinada pelo AWS service (Serviço da AWS) que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e normas aplicáveis.

Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar a DLAMI. Os tópicos a seguir mostram como configurar a DLAMI para atender aos seus objetivos de segurança e conformidade. Você também aprende a usar outros Serviços da AWS que o ajudem a monitorar e proteger seus recursos de DLAMI.

Para obter mais informações, consulte [Segurança na Amazon EC2](#) no Guia do EC2 usuário da Amazon.

Tópicos

- [Proteção de dados em AMIs de deep learning da AWS](#)
- [Gerenciamento de identidade e acesso para AMIs de deep learning da AWS](#)
- [Validação de conformidade para AMIs de deep learning da AWS](#)
- [Resiliência em AMIs de deep learning da AWS](#)
- [Segurança da infraestrutura em AMIs de deep learning da AWS](#)
- [AMIs de deep learning da AWS Instâncias de monitoramento](#)

Proteção de dados em AMIs de deep learning da AWS

O modelo de [responsabilidade AWS compartilhada modelo](#) se aplica à proteção de dados em AMIs de deep learning da AWS. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as [Data Privacy FAQ](#). Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog [AWS Shared Responsibility Model and RGPD](#) no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como [trabalhar com CloudTrail trilhas](#) no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para obter mais informações sobre os endpoints FIPS disponíveis, consulte [Federal Information Processing Standard \(FIPS\) 140-3](#).

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sigilosas, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com DLAMI ou Serviços da AWS outro usando o console, a API AWS CLI ou. AWS SDKs Quaisquer dados inseridos em tags ou em campos de texto de formato livre

usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Gerenciamento de identidade e acesso para AMIs de deep learning da AWS

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (fazer login) e autorizado (ter permissões) para usar os recursos da DLAMI. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Para obter mais informações sobre gerenciamento de identidade e acesso, consulte Gerenciamento de [identidade e acesso para a Amazon EC2](#).

Tópicos

- [Autenticação com identidades](#)
- [Gerenciar o acesso usando políticas](#)
- [IAM com o Amazon EMR](#)

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login AWS, consulte [Como fazer login Conta da AWS no](#) Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para designar solicitações por conta própria, consulte [Versão 4 do AWS Signature para solicitações de API](#) no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser necessário fornecer informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte [Autenticação multifator](#) no Guia do usuário do AWS IAM Identity Center e [Usar a autenticação multifator da AWS no IAM](#) no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário-raiz para tarefas diárias. Proteja as credenciais do usuário-raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário-raiz, consulte [Tarefas que exigem credenciais de usuário-raiz](#) no Guia do Usuário do IAM.

Usuários e grupos do IAM

Um [usuário do IAM](#) é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, é recomendável contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, é recomendável alternar as chaves de acesso. Para obter mais informações, consulte [Alternar as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo](#) no Guia do Usuário do IAM.

Um [grupo do IAM](#) é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte [Casos de uso para usuários do IAM](#) no Guia do usuário do IAM.

Perfis do IAM

Uma [função do IAM](#) é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Para assumir temporariamente uma função do IAM no AWS Management Console, você pode [alternar de um usuário para uma função do IAM \(console\)](#). Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para usar perfis, consulte [Métodos para assumir um perfil](#) no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- **Acesso de usuário federado:** para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas por ele. Para ter mais informações sobre perfis para federação, consulte [Criar um perfil para um provedor de identidade de terceiros \(federação\)](#) no Guia do usuário do IAM. Se usar o Centro de Identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de Identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte [Conjuntos de Permissões](#) no Guia do Usuário do AWS IAM Identity Center .
- **Permissões temporárias para usuários do IAM:** um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- **Acesso entre contas:** é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.
- **Acesso entre serviços** — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso

usando as permissões da entidade principal da chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.

- **Sessões de acesso direto (FAS)** — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte [Sessões de acesso direto](#).
- **Perfil de serviço:** um perfil de serviço é um [perfil do IAM](#) que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte [Criar um perfil para delegar permissões a um AWS service \(Serviço da AWS\)](#) no Guia do Usuário do IAM.
- **Função vinculada ao serviço** — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode presumir o perfil para executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados a serviço.
- **Aplicativos em execução na Amazon EC2** — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo solicitações AWS CLI de AWS API. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte [Usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon](#) no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida

ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte [Visão geral das políticas JSON](#) no Guia do usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal

especificado pode executar nesse atributo e em que condições. Você deve [especificar uma entidade principal](#) em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a [visão geral da lista de controle de acesso \(ACL\)](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte [Limites de permissões para identidades do IAM](#) no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations e SCPs, consulte [Políticas de controle de serviços](#) no Guia AWS Organizations do Usuário.

- Políticas de controle de recursos (RCPs) — RCPs são políticas JSON que você pode usar para definir o máximo de permissões disponíveis para recursos em suas contas sem atualizar as políticas do IAM anexadas a cada recurso que você possui. O RCP limita as permissões para recursos nas contas dos membros e pode afetar as permissões efetivas para identidades, incluindo a Usuário raiz da conta da AWS, independentemente de pertencerem à sua organização. Para obter mais informações sobre Organizations e RCPs, incluindo uma lista Serviços da AWS desse suporte RCPs, consulte [Políticas de controle de recursos \(RCPs\)](#) no Guia AWS Organizations do usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte [Lógica de avaliação de políticas](#) no Guia do usuário do IAM.

