aws

Guide du développeur

AWS Apprentissage profond (deep learning) AMIs

AWS Apprentissage profond (deep learning) AMIs: Guide du développeur

Table of Contents

Qu'est-ce que c'est AWS Apprentissage profond (deep learning) AMIs ?

AWS Apprentissage profond (deep learning) AMIs (DLAMI) fournit des images de machine personnalisées que vous pouvez utiliser pour le deep learning dans le cloud. Elles DLAMIs sont disponibles dans la plupart des cas Régions AWS pour différents types d'instances Amazon Elastic Compute Cloud (Amazon EC2), qu'il s'agisse d'une petite instance utilisant uniquement un processeur ou des instances multi-GPU très puissantes les plus récentes. Ils DLAMIs sont préconfigurés avec NVIDIA CUDA et NVIDIA cuDNN ainsi que les dernières versions des frameworks d'apprentissage profond les plus populaires.

A propos de ce manuel

Le contenu du peut vous aider à lancer et à utiliser le DLAMIs. Le guide couvre plusieurs cas d'utilisation courants du deep learning, à la fois pour la formation et pour l'inférence. Il explique également comment choisir l'AMI adaptée à vos besoins et le type d'instances que vous pourriez préférer.

En outre, ils DLAMIs incluent plusieurs didacticiels fournis par leurs frameworks pris en charge. Ce guide peut vous montrer comment activer chaque framework et trouver les didacticiels appropriés pour commencer. Il propose également des didacticiels sur la formation distribuée, le débogage, l'utilisation d' AWS Inferentia et de AWS Trainium, ainsi que sur d'autres concepts clés. Pour savoir comment configurer un serveur de bloc-notes Jupyter pour exécuter les didacticiels dans votre navigateur, consultez. Configuration d'un serveur Jupyter Notebook sur une instance DLAMI

Prérequis

Pour exécuter correctement le DLAMIs, nous vous recommandons de vous familiariser avec les outils de ligne de commande et les bases de Python.

Exemples de cas d'utilisation du DLAMI

Vous trouverez ci-dessous des exemples de cas d'utilisation courants pour AWS Apprentissage profond (deep learning) AMIs (DLAMI).

En savoir plus sur l'apprentissage profond — Le DLAMI est un excellent choix pour l'apprentissage ou l'enseignement de cadres d'apprentissage automatique et d'apprentissage profond. Cela DLAMIs

vous évitera le casse-tête lié au dépannage des installations de chaque framework et à leur capacité à jouer sur le même ordinateur. Ils DLAMIs incluent un bloc-notes Jupyter et facilitent l'exécution des didacticiels proposés par les frameworks aux personnes novices en apprentissage automatique et en apprentissage profond.

Développement d'applications — Si vous êtes un développeur d'applications qui souhaite utiliser le deep learning pour que vos applications utilisent les dernières avancées en matière d'IA, le DLAMI est le banc d'essai idéal pour vous. Chaque infrastructure est fournie avec des didacticiels pour vous aider à faire vos premiers pas avec l'apprentissage profond, et nombre d'entre elles ont des zoos modèles qui facilite l'adoption de l'apprentissage profond sans avoir à créer les réseaux neuronaux vous-même ou pour effectuer la formation du modèle. Certains exemples vous montrent comment construire une application de détection d'image en seulement quelques minutes ou comment construire une application de reconnaissance vocale pour votre propre chatbot.

Apprentissage automatique et analyse des données — Si vous êtes un scientifique des données ou si vous souhaitez traiter vos données par le biais du deep learning, vous constaterez que de nombreux frameworks prennent en charge R et Spark. Vous trouverez des didacticiels sur la manière de faire des régressions simples, jusqu'à la création évolutive de systèmes de traitement de données évolutifs pour les systèmes de personnalisation et de prévisions.

Recherche — Si vous êtes un chercheur qui souhaite tester un nouveau framework, tester un nouveau modèle ou former de nouveaux modèles, le DLAMI AWS et ses capacités d'évolutivité peuvent atténuer les difficultés liées aux installations fastidieuses et à la gestion de plusieurs nœuds d'entraînement.

> **Note**
>
> Bien que votre choix initial soit de mettre à niveau votre type d'instance vers une instance plus grande GPUs (jusqu'à 8), vous pouvez également effectuer une mise à l'échelle horizontale en créant un cluster d'instances DLAMI. Consultez Informations connexes sur DLAMI pour plus d'informations sur les builds de cluster.

Caractéristiques du DLAMI

Les fonctionnalités de AWS Apprentissage profond (deep learning) AMIs (DLAMI) incluent des frameworks d'apprentissage profond préinstallés, des logiciels GPU, des serveurs de modèles et des outils de visualisation de modèles.

Frameworks préinstallés

Il existe actuellement deux versions principales de DLAMI, avec d'autres variantes liées au système d'exploitation (OS) et aux versions logicielles :

- AMI d'apprentissage profond avec Conda— Frameworks installés séparément à l'aide de `conda` packages et d'environnements Python distincts.
- AMI de base de Deep Learning— Aucun framework n'est installé ; uniquement NVIDIA CUDA et autres dépendances.

L'AMI Deep Learning avec Conda utilise des `conda` environnements pour isoler chaque framework, afin que vous puissiez passer de l'un à l'autre à votre guise sans vous soucier des conflits de dépendances entre eux. L'AMI Deep Learning avec Conda prend en charge les frameworks suivants :

- PyTorch
- TensorFlow 2

> **Note**
>
> DLAMI ne prend plus en charge les frameworks d'apprentissage profond suivants : Apache MXNet, Microsoft Cognitive Toolkit (CNTK), Caffe, Caffe2, Theano, Chainer et Keras.

Logiciel GPU préinstallé

Même si vous utilisez une instance utilisant uniquement le processeur, elle DLAMIs disposera de NVIDIA CUDA et de NVIDIA cuDNN. Le logiciel installé est le même, quel que soit le type d'instance. N'oubliez pas que les outils spécifiques au GPU ne fonctionnent que sur une instance dotée d'au moins un GPU. Pour plus d'informations sur les types d'instances, consultezChoix d'un type d'instance DLAMI.

Pour plus d'informations sur CUDA, consultezInstallations de CUDA et liaisons d'infrastructures.

Service et visualisation de modèles

L'AMI Deep Learning avec Conda est préinstallée avec des serveurs de modèles pour TensorFlow, ainsi que TensorBoard pour les visualisations de modèles. Pour de plus amples informations, veuillez consulter TensorFlow Servir.

Notes de AMIs mise à jour du Deep Learning

Vous trouverez ici des notes de mise à jour détaillées pour toutes les options actuellement prises en charge AWS Apprentissage profond (deep learning) AMIs (DLAMI).

Pour les notes de publication relatives aux frameworks DLAMI que nous ne prenons plus en charge, consultez la section Archive des notes de version des frameworks non pris en charge de la page Politique de support du cadre DLAMI.

> **Note**
>
> Ils AWS Apprentissage profond (deep learning) AMIs publient les correctifs de sécurité à une cadence nocturne. Nous n'incluons pas ces correctifs de sécurité incrémentiels dans les notes de publication.

Notes de mise à jour

- DLAMI compatible P6
- Notes de mise à jour pour Base DLAMIs
- Notes de mise à jour pour Single Framework DLAMIs
- Notes de mise à jour pour Multi-Framework DLAMIs

DLAMI compatible P6

Vous trouverez ci-dessous les exigences détaillées pour exécuter le DLAMI sur EC2 les instances Amazon P6

P6 pris en charge DLAMIs

Les DLAMI suivants prennent en charge les instances P6 :

- AWS AMI de base de Deep Learning (Amazon Linux 2023)
- AWS AMI de base d'apprentissage profond (Ubuntu 24.04)
- AWS AMI de base d'apprentissage profond (Ubuntu 22.04)

Ces DLAMI contiennent les logiciels suivants requis pour faire fonctionner les instances P6-B200 :

Logiciels	Version minimale requise
Boîte à outils Nvidia CUDA	12,8
Pilote Nvidia	R570
NV LINK 5	R570
Noyau Linux	6.1
Adaptateur Elastic Fabric (EFA)	1,41,0
AWS Plug-in OFI NCCL	1.15.0

Confirmer le fonctionnement du GPU

Pour confirmer le bon fonctionnement GPUs :

1. Exécutez le test de requête sur le périphérique GPU Nvidia suivant

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

2. Confirmez le résultat suivant de l'exécution de la requête sur le périphérique :

```
$ /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
```

Pour vérifier le bon fonctionnement du pilote NVIDIA :

1. Exécutez l'interface de gestion du système Nvidia

```
$ nvidia-smi
```

2. Confirmez le résultat suivant à partir de l'interface de gestion du système

```
+-----------------------------------------------------------------------------------------
+
| 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
       |
+-----------------------------------------------------------------------------------------
+
```

Si vous rencontrez des problèmes avec les instances P6-B200, contactez le Support. AWS

Notes de mise à jour pour Base DLAMIs

Notes de mise à jour du DLAMI X86 Base

- Notes de mise à jour du DLAMI X86 Base
- ARM64 Notes de mise à jour du DLAMI de base

Notes de mise à jour du DLAMI X86 Base

Vous trouverez ci-dessous les notes de mise à jour du DLAMI X86 Base :

GPU

- AWS AMI de base d'apprentissage profond (Amazon Linux 2023) (compatible avec le P6-B200)
- AWS AMI de base d'apprentissage profond (Ubuntu 24.04) (compatible avec le P6-B200)
- AWS AMI de base d'apprentissage profond (Ubuntu 22.04) (compatible avec le P6-B200)
- AWS AMI de base de Deep Learning (Amazon Linux 2)

Qualcomm

- AWS Base d'apprentissage profond (AMI Qualcomm) (Amazon Linux 2)

AWS Neurone

- Reportez-vous au guide de l'utilisateur du DLAMI Neuron.