IAM com o Amazon EMR

Você pode usar o IAM com o Amazon EMR para definir usuários, AWS recursos, grupos, funções e políticas. Você também pode controlar Serviços da AWS quais usuários e funções podem acessar.

Para ter mais informações sobre como usar o IAM com o Amazon EMR, consulte [AWS Identity and Access Management para Amazon EMR](#).

Validação de conformidade para AMIs de deep learning da AWS

Audidores terceirizados avaliam a segurança e a conformidade AMIs de deep learning da AWS como parte de vários programas de AWS conformidade. Para obter informações sobre os programas de conformidade suportados, consulte [Validação de conformidade para a Amazon EC2](#).

Para obter uma lista dos Serviços da AWS escopo de programas de conformidade específicos, consulte [AWS Serviços no escopo do programa de conformidade AWS](#) . Para obter informações gerais, consulte Programas de [AWS conformidade Programas AWS](#) de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para ter mais informações, consulte [Downloading Reports in AWS Artifact](#).

Sua responsabilidade de conformidade ao usar o DLAMI é determinada pela sensibilidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentos aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- [Guias de início rápido de segurança e compatibilidade](#): esses guias de implantação abordam as considerações de arquitetura e fornecem etapas para implantação de ambientes de linha de base focados em compatibilidade e segurança na AWS.
- AWS Recursos de <https://aws.amazon.com/compliance/resources/> de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- [Avaliação de recursos com AWS Config regras](#) no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- [AWS Security Hub](#)— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar seus AWS recursos e verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.

Resiliência em AMIs de deep learning da AWS

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade. Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte [Infraestrutura AWS global](#).

Para obter informações sobre os EC2 recursos da Amazon para ajudar a suportar suas necessidades de resiliência e backup de dados, consulte [Resiliência EC2 na Amazon no Guia EC2](#) do usuário da Amazon.

Segurança da infraestrutura em AMIs de deep learning da AWS

A segurança da infraestrutura do AMIs de deep learning da AWS é apoiada pela Amazon EC2. Para obter mais informações, consulte [Segurança da infraestrutura na Amazon EC2](#) no Guia EC2 do usuário da Amazon.

AMIs de deep learning da AWS Instâncias de monitoramento

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho da sua AMIs de deep learning da AWS instância e de suas outras AWS soluções. Sua instância DLAMI vem com várias ferramentas de monitoramento de GPU, incluindo um utilitário que reporta estatísticas de uso da GPU para a Amazon. CloudWatch Para obter mais informações [Monitoramento e otimização de GPU](#), consulte e consulte [Monitorar EC2 recursos da Amazon](#) no Guia EC2 do usuário da Amazon.

Optar por não rastrear o uso de instâncias de DLAMI

As distribuições de sistema AMIs de deep learning da AWS operacional a seguir incluem código que permite AWS coletar o tipo de instância, ID da instância, tipo de DLAMI e informações do sistema operacional.

Note

AWS não coleta nem retém nenhuma outra informação sobre a DLAMI, como os comandos que você usa na DLAMI.

- Amazon Linux 2
- Amazon Linux 2023
- Ubuntu 20.04
- Ubuntu 22.04

Como desativar o rastreamento de uso

Se preferir, você pode desativar o rastreamento de uso de uma nova instância de DLAMI. Para optar por não participar, você deve adicionar uma tag à sua EC2 instância da Amazon durante o lançamento. A tag deve usar a chave `OPT_OUT_TRACKING` com o valor associado definido como `true`. Para obter mais informações, consulte [Marcar seus EC2 recursos da Amazon](#) no Guia EC2 do usuário da Amazon.

Política de suporte da DLAMI

Aqui você pode encontrar detalhes da política de suporte para AMIs de deep learning da AWS (DLAMI).

[Para obter uma lista das estruturas e do sistema operacional DLAMI AWS que atualmente oferecem suporte, consulte a página da Política de Suporte da DLAMI.](#) A terminologia a seguir se aplica a todos os itens DLAMIs mencionados na página da política de Support e nesta página:

- A Versão atual especifica a versão do framework no formato x.y.z. Nesse formato, x se refere à versão principal, y se refere à versão secundária e z se refere à versão do patch. Por exemplo, para TensorFlow 2.10.1, a versão principal é 2, a versão secundária é 10 e a versão do patch é 1.
- O fim do patch especifica por quanto tempo AWS oferece suporte a uma estrutura específica ou versão do sistema operacional.

Para obter informações detalhadas sobre questões específicas DLAMIs, consulte [Notas de AMIs lançamento do Deep Learning](#).

Suporte do DLAMI FAQs

- [Quais versões da estrutura recebem patches de segurança?](#)
- [Qual sistema operacional recebe patches de segurança?](#)
- [Quais imagens são AWS publicadas quando novas versões do framework são lançadas?](#)
- [Quais imagens recebem novos AWS recursos de SageMaker IA/I?](#)
- [Como a versão atual é definida na tabela de Estruturas compatíveis?](#)
- [E se eu estiver executando uma versão que não está na tabela suportada?](#)
- [Oferecem DLAMIs suporte às versões anteriores de patch de uma versão do Framework?](#)
- [Como posso encontrar a imagem com patch aplicado mais recente de uma versão compatível da estrutura?](#)
- [Com que frequência novas imagens são lançadas?](#)
- [Minha instância receberá um patch enquanto a workload estiver em execução?](#)
- [O que acontece quando uma nova versão com patch aplicado ou atualizada da estrutura está disponível?](#)
- [As dependências são atualizadas sem alterar a versão da estrutura?](#)

- [Quando o suporte ativo para minha versão da estrutura termina?](#)
- [As imagens com versões de estrutura que não são mais mantidas ativamente receberão um patch?](#)
- [Como usar uma versão de estrutura mais antiga?](#)
- [Como faço para up-to-date acompanhar as mudanças de suporte nas estruturas e suas versões?](#)
- [Preciso de uma licença comercial para usar o repositório Anaconda?](#)

Quais versões da estrutura recebem patches de segurança?

Se a versão da estrutura estiver em Supported Framework Versions na [tabela AMIs de deep learning da AWS Support Policy](#), ela receberá patches de segurança.

Qual sistema operacional recebe patches de segurança?

Se o sistema operacional estiver listado em Versões de sistema operacional suportadas na [tabela AMIs de deep learning da AWS Support Policy](#), ele receberá patches de segurança.

Quais imagens são AWS publicadas quando novas versões do framework são lançadas?

Publicamos novas versões DLAMIs logo após o lançamento TensorFlow e o lançamento PyTorch de novas versões. Isso inclui versões principais, versões principais e secundárias e major-minor-patch versões de estruturas. Também atualizamos as imagens quando novas versões de drivers e bibliotecas são disponibilizadas. Para obter mais informações sobre a manutenção da imagem, consulte [Quando o suporte ativo para minha versão da estrutura termina?](#)

Quais imagens recebem novos AWS recursos de SageMaker IA/I?

Os novos recursos geralmente são lançados na versão mais recente do DLAMIs for PyTorch TensorFlow e. Consulte as notas de lançamento para obter uma imagem específica para obter detalhes sobre a nova SageMaker IA ou os novos AWS recursos. Para obter uma lista dos disponíveis DLAMIs, consulte as [notas de lançamento do DLAMI](#). Para obter mais informações sobre a manutenção da imagem, consulte [Quando o suporte ativo para minha versão da estrutura termina?](#)

Como a versão atual é definida na tabela de Estruturas compatíveis?

A versão atual na [tabela AMIs de deep learning da AWS Support Policy](#) se refere à versão mais recente da estrutura AWS disponibilizada em GitHub. Cada versão mais recente inclui atualizações

de drivers, bibliotecas e pacotes relevantes na DLAMI. Para obter informações sobre a manutenção de imagens, consulte [Quando o suporte ativo para minha versão da estrutura termina?](#)

E se eu estiver executando uma versão que não está na tabela suportada?

Se você estiver executando uma versão que não está na [tabela AMIs de deep learning da AWS Support Policy](#), talvez você não tenha os drivers, as bibliotecas e os pacotes relevantes mais atualizados. Para uma up-to-date versão adicional, recomendamos que você atualize para uma das estruturas ou sistemas operacionais compatíveis disponíveis usando o DLAMI mais recente de sua escolha. Para obter uma lista dos disponíveis DLAMIs, consulte as [notas de lançamento do DLAMI](#).

Oferecem DLAMIs suporte às versões anteriores de patch de uma versão do Framework?

Não. Oferecemos suporte à versão de patch mais recente da versão principal mais recente de cada estrutura, lançada 365 dias após o GitHub lançamento inicial, conforme declarado na [tabela de políticas de AMIs de deep learning da AWS suporte](#). Para obter mais informações, consulte [E se eu estiver executando uma versão que não está na tabela suportada?](#).

Como posso encontrar a imagem com patch aplicado mais recente de uma versão compatível da estrutura?

[Para usar um DLAMI com a versão mais recente da estrutura, você pode usar parâmetros AWS CLI ou SSM para recuperar o ID do DLAMI e usá-lo para iniciar o DLAMI usando o console. EC2](#)
[Para exemplos de comandos de parâmetros AWS CLI ou SSM para recuperar o AMIs de deep learning da AWS ID, consulte a página de notas de lançamento do DLAMI, notas de lançamento do DLAMI de estrutura única.](#) A versão da estrutura que você escolher deve estar listada em Supported Framework Versions na [tabela AMIs de deep learning da AWS Support Policy](#).

Com que frequência novas imagens são lançadas?

Nossa maior prioridade é fornecer versões de patch atualizadas. Criamos rotineiramente imagens com patch aplicado na primeira oportunidade. Monitoramos as versões recém-corrigidas da estrutura (ex. TensorFlow 2.9 a TensorFlow 2.9.1) e novas versões secundárias (ex. TensorFlow 2.9 a TensorFlow 2.10) e disponibilize-os o mais rápido possível. Quando uma versão existente do TensorFlow é lançada com uma nova versão do CUDA, lançamos um novo DLAMI para essa versão TensorFlow do com suporte para a nova versão do CUDA.