AWS AMI GPU basée sur le Deep Learning (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- AMI GPU du pilote Nvidia OSS basé sur le Deep Learning (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI
- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en, P6-B200

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2023
- Architecture de calcul : x86
- La dernière version disponible est installée pour les packages suivants :
  - Noyau Linux : 6.1
  - FSx Lustre
  - NVIDIA GDS
  - Docker
  - AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/bin/aws
  - NVIDIA DCGM
  - Boîte à outils pour conteneurs Nvidia :
    - Commande de version : nvidia-container-cli -V
  - NVidia-Docker 2 :
    - Commande de version : nvidia-docker version
- Pilote NVIDIA : 570.133.20

- Stack NVIDIA CUDA 12.4-12.6 et 12.8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-xx.x/ usr/local/cuda
    - Exemple :/usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
  - Version NCCL compilée : 2.26.5
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/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
      - CHEMIN à avoir/usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include/
      - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- Installateur EFA : 1.40.0
- Nvidia GDRCopy : 2,5
- AWS NCCL OFI : 1.14.2-aws
  - AWS OFI NCCL prend désormais en charge plusieurs versions NCCL avec une seule version
  - Le chemin d'installation :/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
- AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/bin/aws
- Type de volume EBS : GP3
- Python :/ usr/bin/python 3.9
- NVMe Emplacement du magasin d'instances (sur les EC2 instances prises en charge) :/opt/dlami/nvme
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote OSS Nvidia :

```
aws ssm get-parameter --region us-east-1 \
    --name /aws/service/deeplearning/ami/x86_64/base-oss-nvidia-driver-gpu-al2023/latest/ami-id \
    --query "Parameter.Value" --output text
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Avis

Boîte à outils NVIDIA Container 1.17.4

Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Politique de support

Ces AMIs composants de cette AMI, tels que les versions CUDA, peuvent être supprimés et modifiés en fonction de la politique de support du framework ou pour optimiser les performances des conteneurs de deep learning ou pour réduire la taille de l'AMI dans une future version, sans préavis. Nous supprimons les versions CUDA AMIs si elles ne sont utilisées par aucune version du framework prise en charge.

Instances P6-B200

Les instances P6-B200 contiennent 8 cartes d'interface réseau et peuvent être lancées à l'aide de la commande suivante : 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"
```

Instances P5en

Les instances P5en contiennent 16 cartes d'interface réseau et peuvent être lancées à l'aide de la commande suivante : 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"
```

Instances P5/P5e

Les instances P5 et P5e contiennent 32 cartes d'interface réseau et peuvent être lancées à l'aide de la commande suivante : 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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
sudo dnf versionlock kernel*
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer une mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

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

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-05-15

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250515

Ajouté

- Ajout du support pour les instances P6-B200 EC2

Mis à jour

- Mise à niveau du programme d'installation d'EFA de la version 1.38.1 à la version 1.40.0
- Mise à niveau GDRCopy de la version 2.4 à la version 2.5
- Plugin AWS OFI NCCL amélioré de la version 1.13.0-aws à la version 1.14.2-aws
- Version NCCL compilée mise à jour de la version 2.25.1 à 2.26.5
- Version CUDA par défaut mise à jour de la version 12.6 à 12.8
- Version Nvidia DCGM mise à jour de 3.3.9 à 4.4.3

Date de sortie : 2025-04-22

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250421

Mis à jour

- Mise à niveau du pilote Nvidia de la version 570.124.06 à la version 570.133.20 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA d'avril 2025

Date de sortie : 2025-03-31

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250328

Ajouté

- Ajout du support pour NVIDIA GPU Direct Storage (GDS)

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250215

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de

CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-02-05

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250205

Ajouté

- Ajout de la version 12.6 du kit d'outils CUDA dans le répertoire/-12.6 usr/local/cuda
- Support supplémentaire pour les instances G5 EC2

Supprimé

- Les versions 12.1 et 12.2 de CUDA ont été supprimées de ce DLAMI. Les clients qui ont besoin de ces versions du kit d'outils CUDA peuvent les installer directement depuis NVIDIA en utilisant le lien ci-dessous
  - https://developer.nvidia.com/cuda-toolkit-archive

Date de sortie : 2025-02-03

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250131

Mis à jour

- Version EFA mise à niveau de 1.37.0 à 1.38.0
  - EFA intègre désormais le plugin AWS OFI NCCL, qui se trouve désormais dans/-ofi-nccl/. opt/amazon/ofi-nccl rather than the original /opt/aws Si vous mettez à jour votre variable LD_LIBRARY_PATH, assurez-vous de modifier correctement votre emplacement OFI NCCL.
- Mise à niveau de Nvidia Container Toolkit de la version 1.17.3 à la version 1.17.4

Date de sortie : 2025-01-08

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250107

Mis à jour

- Ajout du support pour les instances G4dn

Date de sortie : 2024-12-09

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20241206

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.17.0 à la version 1.17.3

Date de sortie : 2024-11-21

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20241121

Ajouté

- Ajout du support pour les instances P5en. EC2

Mis à jour

- Mise à niveau du programme d'installation d'EFA de la version 1.35.0 à la version 1.37.0
- Mise à niveau du plugin AWS OFI NCCL de la version 1.121-aws à la version 1.13.0-aws

Date de sortie : 2024-10-30

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20241030

Ajouté

- Version initiale du DLAMI OSS (Deep Learning Base) pour Amazon Linux 2023

Problèmes connus

- Ce DLAMI ne prend pas en charge les instances G4dn et EC2 G5 pour le moment. AWS est conscient d'une incompatibilité susceptible d'entraîner des échecs d'initialisation de CUDA, affectant à la fois les familles d'instances G4dn et G5 lors de l'utilisation des pilotes NVIDIA open source avec un noyau Linux version 6.1 ou ultérieure. Ce problème concerne les distributions Linux telles qu'Amazon Linux 2023, Ubuntu 22.04 ou version ultérieure, ou SUSE Linux Enterprise Server 15 SP6 ou version ultérieure, entre autres.

AWS AMI GPU basée sur le Deep Learning (Ubuntu 24.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- AMI GPU du pilote Nvidia OSS basé sur le Deep Learning (Ubuntu 24.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.
- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en, P6-B200.

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 24.04
- Architecture de calcul : x86
- La dernière version disponible est installée pour les packages suivants :
  - Noyau Linux : 6. 8
  - FSx Lustre
  - Docker
  - AWS CLI v2 à/usr/bin/aws
  - NVIDIA DCGM
  - Boîte à outils pour conteneurs Nvidia :

    - Commande de version : nvidia-container-cli -V
  - NVidia-Docker 2 :
    - Commande de version : nvidia-docker version
- Pilote NVIDIA : 570.133.20
- Stack NVIDIA CUDA 12.6 et 12.8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-xx.x/ usr/local/cuda
    - Exemple :/usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
  - Version NCCL compilée : 2.25.1
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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
      - CHEMIN à avoir/usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include/
      - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- Programme d'installation EFA : 1.40.0
- Nvidia GDRCopy : 2,5.1
- AWS NCCL OFI : 1.14.2-aws
  - Le chemin d'installation :/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
- AWS CLI v2 à/usr/bin/aws
- Type de volume EBS : GP3
- Python :/ usr/bin/python 3,12
- NVMe Emplacement du magasin d'instances (sur les EC2 instances prises en charge) :/opt/dlami/nvme
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote OSS Nvidia :

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

```
    --name /aws/service/deeplearning/ami/x86_64/base-oss-nvidia-driver-gpu-
ubuntu-24.04/latest/ami-id \
    --query "Parameter.Value" \
    --output text
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Avis

Politique de support

Ces AMIs composants de cette AMI, tels que les versions CUDA, peuvent être supprimés et modifiés en fonction de la politique de support du framework ou pour optimiser les performances des conteneurs de deep learning ou pour réduire la taille de l'AMI dans une future version, sans préavis. Nous supprimons les versions CUDA AMIs si elles ne sont utilisées par aucune version du framework prise en charge.