Minha instância receberá um patch enquanto a workload estiver em execução?

Não. As atualizações de patch para a DLAMI não são atualizações “locais”.

Você deve ativar uma nova EC2 instância, migrar suas cargas de trabalho e scripts e, em seguida, desativar sua instância anterior.

O que acontece quando uma nova versão com patch aplicado ou atualizada da estrutura está disponível?

[Para ser notificado sobre mudanças no DLAMI, assine as notificações do DLAMI relevante, consulte Receber notificações sobre novas atualizações.](#)

As dependências são atualizadas sem alterar a versão da estrutura?

Atualizamos as dependências sem alterar a versão da estrutura. No entanto, se uma atualização de dependência causar uma incompatibilidade, criaremos uma imagem com uma versão diferente. Verifique as [notas de versão da DLAMI para obter](#) informações atualizadas sobre as dependências.

Quando o suporte ativo para minha versão da estrutura termina?

As imagens da DLAMI são imutáveis. Depois de criadas, elas não mudam. Há quatro motivos principais para o fim do suporte ativo de uma versão da estrutura:

- [Atualizações da versão da estrutura \(patch\)](#)
- [AWS patches de segurança](#)
- [Data de fim do patch \(descontinuação\)](#)
- [Dependência end-of-support](#)

Note

Devido à frequência de atualizações de patches de versão e de segurança, é recomendado verificar a página das notas de versão da DLAMI com frequência e atualizá-la quando houver alterações.

Atualizações da versão da estrutura (patch)

Se você tem uma carga de trabalho de DLAMI TensorFlow baseada em 2.7.0 TensorFlow e libera a versão 2.7.1, então libera uma nova DLAMI GitHub com 2.7.1. AWS TensorFlow As imagens anteriores com 2.7.0 não são mais mantidas ativamente depois que a nova imagem com TensorFlow 2.7.1 é lançada. O DLAMI TensorFlow com 2.7.0 não recebe mais patches. A página de notas de lançamento do DLAMI TensorFlow para 2.7 é então atualizada com as informações mais recentes. Não há uma página individual das notas de versão para cada patch secundário.

Os novos DLAMIs criados devido a atualizações de patches são designados com um novo [ID de AMI](#).

AWS patches de segurança

Se você tiver uma carga de trabalho baseada em uma imagem com TensorFlow 2.7.0 e AWS fizer um patch de segurança, uma nova versão do DLAMI será lançada para a 2.7.0. TensorFlow A versão anterior das imagens com TensorFlow 2.7.0 não é mais mantida ativamente. Para obter mais informações, consulte [Minha instância receberá um patch enquanto a workload estiver em execução?](#) Para ver as etapas e encontrar a DLAMI mais recente, consulte [Como posso encontrar a imagem com patch aplicado mais recente de uma versão compatível da estrutura?](#)

Os novos DLAMIs criados devido a atualizações de patches são designados com um novo [ID de AMI](#).

Data de fim do patch (descontinuação)

DLAMIs atingiram a data final do patch 365 dias após a data GitHub de lançamento.

Para [várias estruturas DLAMIs](#), quando uma das versões da estrutura é atualizada, é necessária uma nova DLAMI com a versão atualizada. A DLAMI com a versão antiga da estrutura não é mais mantida ativamente.

Important

A exceção é quando há uma grande atualização da estrutura. Por exemplo, se a versão TensorFlow 1.15 for atualizada para TensorFlow 2.0, continuaremos a oferecer suporte à versão mais recente da TensorFlow 1.15 por um período de dois anos a partir da data de GitHub lançamento ou seis meses após a equipe de manutenção da estrutura de origem cancelar o suporte, qualquer que seja a data anterior.

Dependência end-of-support

Se você estiver executando uma carga de trabalho em uma imagem DLAMI TensorFlow 2.7.0 com o Python 3.6 e essa versão do Python estiver marcada como sim, todas as imagens DLAMI baseadas no Python 3.6 não end-of-support serão mais mantidas ativamente. Da mesma forma, se uma versão do sistema operacional como o Ubuntu 16.04 estiver marcada para end-of-support, todas as imagens DLAMI que dependem do Ubuntu 16.04 não serão mais mantidas ativamente.

As imagens com versões de estrutura que não são mais mantidas ativamente receberão um patch?

Não. As imagens que não são mais mantidas ativamente não terão novos lançamentos.

Como usar uma versão de estrutura mais antiga?

[Para usar uma DLAMI com uma versão mais antiga da estrutura, recupere a ID da DLAMI e use-a para iniciar a DLAMI usando o console. EC2](#) Para obter comandos da AWS CLI para recuperar o ID da AMI, consulte a página de notas de versão nas notas de lançamento da [DLAMI](#) de estrutura única.

Como faço para up-to-date acompanhar as mudanças de suporte nas estruturas e suas versões?

[Fique up-to-date com as estruturas e versões do DLAMI usando a tabela de políticas de suporte do AMIs de deep learning da AWS Framework e as notas de lançamento do DLAMI.](#)

Preciso de uma licença comercial para usar o repositório Anaconda?