EC2 instance avec plusieurs cartes réseau

- De nombreux types d'instances compatibles avec EFA possèdent également plusieurs cartes réseau.
- DeviceIndex est unique à chaque carte réseau et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1.
  - Pour l'interface réseau principale (index de carte réseau 0, indice de périphérique 0), créez une interface EFA (EFA avec ENA). Vous ne pouvez pas utiliser une interface réseau uniquement EFA comme interface réseau principale.
  - Pour chaque interface réseau supplémentaire, utilisez le prochain index de carte réseau inutilisé, l'index de périphérique 1, et une interface réseau EFA (EFA avec ENA) ou EFA uniquement, selon votre cas d'utilisation, comme les exigences en bande passante ENA ou l'espace

d'adressage IP. Pour des exemples de cas d'utilisation, consultez la section Configuration EFA pour une instance P5.

- Pour plus d'informations, consultez le guide EFA ici.

Instances P6-B200

Les instances P6-B200 contiennent 8 cartes d'interface réseau et peuvent être lancées à l'aide de la commande suivante : 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"
```

Instances P5en

Le P5en contient 16 cartes d'interface réseau et peut être lancé à l'aide de la commande suivante : 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"
```

Instances P5/P5e

Les instances P5 et P5e contiennent 32 cartes d'interface réseau et peuvent être lancées à l'aide de la commande suivante : 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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

  ```
  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
  ```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer une mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

```
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
```

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-05-22

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 24.04) 20250522

Ajouté

- Ajout du support pour les instances P6-B200 EC2

Mis à jour

- Mise à niveau du programme d'installation d'EFA de la version 1.40.0 à la version 1.41.0
- Version NCCL compilée mise à jour de la version 2.25.1 à 2.26.5
- Version Nvidia DCGM mise à jour de 3.3.9 à 4.4.3

Date de sortie : 2025-05-13

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 24.04) 20250513

Ajouté

- Première version du DLAMI OSS (Deep Learning Base) pour Ubuntu 24.04

AWS AMI GPU basée sur le Deep Learning (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- AMI GPU du pilote Nvidia OSS basé sur le Deep Learning (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.

- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P6-B200.

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture de calcul : x86
- La dernière version disponible est installée pour les packages suivants :
  - Noyau Linux : 6. 8
  - FSx Lustre
  - Docker
  - AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/bin/aws
  - NVIDIA DCGM
  - Boîte à outils pour conteneurs Nvidia :
    - Commande de version : nvidia-container-cli -V
  - NVidia-Docker 2 :
    - Commande de version : nvidia-docker version
- Pilote NVIDIA : 570.133.20
- Stack NVIDIA CUDA 12.4-12.6 et 12.8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-xx.x/ usr/local/cuda
    - Exemple :/usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
  - Version NCCL compilée : 2.26.5
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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
      - CHEMIN à avoir/usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include/
      - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- Installateur EFA : 1.40.0

- Nvidia GDRCopy : 2,5
- AWS NCCL OFI : 1.14.2-aws
  - Le chemin d'installation :/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
- AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/bin/aws
- Type de volume EBS : GP3
- Python :/ usr/bin/python 3.10
- NVMe Emplacement du magasin d'instances (sur les EC2 instances prises en charge) :/opt/dlami/nvme
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote OSS Nvidia :

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

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Avis

Boîte à outils NVIDIA Container 1.17.4

Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Mises à jour de l'EFA de 1.37 à 1.38 (sortie le 31/01/2020)

EFA intègre désormais le plugin AWS OFI NCCL, qui se trouve désormais dans/-ofi-nccl/. opt/amazon/ofi-nccl rather than the original /opt/aws Si vous mettez à jour votre variable LD_LIBRARY_PATH, assurez-vous de modifier correctement l'emplacement NCCL de votre OFI.

Support multi-ENI

- Ubuntu 22.04 installe et configure automatiquement le routage des sources sur plusieurs à NICss l'aide de cloud-init lors de son démarrage initial. Si votre flux de travail inclut attaching/detaching le vôtre ENIs lorsqu'une instance est arrêtée, une configuration supplémentaire doit être ajoutée aux données utilisateur de cloud-init afin de garantir une configuration correcte des cartes réseau lors de ces événements. Un exemple de configuration du cloud est fourni ci-dessous.
- Veuillez consulter cette documentation canonique ici pour plus d'informations sur la façon de configurer la configuration cloud pour vos instances - https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automatically-/setup-multiple-nics

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

Politique de support

Ces AMIs composants de cette AMI, tels que les versions CUDA, peuvent être supprimés et modifiés en fonction de la politique de support du framework ou pour optimiser les performances des conteneurs de deep learning ou pour réduire la taille de l'AMI dans une future version, sans préavis. Nous supprimons les versions CUDA AMIs si elles ne sont utilisées par aucune version du framework prise en charge.

EC2 instances avec plusieurs cartes réseau

- De nombreux types d’instances compatibles avec EFA possèdent également plusieurs cartes réseau.
- DeviceIndex est unique à chaque carte réseau et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1.

- Pour l'interface réseau principale (index de carte réseau 0, indice de périphérique 0), créez une interface EFA (EFA avec ENA). Vous ne pouvez pas utiliser une interface réseau uniquement EFA comme interface réseau principale.
- Pour chaque interface réseau supplémentaire, utilisez le prochain index de carte réseau inutilisé, l'index de périphérique 1, et une interface réseau EFA (EFA avec ENA) ou EFA uniquement, selon votre cas d'utilisation, comme les exigences en bande passante ENA ou l'espace d'adressage IP. Pour des exemples de cas d'utilisation, consultez la section Configuration EFA pour une instance P5.
- Pour plus d'informations, consultez le guide EFA ici.

Instances P6-B200

Le P6-B200 contient 8 cartes d'interface réseau et peut être lancé à l'aide de la commande suivante : 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"
```

Instances P5en

Le P5en contient 16 cartes d'interface réseau et peut être lancé à l'aide de la commande suivante : 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"
```

Instances P5/P5e

Les instances P5 et P5e contiennent 32 cartes d'interface réseau et peuvent être lancées à l'aide de la commande suivante : 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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

  ```
  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
  ```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer une mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

  ```
  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
```

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-05-16

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250516

Ajouté

- Ajout du support pour les instances P6-B200 EC2

Mis à jour

- Mise à niveau du programme d'installation d'EFA de la version 1.39.0 à la version 1.40.0
- Mise à niveau du plugin AWS OFI NCCL de la version 1.13.0-aws à la version 1.14.2-aws
- Version NCCL compilée mise à jour de la version 2.22.3 à 2.26.5
- Version CUDA par défaut mise à jour de la version 12.6 à 12.8
- Version Nvidia DCGM mise à jour de 3.3.9 à 4.4.3

Date de sortie : 2025-05-05

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250503

Mis à jour

- Mise à niveau GDRCopy de la version 2.4.1 à la version 2.5.1

Date de sortie : 2025-04-24

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250424

Mis à jour

- Mise à niveau du pilote Nvidia de la version 570.124.06 à la version 570.133.20 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA d'avril 2025

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250214

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-02-07

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250205

Ajouté

- Ajout de la version 12.6 du kit d'outils CUDA dans le répertoire/-12.6 usr/local/cuda

Supprimé

- Les versions 12.1 et 12.2 de CUDA ont été supprimées de ce DLAMI. Les clients peuvent installer ces versions depuis NVIDIA en utilisant le lien ci-dessous
  - https://developer.nvidia.com/cuda-toolkit-archive

Date de sortie : 2025-01-31

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250131

Mis à jour

- Version EFA mise à niveau de 1.37.0 à 1.38.0
  - EFA intègre désormais le plugin AWS OFI NCCL, qui se trouve désormais dans/-ofi-nccl/. opt/amazon/ofi-nccl rather than the original /opt/aws Si vous mettez à jour votre variable LD_LIBRARY_PATH, assurez-vous de modifier correctement l'emplacement NCCL de votre OFI.
- Mise à niveau de Nvidia Container Toolkit de la version 1.17.3 à la version 1.17.4

Date de sortie : 2025-01-17

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250117

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025

Date de sortie : 2024-11-18

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20241115

Ajouté

- Ajout FSx du package Amazon pour le support Lustre.