O Anaconda mudou para um modelo de licenciamento comercial com foco em determinados usuários. DLAMIs As mantidas ativamente foram migradas para a versão de código aberto disponível ao público do Conda ([conda-forge](#)) do canal Anaconda.

Tabela de políticas de suporte do DLAMI

Para obter mais detalhes, consulte a [Política de Suporte](#).

Versões de estrutura suportadas

Framework	Versão atual	Versão CUDA	GitHub GA	Fim do patch
PyTorch	2.7.0	12,8	2025-04-23	2026-04-23
PyTorch	2.6.0	12.6	2025-01-29	2026-01-29
PyTorch	2.5.1	12.4	2024-11-24	2025-11-24
PyTorch	2.4.1	12.4	2024-07-24	2025-07-24
TensorFlow	2.18.0	12,5	2024-10-24	2025-10-24
TensorFlow	2.17.0	12.3	2024-11-07	2025-11-07

Versões do sistema operacional suportadas

Sistema operacional	Fim do patch
Amazon Linux 2023	2029-06-30
Amazon Linux 2	2026-06-30
Ubuntu 24.04	2029-04-30
Ubuntu 22.04	2027-04-30

Versões de estrutura não suportadas

As versões listadas nesta tabela aparecerão por dois anos após a data de suporte.

Framework	Versão atual	Versão CUDA	GitHub GA	Fim do patch
PyTorch	2.3.0	12.1	2024-04-24	2025-04-24
PyTorch	2.2.0	12.1	2024-01-30	2025-01-30
PyTorch	1.13.1	11.7	28/10/2022	2024-10-28
PyTorch	2.1.0	12.1	2023-10-04	2024-10-04
PyTorch	2.0.0	12.1	2023-03-15	2024-03-15
PyTorch	1.12.1	11.6	2022-07-01	2023-07-01
PyTorch	1.11.0	11.5	2022-03-10	2023-03-10
TensorFlow	2.16.0	12.3	2024-03-07	2025-03-07
TensorFlow	2.15.0	12.2	2023-11-14	2024-11-14
TensorFlow	2.13.0	11.8	2023-07-19	2024-07-19
TensorFlow	2.12.0	11.8	2023-03-23	2024-03-23
TensorFlow	2.11.0	11.2	2022-11-18	2023-11-18
TensorFlow	2.10.1	11.2	2022-09-06	2023-09-06
TensorFlow	2.9.3	11.2	2022-05-17	2023-05-17

Versões de sistema operacional não suportadas

Sistema operacional	Fim do patch
Ubuntu 20.04	2025-05-31
Ubuntu 18.04	2023-05-31

Arquivo de notas de lançamento do DLAMI não suportado

Base

GPU

- [AWS AMI de base de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI de base de aprendizado profundo \(Ubuntu 18.04\)](#)

Estrutura única

PyTorch AMI específica

GPU

- [AWS GPU AMI PyTorch 2.3 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS GPU AMI PyTorch 2.3 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU ARM64 AMI PyTorch 2.3 de aprendizado profundo \(Ubuntu 22.04\)](#)
- [AWS GPU AMI PyTorch 2.2 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo PyTorch 2.2 \(Ubuntu 20.04\)](#)
- [AWS AMI GPU PyTorch 2.1 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI GPU PyTorch 2.0 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS AMI GPU PyTorch 2.0 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS GPU AMI PyTorch 1.13 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo PyTorch 1.13 \(Ubuntu 20.04\)](#)
- [AWS GPU AMI PyTorch 1.12 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo PyTorch 1.12 \(Ubuntu 20.04\)](#)
- [AWS GPU AMI PyTorch 1.11 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI PyTorch 1.11 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS GPU AMI PyTorch 1.10 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI PyTorch 1.10 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI Graviton GPU PyTorch 1.10 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS GPU AMI de aprendizado profundo PyTorch 1.10 \(Ubuntu 18.04\)](#)

- [AWS AMI GPU PyTorch 1.9 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS AMI GPU PyTorch 1.9 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI GPU PyTorch 1.9 de aprendizado profundo \(Ubuntu 18.04\)](#)

TensorFlow AMI específica

GPU

- [AWS GPU AMI TensorFlow 2.16 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.16 \(Ubuntu 20.04\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.15 \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.15 \(Ubuntu 20.04\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.13 \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.13 \(Ubuntu 20.04\)](#)
- [AWS GPU AMI TensorFlow 2.12 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.12 \(Ubuntu 20.04\)](#)
- [AWS GPU AMI TensorFlow 2.11 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.11 \(Ubuntu 20.04\)](#)
- [AWS GPU AMI TensorFlow 2.10 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI TensorFlow 2.10 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS GPU AMI TensorFlow 2.9 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI TensorFlow 2.9 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS GPU AMI TensorFlow 2.8 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS AMI GPU TensorFlow 2.8 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI GPU TensorFlow 2.7 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS AMI GPU TensorFlow 2.7 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI GPU TensorFlow 2.6 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS AMI GPU TensorFlow 2.6 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI Graviton GPU TensorFlow 2.6 de aprendizado profundo \(Ubuntu 20.04\)](#)
- [AWS AMI GPU TensorFlow 2.6 de aprendizado profundo \(Ubuntu 18.04\)](#)
- [AWS GPU AMI TensorFlow 2.5 de aprendizado profundo \(Amazon Linux 2\)](#)
- [AWS GPU AMI de aprendizado profundo TensorFlow 2.5 \(Ubuntu 20.04\)](#)