Fixe

- En raison d'une modification apportée au noyau Ubuntu pour corriger un défaut de la fonctionnalité KASLR (Kernel Address Space Layout Randomization), les instances G4Dn/G5 ne sont pas en mesure d'initialiser correctement CUDA sur le pilote OSS Nvidia. Afin d'atténuer ce problème, ce DLAMI inclut une fonctionnalité qui charge dynamiquement le pilote propriétaire pour les instances G4Dn et G5. Veuillez prévoir une brève période d'initialisation pour ce chargement afin de garantir le bon fonctionnement de vos instances.

  Pour vérifier l'état et l'état de santé de ce service, vous pouvez utiliser la commande suivante :

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

```
active
```

Date de sortie : 2024-10-23

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20241023

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.90.07 à la version 550.127.05 pour corriger un problème CVEs présent dans le bulletin de sécurité d'affichage des GPU NVIDIA d'octobre 2024

Date de sortie : 2024-10-01

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 20.04) 20240930

Mis à jour

- Mise à niveau du pilote Nvidia et du Fabric Manager de la version 535.183.01 à la version 550.90.07
- Mise à niveau de Nvidia Container Toolkit de la version 1.16.1 à la version 1.16.2, corrigeant la vulnérabilité de sécurité CVE-2024-0133.
- Version EFA mise à niveau de la version 1.32.0 à la version 1.34.0
- Mise à niveau de NCCL vers la dernière version 2.22.3 pour toutes les versions de CUDA
  - CUDA 12.1, 12.2 mis à jour à partir de 2.18.5+ 2 CUDA12
  - CUDA 12.3 mis à jour à partir de la version 2.21.5+ 4 CUDA12

Ajouté

- Ajout de la version 12.4 du kit d'outils CUDA dans le répertoire/-12.4 usr/local/cuda
- Ajout du support pour les EC2 instances P5e.

Date de sortie : 2024-08-19

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240816

Ajouté

- Ajout du support pour l' EC2 instance G6e.

Date de sortie : 2024-06-06

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240606

Mis à jour

- Version du pilote Nvidia mise à jour vers 535.183.01 à partir de 535.161.08

Date de sortie : 2024-05-15

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240513

Supprimé

- La prise en charge d'Amazon FSx for Lustre a été supprimée dans cette version en raison d'une incompatibilité avec les dernières versions du noyau Ubuntu 22.04. Support FSx pour Lustre sera rétabli une fois que la dernière version du noyau sera prise en charge. Les clients qui ont besoin FSx de Lustre doivent continuer à utiliser l'AMI GPU Deep Learning Base (Ubuntu 20.04).

Date de sortie : 2024-04-29

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240429

Ajouté

- Première version du DLAMI OSS (Deep Learning Base) pour Ubuntu 22.04

AWS AMI de base de Deep Learning (Amazon Linux 2)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning Base (Amazon Linux 2) Version $ {XX.X}
- Version $ {XX.X} du pilote Nvidia propriétaire de Deep Learning Base (Amazon Linux 2)

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.

- Apprentissage profond avec OSS Le pilote Nvidia est compatible avec G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en
- Le Deep Learning avec pilote propriétaire Nvidia prend en charge les formats G3 (G3.16x non pris en charge), P3, P3dn

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2
- Architecture de calcul : x86
- La dernière version disponible est installée pour les packages suivants :
  - Noyau Linux : 5.10
  - Docker
  - AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/bin/aws
  - Boîte à outils pour conteneurs Nvidia :
    - Commande de version : nvidia-container-cli -V
  - NVidia-Docker 2 :
    - Commande de version : nvidia-docker version
- Python :/ usr/bin/python 3.7
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 550.163.01
  - Pilote Nvidia propriétaire : 550.163.01
- Stack NVIDIA CUDA 12.1-12.4 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-xx.x/ usr/local/cuda
  - CUDA par défaut : 12.1
    - PATH/usr/local/cudapointe vers CUDA 12.1
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/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
      - CHEMIN à avoir/usr/local/cuda-12.1/bin/:/usr/local/cuda-12.1/include/
      - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.

  - Version NCCL compilée : 2.22.3
  - Lieu des tests du NCCL :
    - all_reduce, all_gather et reduce_scatter :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
    - Pour exécuter des tests NCCL, LD_LIBRARY_PATH doit réussir avec les mises à jour ci-dessous.
      - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :
        - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
      - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- Installateur EFA : 1.38.0
- Nvidia GDRCopy : 2,4
- AWS NCCL OFI : 1.13.2
  - AWS OFI NCCL prend désormais en charge plusieurs versions NCCL avec une seule version
  - Le chemin d'installation :/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/lib64 est ajouté à LD_LIBRARY_PATH.
- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

  - Pilote Nvidia propriétaire :

```
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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

- Pilote Nvidia propriétaire :

```
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
```

Avis

Boîte à outils NVIDIA Container 1.17.4

Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Mises à jour EFA de 1.37 à 1.38 (sortie le 04/02/2025)

EFA intègre désormais le plugin AWS OFI NCCL, qui se trouve désormais dans/-ofi-nccl/. opt/amazon/ofi-nccl rather than the original /opt/aws Si vous mettez à jour votre variable LD_LIBRARY_PATH, assurez-vous de modifier correctement votre emplacement OFI NCCL.

Politique de support

Ces AMIs composants de cette AMI, tels que les versions CUDA, peuvent être supprimés et modifiés en fonction de la politique de support du framework ou pour optimiser les performances des conteneurs de deep learning ou pour réduire la taille de l'AMI dans une future version, sans préavis. Nous supprimons les versions CUDA AMIs si elles ne sont utilisées par aucune version du framework prise en charge.

EC2 instances avec plusieurs cartes réseau

- De nombreux types d'instances compatibles avec EFA possèdent également plusieurs cartes réseau.
- DeviceIndex est unique à chaque carte réseau et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1.
  - Pour l'interface réseau principale (index de carte réseau 0, indice de périphérique 0), créez une interface EFA (EFA avec ENA). Vous ne pouvez pas utiliser une interface réseau uniquement EFA comme interface réseau principale.
  - Pour chaque interface réseau supplémentaire, utilisez le prochain index de carte réseau inutilisé, l'index de périphérique 1, et une interface réseau EFA (EFA avec ENA) ou EFA uniquement, selon votre cas d'utilisation, comme les exigences en bande passante ENA ou l'espace d'adressage IP. Pour des exemples de cas d'utilisation, consultez la section Configuration EFA pour une instance P5.
  - Pour plus d'informations, consultez le guide EFA ici.

Instances P5/P5e

- Les instances P5 et P5e contiennent 32 cartes d'interface réseau et peuvent être lancées à l'aide de la commande suivante : 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"
```

Instances P5en

- Le P5en contient 16 cartes d'interface réseau et peut être lancé à l'aide de la commande suivante : 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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
sudo yum versionlock kernel*
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

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

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-04-22

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2) Version 69.3
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 67.0

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.144.03 à la version 550.163.01 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA d'avril 2025

Date de sortie : 2025-02-17

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 68.5
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 66.3

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4. Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-02-04

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 68.4
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 66.1

Mis à jour

- Version EFA mise à niveau de 1.37.0 à 1.38.0

Date de sortie : 2025-01-17

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 68.3
- Pilote AMI propriétaire Nvidia (Amazon Linux 2) Deep Learning Base, version 66.0

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025

Date de sortie : 2025-01-06

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 68.2
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 65.9

Mis à jour

- EFA mis à jour de la version 1.34.0 à la version 1.37.0
- Mise à niveau de AWS OFI NCCL de la version 1.11.0 à la version 1.13.0

Date de sortie : 2024-12-09

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 68.1
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 65.8

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.17.0 à la version 1.17.3

Date de sortie : 2024-11-09

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 67.9
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 65.6

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.16.2 à la version 1.17.0, corrigeant la vulnérabilité de sécurité CVE-2024-0134.

Date de sortie : 2024-10-22

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 67.7
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 65.4

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.90.07 à la version 550.127.05 pour corriger un problème CVEs présent dans le bulletin de sécurité d'affichage des GPU NVIDIA d'octobre 2024

Date de sortie : 2024-10-03

Noms des AMI

- Version AMI du pilote Nvidia OSS basé sur le Deep Learning (Amazon Linux 2)
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 65.2

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.16.1 à la version 1.16.2, corrigeant la vulnérabilité de sécurité CVE-2024-0133.

Date de sortie : 2024-08-27

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version 67.0

Mis à jour

- Mise à niveau du pilote Nvidia et du Fabric Manager de la version 535.183.01 à la version 550.90.07
  - Suppression de l'exigence de shell multi-utilisateurs de Fabric Manager sur la base des recommandations de Nvidia
  - Pour plus d'informations, veuillez consulter les problèmes connus relatifs au pilote Tesla 550.90.07 ici
- Version EFA mise à niveau de la version 1.32.0 à la version 1.34.0
- Mise à niveau de NCCL vers la dernière version 2.22.3 pour toutes les versions de CUDA
  - CUDA 12.1, 12.2 mis à jour à partir de 2.18.5+ 2 CUDA12
  - CUDA 12.3 a été mis à jour depuis 2.21.5+ 4 CUDA12

Ajouté

- Ajout de la version 12.4 du kit d'outils CUDA dans le répertoire/-12.4 usr/local/cuda
- Ajout du support pour les EC2 instances P5e.

Supprimé

- Suppression de la pile de la version 11.8 du kit d'outils CUDA présente dans le répertoire/-11.8 usr/local/cuda

Date de sortie : 2024-08-19

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version 66.3

Ajouté

- Ajout de la prise en charge des EC2 instances G6e.

Date de sortie : 2024-06-06

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 65.4
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 63.9

Mis à jour

- Version du pilote Nvidia mise à jour vers 535.183.01 à partir de 535.161.08

Date de sortie : 2024-05-02

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 64.7
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 63.2

Mis à jour

- Version EFA mise à jour de la version 1.30 à la version 1.32
- Plugin AWS OFI NCCL mis à jour de la version 1.7.4 à la version 1.9.1
- Boîte à outils de conteneurs Nvidia mise à jour de la version 1.13.5 à la version 1.15.0

Ajouté

- Ajout d'une pile CUDA12 .3 avec CUDA12 .3, NCCL 2.21.5, cuDNN 8.9.7

  La version 1.15.0 n'inclut PAS les packages nvidia-docker2 nvidia-container-runtime et nvidia-docker2. Il est recommandé d'utiliser les nvidia-container-toolkit packages directement en suivant la documentation du kit d'outils de conteneurs Nvidia.

Supprimé

- Suppression des piles de CUDA11 0,7, CUDA12 ,0 présentes à +/- 12,0 usr/local/cuda-11.7 and / usr/local/cuda
- Suppression du package nvidia-docker2 et de sa commande nvidia-docker dans le cadre de la mise à jour de la boîte à outils de conteneurs Nvidia de la version 1.13.5 à la version 1.15.0, qui n'inclut PAS les packages nvidia-docker2 et nvidia-docker2. nvidia-container-runtime

Date de sortie : 2024-04-04

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version 64.0

Ajouté

- Pour le pilote OSS Nvidia DLAMIs, ajout du support des instances G6 et Gr6 EC2

Date de sortie : 2024-03-29

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2), version 62.3
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 63.2

Mis à jour

- Mise à jour du pilote Nvidia de 535.104.12 à 535.161.08 dans le pilote Nvidia propriétaire et OSS. DLAMIs
- Les nouvelles instances prises en charge pour chaque DLAMI sont les suivantes :
  - Le Deep Learning avec pilote propriétaire Nvidia prend en charge les formats G3 (G3.16x non pris en charge), P3, P3dn
  - Apprentissage profond avec OSS Le pilote Nvidia est compatible avec G4dn, G5, P4d, P4de, P5.

Supprimé

- Suppression de la prise en charge des EC2 instances G4dn, G5, G3.16x par le pilote propriétaire Nvidia DLAMI.

Date de sortie : 2024-03-20

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version 63.1

Ajouté

- Ajout de awscliv2 dans l'AMI en tant que/usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awssur l'AMI du pilote OSS Nvidia

Date de sortie : 2024-03-13

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version 63.0

Mis à jour

- Pilote OSS Nvidia DLAMI mis à jour avec support G4dn et G5. Sur cette base, le support actuel ressemble à ce qui suit :
  - L'AMI de pilote Nvidia propriétaire de Deep Learning Base (Amazon Linux 2) prend en charge les formats P3, P3dn, G3, G4dn et G5.
  - Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) est compatible avec G4dn, G5, P4, P5.
- Il est recommandé d'utiliser le pilote DLAMIs OSS Nvidia pour G4dn, G5, P4, P5.

Date de sortie : 2024-02-13

Noms des AMI

- Pilote AMI Nvidia OSS basé sur le Deep Learning (Amazon Linux 2) Version 62.1
- Pilote AMI propriétaire Nvidia basé sur le Deep Learning Base (Amazon Linux 2), version 62.1

Mis à jour

- Mise à jour du pilote OSS Nvidia de 535.129.03 à 535.154.05
- EFA mis à jour de 1.29.0 à 1.30.0
- Mise à jour de l' AWS OFI NCCL de la version 1.7.3-aws à la version 1.7.4-aws

Date de sortie : 2024-02-01

Nom de l'AMI : AMI du pilote Nvidia propriétaire de Deep Learning Base (Amazon Linux 2), version 62.0

Sécurité

- Version du package runc mise à jour pour utiliser le correctif pour CVE-2024-21626.

La version 61.4

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version 61.4

Mis à jour

- Mise à jour du pilote OSS Nvidia de 535.104.12 à 535.129.03

La version 61.0

Nom de l'AMI : Deep Learning Base OSS Nvidia Driver AMI (Amazon Linux 2) Version 61.4

Mis à jour

- EFA mis à jour de 1.26.1 à 1.29.0
- GDRCopy mis à jour de 2.3 à 2.4

Ajouté

- AWS L'AMI d'apprentissage profond (DLAMI) est divisée en deux groupes distincts :
  - DLAMI utilisant le pilote propriétaire Nvidia (compatible avec P3, P3dn, G3, G5, G4dn).
  - DLAMI qui utilise le pilote Nvidia OSS pour activer EFA (pour prendre en charge les formats P4, P5).
- Veuillez consulter l'annonce publique pour plus d'informations sur la division du DLAMI.
- Pour les AWS CLI requêtes, voir le point Query AMI-ID with AWSCLI (exemple : Region is us-east-1)

La version 60.6

Nom de l'AMI : AMI de base d'apprentissage profond (Amazon Linux 2) version 60.6

Mis à jour

- AWS Plugin OFI NCCL mis à jour de la version 1.7.2 à la version 1.7.3
- Répertoires CUDA 12.0-12.1 mis à jour avec la version 2.18.5 de NCCL
- CUDA121. Mise à jour en tant que version CUDA par défaut
  - LD_LIBRARY_PATH a été mis à jour pour avoir//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
  - Pour les clients qui souhaitent passer à une autre version de CUDA, veuillez définir les variables LD_LIBRARY_PATH et PATH en conséquence.

Ajouté

- Kernel Live Patching est désormais activé. Les correctifs en temps réel permettent aux clients d'appliquer des correctifs de failles de sécurité et de bogues critiques à un noyau Linux en cours d'exécution, sans redémarrage ni interruption de l'exécution des applications. Veuillez noter que la prise en charge des correctifs en direct pour le noyau 5.10.192 prendra fin le 30 novembre 23.

La version 60.5

Nom de l'AMI : AMI de base d'apprentissage profond (Amazon Linux 2) version 60.5

Mis à jour

- Mise à jour du pilote NVIDIA de 535.54.03 à 535.104.12

  Ce dernier pilote corrige les modifications majeures de l'ABI NVML trouvées dans le pilote 535.54.03, ainsi que la régression du pilote trouvée dans le pilote 535.86.10 qui affectait les boîtes à outils CUDA sur les instances P5. Consultez les notes de mise à jour suivantes de NVIDIA pour plus de détails sur les correctifs :
  - 4235941 - Correctif de modification révolutionnaire de l'ABI NVML
  - 4228552 - Correction d'erreur du kit d'outils CUDA
- Répertoires CUDA 12.2 mis à jour avec NCCL 2.18.5
- EFA mis à jour de la version 1.24.1 à la dernière version 1.26.1

Ajouté

- Ajouté CUDA12 2.2 à/usr/local/cuda-12.2

Supprimé

- Suppression du support pour CUDA 11.5 et CUDA 11.6

La version 60.2

Nom de l'AMI : AMI de base d'apprentissage profond (Amazon Linux 2) version 60.2

Mis à jour

- aws-ofi-ncclPlugin mis à jour de la v1.7.1 à la v1.7.2

La version 60.0

Date de sortie : 2023-08-11

Ajouté

- Cette AMI prend désormais en charge les fonctionnalités d'entraînement à nœuds multiples sur P5 et sur toutes les instances précédemment prises en charge EC2
- Pour les EC2 instances P5, il est recommandé d'utiliser NCCL 2.18 et il a été ajouté à CUDA12 .0 et .1. CUDA12

Supprimé

- Suppression du support pour la CUDA11 version 5.