Estrutura múltipla

GPU

- [AWS AMI de aprendizado profundo \(Ubuntu 18.04\)](#)

Alterações importantes do driver NVIDIA para DLAMIs

Em 15 de novembro de 2023, AWS fez mudanças importantes no AMIs de deep learning da AWS (DLAMI) relacionadas ao driver NVIDIA que uso. DLAMIs Para obter informações sobre o que mudou e se isso afeta seu uso de DLAMIs, consulte [Alteração do driver DLAMI NVIDIA FAQs](#).

Alteração do driver DLAMI NVIDIA FAQs

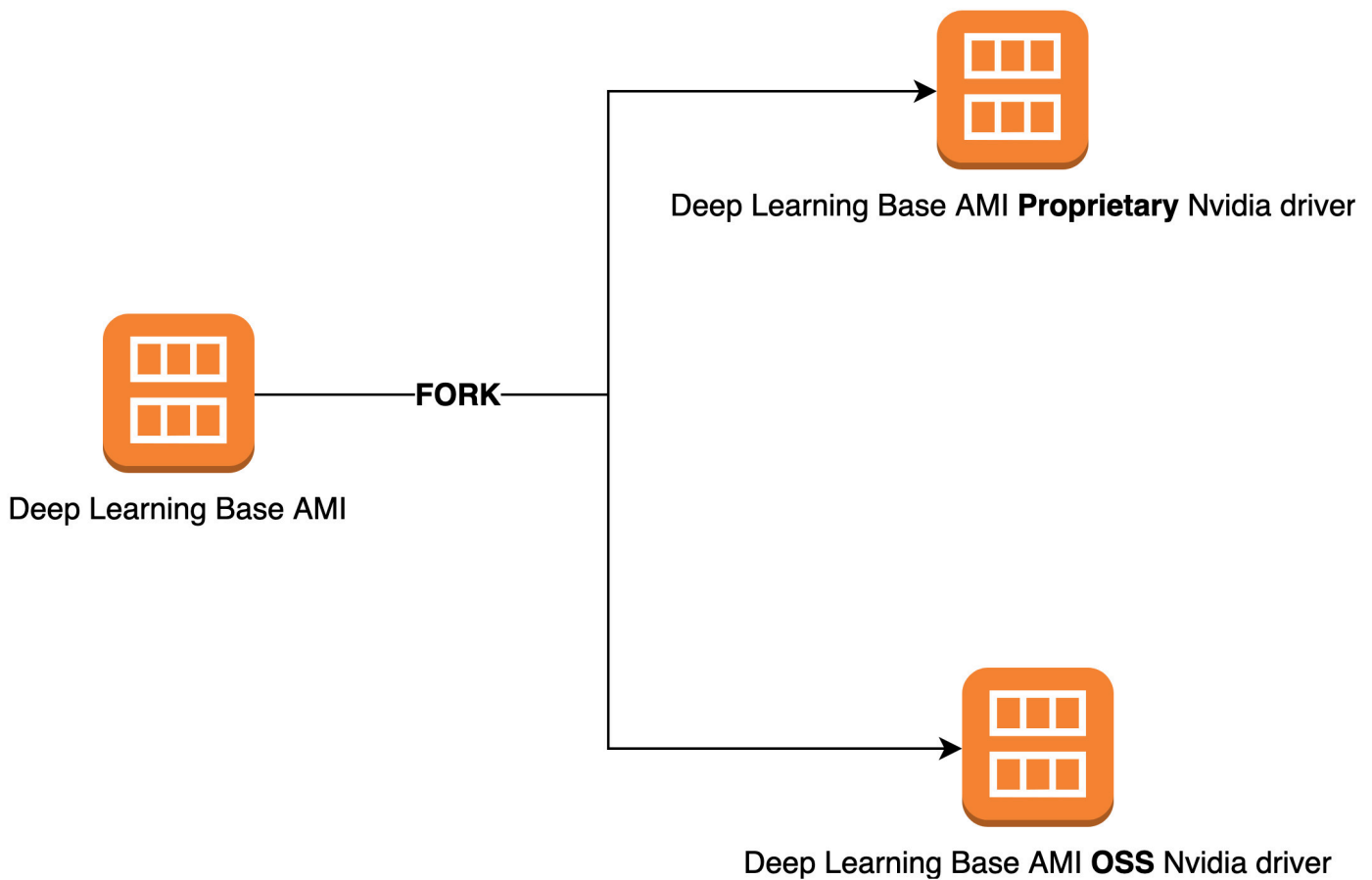
- [O que mudou?](#)
- [Por que essa alteração foi necessária?](#)
- [O DLAMIs que essa mudança afetou?](#)
- [O que isso significa para você?](#)
- [Há alguma perda de funcionalidade com o mais novo DLAMIs?](#)
- [Essa mudança afetou os contêineres de aprendizado profundo?](#)

O que mudou?

Nós nos DLAMIs dividimos em dois grupos separados:

- DLAMIs que usam o driver proprietário da NVIDIA (para suportar P3, P3dn, G3)
- DLAMIs que usam o driver NVIDIA OSS (para suportar G4dn, G5, P4, P5)

Como resultado, criamos novas DLAMIs para cada uma das duas categorias com novos nomes e uma nova AMI IDs. Eles não DLAMIs são intercambiáveis. Ou seja, DLAMIs de um grupo não oferecem suporte a instâncias que o outro grupo suporta. Por exemplo, a DLAMI compatível com P5 não é compatível com G3, enquanto a DLAMI compatível com G3 não é compatível com P5.



Por que essa alteração foi necessária?

Anteriormente, DLAMIs para a NVIDIA GPUs incluía um driver de kernel proprietário da NVIDIA. No entanto, a comunidade do kernel Linux upstream aceitou uma alteração que isola os drivers de kernel proprietários, como o driver da GPU NVIDIA, da comunicação com outros drivers de kernel. Essa alteração desativa o GPUDirect RDMA nas instâncias das séries P4 e P5, que é o mecanismo que permite usar eficientemente o EFA GPUs para treinamento distribuído. Como resultado, DLAMIs agora use o driver OpenRM (driver de código aberto NVIDIA), vinculado aos drivers EFA de código aberto para suportar G4dn, G5, P4 e P5. No entanto, esse driver OpenRM não oferece suporte a instâncias mais antigas (como P3 e G3). Portanto, para garantir que continuemos fornecendo suporte atual, seguro DLAMIs e de alto desempenho para os dois tipos de instância, DLAMIs dividimos em dois grupos: um com o driver OpenRM (compatível com G4dn, G5, P4 e P5) e outro com o driver proprietário mais antigo (compatível com P3, P3dn e G3).

O DLAMIs que essa mudança afetou?

Essa mudança afetou a todos DLAMIs.

O que isso significa para você?

Todos DLAMIs continuarão fornecendo funcionalidade, desempenho e segurança, desde que você os execute em um tipo de instância compatível do Amazon Elastic Compute Cloud (Amazon EC2). Para determinar os tipos de EC2 instância compatíveis com uma DLAMI, verifique as notas de versão dessa DLAMI e, em seguida, procure as Instâncias suportadas. EC2 Para conferir uma lista das opções de DLAMI atualmente compatíveis e links para as respectivas notas de lançamento, consulte [Notas de AMIs lançamento do Deep Learning](#).

Além disso, você deve usar os comandos corretos AWS Command Line Interface (AWS CLI) para invocar o atual DLAMIs.

Para bases DLAMIs que suportam P3, P3dn e G3, use este comando:

```
aws ec2 describe-images --region us-east-1 --owners amazon \
--filters 'Name=name,Values=Deep Learning Base Proprietary Nvidia Driver AMI (Amazon Linux 2) Version ??.' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' --output text
```

Para bases DLAMIs que suportam G4dn, G5, P4 e P5, use este comando:

```
aws ec2 describe-images --region us-east-1 --owners amazon \
--filters 'Name=name,Values=Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version ??.' 'Name=state,Values=available' \
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' --output text
```

Há alguma perda de funcionalidade com o mais novo DLAMIs?

Não, não há nenhuma perda de funcionalidade. As atuais DLAMIs fornecem todas as funcionalidades, desempenho e segurança das anteriores DLAMIs, desde que você as execute em um tipo de EC2 instância compatível.

Essa mudança afetou os contêineres de aprendizado profundo?

Não, essa alteração não afetou os AWS Deep Learning Containers, porque eles não incluem o driver NVIDIA. No entanto, certifique-se de executar Deep Learning Containers AMIs que sejam compatíveis com as instâncias subjacentes.

Informações relacionadas sobre a DLAMI

Você pode encontrar outros recursos com informações relacionadas sobre a DLAMI fora do Guia do desenvolvedor das AMIs de deep learning da AWS . Em AWS re:Post, confira as perguntas de outros clientes sobre o DLAMI ou faça suas próprias perguntas. No Blog do AWS Machine Learning e em outros AWS blogs, leia as postagens oficiais sobre o DLAMI.

AWS re:Post

[Etiqueta: AMIs de deep learning da AWS](#)

AWS Blog

- [AWS Blog de Machine Learning | Categoria: AMIs de deep learning da AWS](#)
- [AWS Blog de Machine Learning | Treinamento mais rápido com TensorFlow 1.6 otimizado nas instâncias EC2 C5 e P3 da Amazon](#)
- [AWS Blog de Machine Learning | Novo AMIs de deep learning da AWS para profissionais de Machine Learning](#)
- [AWS Partner Network \(APN\) Blog | Novos cursos de treinamento disponíveis: Introdução ao Machine Learning e ao Deep Learning em AWS](#)
- [AWS Blog de notícias | Viagem ao aprendizado profundo com AWS](#)

Recursos descontinuados da DLAMI

A tabela a seguir lista os recursos obsoletos do (AMIs de deep learning da AWS DLAMI), a data em que os descontinuamos e os detalhes sobre por que os descontinuamos.