La version 59.2

Date de sortie : 2023-08-08

Supprimé

- Suppression de CUDA-11.3 et CUDA-11.4

La version 59.1

Date de sortie : 2023-08-03

Mis à jour

- Plugin AWS OFI NCCL mis à jour vers la version v1.7.1
- Fabriqué en CUDA11 .8 par défaut car PyTorch 2.0 supporte 11.8 et pour les EC2 instances P5, il est recommandé d'utiliser >= .8 CUDA11
  - LD_LIBRARY_PATH a été mis à jour pour avoir//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
  - Pour toute version différente de cuda, veuillez définir LD_LIBRARY_PATH en conséquence.

Fixe

- Correction du problème de chargement du package Nvidia Fabric Manager (FM) mentionné lors de la date de sortie antérieure du 19/07/2023.

La version 58.9

Date de sortie : 2023-07-19

Mis à jour

- Pilote Nvidia mis à jour de 525.85.12 à 535.54.03
- Programme d'installation d'EFA mis à jour de la version 1.22.1 à la version 1.24.1

Ajouté

- Ajout de modifications de l'état C pour désactiver l'état inactif du processeur en réglant l'état C maximal sur C1. Cette modification est effectuée en définissant `intel_idle.max_cstate=1 processor.max_cstate=1` dans les arguments de démarrage de Linux dans le fichier/etc/default/grub
- AWS EC2 Support des instances P5 :
  - Ajout de la prise en charge des EC2 instances P5 pour les flux de travail utilisant un seul nœud ou une seule instance. La prise en charge de plusieurs nœuds (par exemple pour l'entraînement

multi-nœuds) à l'aide d'EFA (Elastic Fabric Adapter) et du plug-in AWS OFI NCCL sera ajoutée dans une prochaine version.

- Veuillez utiliser CUDA>=11.8 pour des performances optimales.
- Problème connu : le chargement du package Nvidia Fabric Manager (FM) prend du temps sur P5. Les clients doivent attendre 2 à 3 minutes jusqu'à ce que FM se charge après le lancement de l'instance P5. Pour vérifier si FM est démarré, exécutez la commande sudo systemctl is-active nvidia-fabricmanager, elle devrait redevenir active avant de démarrer un flux de travail. Cela sera corrigé dans la prochaine version.

La version 58.0

Date de sortie : 2023-05-19

Supprimé

- Suppression de la pile CUDA11 .0-11.2 conformément à la politique de support mentionnée dans la section supérieure de ce document.

La version 57.3

Date de sortie : 2023-04-06

Ajouté

- Ajout de Nvidia GDRCopy 2.3

La version 5.6.8

Date de sortie : mars 2009

Mis à jour

- Mise à jour du pilote NVIDIA 515.65.01 à 525.85.12

Ajouté

- Ajouté cuda-11.8 à/-11.8/ usr/local/cuda

La version 56.0

Date de sortie : 03.12-06

Mis à jour

- Version EFA mise à jour de 1.17.2 à 1.19.0

La version 5.5.0

Date de sortie : 04-11-04

Mis à jour

- Pilote NVIDIA mis à jour de 510.47.03 à 515.65.01

Ajouté

- Ajouté cuda-11.7 à/-11.7/ usr/local/cuda

La version 54.0

Date de sortie : 03.09-15

Mis à jour

- Version EFA mise à jour de 1.16.0 à 1.17.2

La version 5.3.3

Date de sortie : 05.05-25

Mis à jour

- Mise à jour aws-efa-installer vers version 1.15.2
- Mise à jour aws-ofi-nccl vers la version 1.3.0-aws qui inclut la topologie pour p4de.24xlarge.

Ajouté

- Cette version ajoute la prise en charge des instances EC2 p4de.24xlarge.

La version 53.0

Date de sortie : 04-04-28

Ajouté

- Ajout d'un CloudWatch agent Amazon
- Ajout de trois services systemd qui utilisent des fichiers json prédéfinis disponibles sur path/opt/aws/amazon-cloudwatch-agent/etc/pour configurer les métriques du GPU à l'aide de l'utilisateur linux cwagent
  - dlami-cloudwatch-agent@minimal
    - Commandes pour activer les métriques du GPU :

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

    - Il crée les métriques suivantes : `utilization_gpu` `utilization_memory`
  - dlami-cloudwatch-agent@partial
    - Commandes pour activer les métriques du GPU :

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

    - Il crée les métriques suivantes :`utilization_gpu`,`utilization_memory`,`memory_total`,`memory_used`, `memory_free`
  - dlami-cloudwatch-agent@all
    - Commandes pour activer les métriques du GPU :

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

    - Il crée toutes les métriques GPU disponibles

La version 52.0

Date de sortie : 03-03-08

Mis à jour

- Version du noyau mise à jour vers la version 5.10

La version 51.0

Date de sortie : 03.03-04

Mis à jour

- Mise à jour du pilote Nvidia vers la version 510.47.03

La version 50.0

Date de sortie : 04.02-17

Mis à jour

- Verrouillés aws-neuron-dkms et tensorflow-model-server-neuron lorsqu'ils sont mis à jour vers des versions plus récentes qui ne sont pas prises en charge par les packages Neuron présents dans l'AMI
  - Commandes si le client souhaite déverrouiller le package pour le mettre à jour avec la version la plus récente : sudo yum versionlock delete aws-neuron-dkms sudo yum versionlock delete tensorflow-model-server-neuron

La version 49.0

Date de sortie : 03.01-13

Ajouté

- Ajout de la version CUDA11 2 avec les composants suivants :
  - cuDNN v8.1.1.33
  - NCCL 2.8.4
  - CUDA 11.2.2

Mis à jour

- Mise à jour du lien symbolique pip vers pip3

Obsolescence

- Support obsolète pour le type d'instance P2
- Python2.7 obsolète et suppression des paquets python2.7 associés tels que « python-dev », « python-pip » et « python-tk »

La version 48.0

Date de sortie : 2021-12-27

Mis à jour

- Le fichier org.apache.ant_1.9.2.v201404171502 \ lib \ ant-apache-log 4j.jar a été supprimé des versions de cuda car il n'est pas utilisé et il n'y a aucun risque pour les utilisateurs possédant les fichiers Log4j. Pour plus d'informations, consultez https://nvidia.custhelp.com/app/answers/detail/a_id/5294.

La version 47.0

Date de sortie : 2021-11-24

Mis à jour

- Mise à jour d'EFA vers la version 1.14.1

La version 46.0

Date de sortie : 2021-11-12

Mis à jour

- Packages Neuron mis à jour à partir de aws-neuron-dkms =1.5. *, aws-neuron-runtime-base =1,5. *, aws-neuron-tools =1,6.* à =2,2. aws-neuron-dkms *, aws-neuron-runtime-base =1,6. *, aws-neuron-tools =2,0.*.
- Suppression du package Neuron aws-neuron-runtime =1.5.* car Neuron n'a plus d'environnement d'exécution en tant que démon et le runtime est désormais intégré au framework en tant que bibliothèque.

La version 45.0

Date de sortie : 2021-10-21

Ajouté

- Les rapports d'analyse de sécurité au format JSON sont disponibles à l'adresse/opt/aws/dlami/info/.

La version 44.0

Date de sortie : 2021-10-08

de modification

- Pour chaque lancement d'instance à l'aide du DLAMI, la balise aws-dlami-autogenerated-tag « do-not-delete - » sera ajoutée pour AWS permettre de collecter le type d'instance, l'ID de l'instance, le type de DLAMI et les informations du système d'exploitation. Aucune information sur les commandes utilisées dans le DLAMI n'est collectée ou conservée. Aucune autre information concernant le DLAMI n'est collectée ou conservée. Pour désactiver le suivi de l'utilisation de votre DLAMI, ajoutez une balise à votre instance EC2 Amazon lors du lancement. La balise doit utiliser la clé OPT_OUT_TRACKING avec la valeur associée définie sur true. Pour plus d'informations, consultez Marquer vos EC2 ressources Amazon.

Sécurité

- Version de docker mise à jour vers docker-20.10.7-3

La version 43.0

Date de sortie : 2021-08-24

de modification

- « Notebook » mis à jour vers la version « 6.4.1".

La version 42.0

Date de sortie : 2021-07-23

de modification

- Mise à jour de la version du pilote Nvidia et du gestionnaire Fabric vers 450.142.00.

La version 41.0

Date de sortie : 2021-06-24

de modification

- Packages Neuron mis à jour conformément à la version 1.14.0 de Neuron

La version 40.0

Date de sortie : 2021-06-10

de modification

- Version awscli mise à jour vers la version 1.19.89

La version 39.0

Date de sortie : 2021-05-27

Sécurité

- Suppression des composants vulnérables de CUDA-10.0 (Visual Profiler, Nsight EE et JRE) de l'installation de CUDA-10.0 (/-10.0). usr/local/cuda

La version 38.0

Date de sortie : 2021-05-25

de modification

- Runc mis à jour vers la dernière version

La version 37.0

Date de sortie : 2021-04-23

de modification

- Mise à jour du pilote Nvidia Tesla et de la version Fabric Manager vers 450.119.03.

La version 36.1

Date de sortie : 2021-04-21

Fixe

- Correction d'un problème qui ralentissait la vitesse de lancement de l'instance.

La version 36.0

Date de sortie : 2021-03-24

Ajouté

- Ajouté tensorflow-model-server-neuron pour prendre en charge le service de modèles neuronaux.

de modification

- Mise à niveau de jupyterlab vers la version 3.0.8 pour python3.

Fixe

- L'ancienne installation d'OpenMPI dansusr/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/est disponible.
- Supprimez les définitions dupliquées et inexistantes des environnements shell qui polluaient les variables d'environnement shell telles que PATH et LD_LIBRARY_PATH. Par conséquent, ~/.dlami et/etc/profile.d/var.sh has been removed, and /etc/profile.d/dlami.sh ont été ajoutés.

Sécurité

- Cryptographie des paquets mise à jour pour résoudre le problème CVE-2020-36242

La version 35.0

Date de sortie : 2021-03-08

Ajouté

- Ajout de l'installation de TensorRT CUDA 11.0

La version 34.3

Date de sortie : 2021-02-25

Fixe

- Correction d'une faute de frappe dans le MOTD (message du jour) qui affichait incorrectement la version 34.1.

La version 34.2

Date de sortie : 2021-02-24

Sécurité

- Python2 et python3 corrigés pour CVE-2021-3177

Problème connu

- Il y a une faute de frappe dans le MOTD (message du jour) qui affichait incorrectement la version 34.1. Nous publierons la version 34.3 pour résoudre ce problème.

La version 34.0

Date de sortie : 2021-02-09

de modification

- Épinglé à la version 20.3.4 pour python2, il s'agit de la dernière version de pip supportant python2 et python3.5.

La version 33.0

Date de sortie : 2021-01-19

de modification

- Mise à jour de la version cuDNN vers la version 8.0.5.39 dans les versions 1.0 et 8.1 CUDA11. CUDA11

La version 32.0

Date de sortie : 2020-12-01

Ajouté

- Ajout de la version CUDA11 .1 avec NCCL 2.7.8, cuDNN 8.0.4.30 pour l'AMI d'apprentissage profond (Amazon Linux 2), l'AMI d'apprentissage profond (Ubuntu 16.04), l'AMI d'apprentissage profond (Ubuntu 18.04), l'AMI de base d'apprentissage profond (Ubuntu 16.04), l'AMI de base d'apprentissage profond (Amazon Linux 2).

La version 31.0

Date de sortie : 2020-11-02

de modification

- Installation d'EFA mise à niveau vers la version 1.10.0.
- Mise à niveau de la version cuDNN vers la version 8.0.4.30 pour CUDA 11.0.
- Mise à niveau de AWS Neuron vers la version 1.1

La version 30.0

Date de sortie : 2020-10-08

de modification

- Versions du pilote NVIDIA et du Fabric Manager mises à jour vers la version 450.80.02
- Mise à jour de NCCL vers la version 2.7.8 dans la version 2.0 CUDA11

Fixe

- Correction d'un problème en raison duquel le package python géré par yum était remplacé par des installations gérées par pip. Les exécutables pip, pip3 et pip3.7 ont été déplacés de/usr/binto /usr/local/binasdans le cadre de ce correctif.

La version 29.0

Date de sortie : 2020-09-11

de modification

- Pilote NVIDIA mis à jour de la version 450.51.05 à 450.51.06
- Ajout de la version 450.51.06 de NVIDIA Fabric Manager
- Mise à niveau d'EFA vers la version 1.9.4

La version 28.0

Date de sortie : 2020-08-19

de modification

- Ajout de la pile CUDA 11.0 avec NCCL 2.7.6 et cuDNN 8.0.2.39

La version 27.0

Date de sortie : 2020-08-07

de modification

- EFA mis à jour de la version 1.7.1 à la version 1.9.3 à/opt/amazon/efa
- La mise à niveau d'Open MPI de la version 4.0.3 à la version 4.0.4 dans «/usr/local/mpi'. Open MPI at '/opt/amazon/openmpi/bin/mpirun» est toujours à la version 4.0.3
- Mise à jour du pilote NVIDIA 440.33.01 à 450.51.05
- Mise à niveau de la version NCCL de 2.6.4 à 2.7.6 en 0.2 CUDA1

La version 26.0

Date de sortie : 2020-08-03

de modification

- Mise à jour de l' AWS OFI NCCL vers la version la plus récente, voir ici pour plus de détails.
- Cuda 8.0/9.0/9.2 ont été supprimés de l'AMI

Fixe

- Correction d'une erreur empêchant l'ouverture du fichier objet partagé : libopencv_dnn.so.4.2.

La version 25.0

Date de sortie : 2020-07-19

de modification

- Version EFA mise à jour vers la version 1.7.1 pour prendre en charge NCCL 2.6.4
- Version NCCL mise à jour vers la version 2.6.4 pour CUDA 10.2
- Version awscli mise à jour de la version 1.16.76 à la version 1.18.80
- Version boto3 mise à jour de la version 1.9.72 à la version 1.14.3

La version 24.1

Date de sortie : 2020-06-14

de modification

- Version Docker mise à jour vers la version 19.03.6

La version 24.0

Date de sortie : 2020-05-20

de modification

- Version Docker mise à jour vers la version 19.03.6

La version 23.0

Date de sortie : 2020-04-29

de modification

- Versions du package python mises à niveau

La version 22.0

Date de sortie : 2020-03-04

de modification

- Ajout de la pile CUDA 10.2
- Mise à jour de CUDA 10.0 et 10.1 pour les versions cuDNN et NCCL

AWS Base d'apprentissage profond (AMI Qualcomm) (Amazon Linux 2)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Base d'apprentissage profond (AMI Qualcomm) (Amazon Linux 2) $ {YYYY-MM-DD}

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2
- Architecture de calcul : x86
- Noyau Linux : 5.10.210-201.852.amzn2.x86_64
- Emplacement du SDK : /opt/qai-sdk
- Emplacement des utilitaires QTI : /opt/qti-aic
- Version du SDK de la plate-forme : 1.12.0.88
- Version du SDK pour applications : 1.12.0.87
- Type de volume EBS : GP3
- Python : python3.8
- EC2 Instances prises en charge : dl2q
- Interrogez l'AMI-ID avec AWS CLI (par exemple, la région est 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
```

Avis

[24 avril] Suppression du package d'audit

- DLAMIs publiés entre le 26 mars 2024 (2024-03-26) et le 12 avril 2024 (2024-04-12) ont été expédiés sans le package d'audit. Si vous avez besoin de ce package spécifique pour vos besoins de journalisation et de surveillance, veuillez migrer vos flux de travail vers le DLAMI le plus récent afin de les utiliser avec le package d'audit installé.

Environnement QAIC :

- Par défaut, les qaic-pytools ne sont pas activés via le SDK Qualcomm AI1 00 Apps. Pour activer et créer l'environnement qaic pip qaic-env, veuillez exécuter les commandes suivantes :