Atributo	Data	Detalhes
Ubuntu 16.04	10/07/2021	O Ubuntu Linux 16.04 LTS chegou ao fim da janela de LTS de cinco anos em 30 de abril de 2021 e não o fornecedor não fornece mais suporte. Não há mais atualizações na AMI de deep learning base (Ubuntu 16.04) em novas versões a partir de outubro de 2021. As versões anteriores continuarão disponíveis.
Amazon Linux	10/07/2021	O Amazon Linux está end-of-life em dezembro de 2020. Não há mais atualizações para novas versões da AMI de deep learning (Amazon Linux) a partir de outubro de 2021. As versões anteriores da AMI de deep learning (Amazon Linux) continuarão disponíveis.
Chainer	01/07/2020	A Chainer anunciou o fim dos grandes lançamentos a partir de dezembro de 2019. Consequentemente, deixaremos

Atributo	Data	Detalhes
		s de incluir ambientes Chainer Conda na DLAMI a partir de julho de 2020. As versões anteriores da DLAMI que contêm esses ambientes continuarão disponíveis. Nós só forneceremos atualizações para esses ambientes se houver correções de segurança publicadas pela comunidade de código aberto para essas estruturas de trabalho.
Python 3.6	15/06/2020	Devido às solicitações dos clientes, estamos migrando para o Python 3.7 para novos lançamentos. TF/MX/PT
Python 2	01/01/2020	A comunidade de código aberto Python encerrou oficialmente o suporte a Python 2. As MXNet comunidades TensorFlow, PyTorch, e também anunciaram que as versões TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4 e MXNet 1.6.0 serão as últimas a oferecer suporte ao Python 2.

Histórico de documentos de DLAMI

A tabela a seguir fornece um histórico dos lançamentos recentes de DLAMI e das alterações relacionadas no Guia do desenvolvedor das AMIs de deep learning da AWS .

Alterações recentes

Alteração	Descrição	Data
Usando o TensorFlow Serving para treinar um modelo MNIST	Um exemplo de uso do Tensorflow servindo para treinar o modelo MNIST.	14 de fevereiro de 2025
ARM64 DLAMI	O AMIs de deep learning da AWS agora suporta imagens baseadas no processador GPUs Arm64.	29 de novembro de 2021
TensorFlow 2	A AMI de aprendizado profundo com Conda agora vem com TensorFlow 2 com CUDA 10.	3 de dezembro de 2019
AWS Inferência	A AMI de aprendizado profundo agora oferece suporte ao hardware AWS Inferentia e ao SDK AWS Neuron.	3 de dezembro de 2019
Instalando PyTorch a partir de uma compilação noturna	Foi adicionado um tutorial que aborda como você pode desinstalar e PyTorch, em seguida, instalar uma versão noturna da sua AMI PyTorch de aprendizado profundo com o Conda.	25 de setembro de 2018

[Tutorial do Conda](#)

O exemplo MOTD foi atualizado para refletir uma versão mais recente.

23 de julho de 2018

Alterações anteriores

A tabela a seguir fornece um histórico de lançamentos de DLAMI e alterações relacionadas anteriores a julho de 2018.

Alteração	Descrição	Data
TensorFlow com Horovod	Foi adicionado um tutorial para treinar ImageNet com TensorFlow e Horovod.	6 de junho de 2018
Guia de atualização	Adicionado o guia de atualização.	15 de maio de 2018
Novas regiões e novo tutorial de 10 minutos	Novas regiões adicionadas: Oeste dos EUA (Norte da Califórnia), América do Sul, Canadá (Central), UE (Londres) e UE (Paris). Além disso, o primeiro lançamento de um tutorial de 10 minutos chamado: "Conceitos básicos da Deep Learning AMI".	26 de abril de 2018
Tutorial do Chainer	Um tutorial para usar o Chainer nos modos várias GPUs, uma única GPU e CPU foi adicionado. A integração do CUDA foi atualizada do CUDA 8 para o CUDA 9 para várias estruturas.	28 de fevereiro de 2018

Alteração	Descrição	Data
Linux AMIs v3.0, além da introdução do MXNet Model Server, TensorFlow Serving e TensorBoard	Tutoriais adicionados para o Conda AMIs com novos recursos de exibição de modelos e visualizações usando o MXNet Model Server v0.1.5, TensorFlow Serving v1.4.0 e v0.4.0. TensorBoard Funcionalidades de AMI e CUDA de estrutura descritas nas visões gerais de Conda e CUDA. Notas de versão mais recentes movidas para https://aws.amazon.com/releasenotes/	25 de janeiro de 2018
Linux AMIs v2.0	Base, Source e Conda AMIs atualizados com o NCCL 2.1. Source e Conda AMIs atualizados com MXNet v1.0, PyTorch 0.3.0 e Keras 2.0.9.	11 de dezembro de 2017
Duas opções de AMI do Windows adicionadas	AMIs Lançamentos do Windows 2012 R2 e 2016: adicionados ao guia de seleção da AMI e adicionados às notas de versão.	30 de novembro de 2017
Versão da documentação inicial	Descrição detalhada da alteração com o link para o tópico ou a seção que sofreu a alteração.	15 de novembro de 2017

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.