```
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
```

Version 20240314

Date de sortie : 2024-03-14

Nom de l'AMI : Deep Learning Base Qualcomm AMI (Amazon Linux 2) 20240314

Ajouté

- Mise à jour du SDK de la plateforme AI 100 de la version 1.10.0.200 à la version 1.12.0.88
- Le SDK AI 100 Apps a été mis à jour de la version 1.10.0.193 à la version 1.12.0.87
- La version 1.12 du SDK ajoute la prise en charge des modèles de décodeurs transformateurs tels que Llama-2 et Starcoder

Version 20240110

Date de sortie : 2024-01-10

Nom de l'AMI : Deep Learning Base Qualcomm AMI (Amazon Linux 2) 20240103

Ajouté

- Image de la plateforme Qualcomm AI 100 mise à niveau vers la version 1.10.0.200

Version : 2023 1115

Date de sortie : 2023-11-15

Nom de l'AMI : Deep Learning Base Qualcomm AMI (Amazon Linux 2) 20231115

Ajouté

- Première version de la série Deep Learning Base Qualcomm AMI (Amazon Linux 2).
  - Consultez la documentation officielle de Qualcomm pour plus d'informations sur le SDK de la plateforme et des applications : https://quic.github.io//cloud-ai-sdk-pages
  - Reportez-vous à la AWS documentation officielle pour en savoir plus sur les instances dl2q : DL2q instances.

ARM64 Notes de mise à jour du DLAMI de base

- ARM64 Notes de mise à jour du DLAMI de base

ARM64 Notes de mise à jour du DLAMI de base

Vous trouverez ci-dessous les notes de mise à jour relatives au ARM64 DLAMI de base :

GPU

- AWS ARM64 AMI de base de Deep Learning (Amazon Linux 2023)
- AWS ARM64 AMI de base d'apprentissage profond (Ubuntu 22.04)
- AWS ARM64 AMI de base de Deep Learning (Amazon Linux 2)

AWS Neurone

- Reportez-vous au guide de l'utilisateur du DLAMI Neuron.

AWS AMI GPU ARM64 basée sur le Deep Learning (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- AMI GPU du pilote Nvidia OSS ARM64 basé sur le Deep Learning (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2023
- Architecture informatique : ARM64
- Noyau Linux : 6.12
- Pilote NVIDIA : 570.133.20
- Pile NVIDIA CUDA 12,4, 12,5, 12,6, 12,8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-xx.x/ usr/local/cuda
    - Exemple :/usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
  - Version NCCL compilée :
    - Pour le répertoire CUDA de 12.4, compilé la version 2.22.3+ 4 de NCCL CUDA12
    - Pour le répertoire CUDA de 12.5, compilé la version 2.22.3+ .5 de NCCL CUDA12
    - Pour le répertoire CUDA de 12.6, compilé la version NCCL 2.24.3+ .6 CUDA12
    - Pour le répertoire CUDA de 12.8, compilé la version NCCL 2.26.2+ .8 CUDA12
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :

    - LD_LIBRARY_PATH doit avoir/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
    - CHEMIN à avoir/usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include/
    - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- AWS CLI v2 à/usr/local/bin/aws
- Type de volume EBS : GP3
- Boîte à outils pour conteneurs Nvidia : 1.17.4
  - Commande de version : nvidia-container-cli -V
- Docker : 25,0,5
- Python :/ usr/bin/python 3.9
- Requête AMI-ID avec le paramètre SSM (exemple de région : us-east-1) :

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

- Requête AMI-ID avec AWSCLI (exemple de région : 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
```

Avis

Boîte à outils NVIDIA Container 1.17.4

Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Politique de support

Ces AMIs composants de cette AMI, tels que les versions CUDA, peuvent être supprimés et modifiés en fonction de la politique de support du framework ou pour optimiser les performances des conteneurs de deep learning ou pour réduire la taille de l'AMI dans une future version, sans préavis. Nous supprimons les versions CUDA AMIs si elles ne sont utilisées par aucune version du framework prise en charge.

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
sudo dnf versionlock kernel*
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

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

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-04-24

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250424

Mis à jour

- Mise à niveau du pilote Nvidia de la version 570.86.15 à la version 570.133.20 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA d'avril 2025
- Mise à jour de CUDA12 1.8 stack avec NCCL 2.26.2
- CUDA par défaut mis à jour de 12.6 à 12.8

Date de sortie : 2025-04-22

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250421

Mis à jour

- Mise à niveau du pilote Nvidia de la version 570.124.06 à la version 570.133.20 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA d'avril 2025

Date de sortie : 2025-04-04

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250404

Mis à jour

- Version du noyau mise à jour de 6.1 à 6.12

Date de sortie : 2025-03-03

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250303

Mis à jour

- Pilote Nvidia de 550.144.03 à 570.86.15
- Le CUDA par défaut est passé de CUDA12 0,4 à CUDA12 6.

Ajouté

- Répertoire CUDA de 12.5 avec version NCCL CUDA12 2.22.3+ .5 compilée et cuDNN 9.7.1.26
- Répertoire CUDA de 12.6 avec version NCCL CUDA12 2.24.3+ .6 compilée et cuDNN 9.7.1.26
- Répertoire CUDA de 12.8 avec version NCCL CUDA12 2.25.1+ .8 compilée et cuDNN 9.7.1.26

Date de sortie : 2025-02-14

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) 20250214

Ajouté

- Publication initiale du DLAMI OSS (Deep Learning ARM64 Base) pour Amazon Linux 2023

AWS AMI GPU ARM64 basée sur le Deep Learning (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- AMI GPU du pilote Nvidia OSS ARM64 basé sur le Deep Learning (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture informatique : ARM64
- Noyau Linux : 6.8.0-1027-aws
- Pilote NVIDIA : 570.133.20
- Pile NVIDIA CUDA 12,4, 12,5, 12,6, 12,8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-xx.x/ usr/local/cuda
    - Exemple :/usr/local/cuda-12.8/ , /usr/local/cuda-12.8/
  - Version NCCL compilée :
    - Pour le répertoire CUDA de 12.4, compilé la version 2.22.3+ 4 de NCCL CUDA12
    - Pour le répertoire CUDA de 12.5, compilé la version 2.22.3+ .5 de NCCL CUDA12
    - Pour le répertoire CUDA de 12.6, compilé la version NCCL 2.24.3+ .6 CUDA12
    - Pour le répertoire CUDA de 12.8, compilé la version NCCL 2.26.2+ .8 CUDA12
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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

    - CHEMIN à avoir/usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include/
    - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/bin/aws
- Type de volume EBS : GP3
- Boîte à outils pour conteneurs Nvidia : 1.17.4
  - Commande de version : nvidia-container-cli -V
- NVIDIA DCGM : 3,3
  - Commande de version dcgmi -v
- Docker : 26,12
- Python :/ usr/bin/python 3.10
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :

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

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est 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
```

Avis

Boîte à outils NVIDIA Container 1.17.4

Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Support multi-ENI

- Ubuntu 22.04 installe et configure automatiquement le routage des sources sur plusieurs NICs via cloud-init lors de son démarrage initial. Si votre flux de travail inclut attaching/detaching vos ENI lorsqu'une instance est arrêtée, une configuration supplémentaire doit être ajoutée aux données utilisateur de cloud-init pour garantir une configuration correcte des cartes réseau lors de ces événements. Un exemple de configuration du cloud est fourni ci-dessous.
- Veuillez consulter cette documentation canonique ici pour plus d'informations sur la façon de configurer la configuration cloud pour vos instances - https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automatically-/setup-multiple-nics

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

Politique de support

Ces AMIs composants de cette AMI, tels que les versions CUDA, peuvent être supprimés et modifiés en fonction de la politique de support du framework ou pour optimiser les performances des conteneurs de deep learning ou pour réduire la taille de l'AMI dans une future version, sans préavis. Nous supprimons les versions CUDA AMIs si elles ne sont utilisées par aucune version du framework prise en charge.

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
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
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

```
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
```

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-04-24

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250424

Mis à jour

- Mise à niveau du pilote Nvidia de la version 570.86.15 à la version 570.133.20 afin de remédier aux problèmes CVE présents dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA d'avril 2025
- Stack CUDA 12.8 mis à jour avec NCCL 2.26.2
- CUDA par défaut mis à jour de 12.6 à 12.8
- CUDA 12.3 supprimé

Date de sortie : 2025-03-03

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250303

Mis à jour

- Pilote Nvidia de 550.144.03 à 570.86.15
- Le CUDA par défaut est passé de CUDA12 1 à CUDA12 6.

Ajouté

- Répertoire CUDA de 12.4 avec version NCCL CUDA12 2.22.3+ .4 compilée et cuDNN 9.7.1.26
- Répertoire CUDA de 12.5 avec version NCCL CUDA12 2.22.3+ .5 compilée et cuDNN 9.7.1.26
- Répertoire CUDA de 12.6 avec version NCCL CUDA12 2.24.3+ .6 compilée et cuDNN 9.7.1.26
- Répertoire CUDA de 12.8 avec version NCCL CUDA12 2.25.1+ .8 compilée et cuDNN 9.7.1.26

Supprimé

- Répertoire CUDA des versions 12.1 et 12.2

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250214

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-17

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20250117

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025

Date de sortie : 2024-10-23

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20241023

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.90.07 à la version 550.127.05 pour corriger un problème CVEs présent dans le bulletin de sécurité d'affichage des GPU NVIDIA d'octobre 2024

Date de sortie : 2024-06-06

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240606

Mis à jour

- Version du pilote Nvidia mise à jour vers 535.183.01 à partir de 535.161.08

Date de sortie : 2024-05-15

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04) 20240514

Ajouté

- Première version du DLAMI OSS (Deep Learning ARM64 Base) pour Ubuntu 22.04

AWS AMI GPU ARM64 basée sur le Deep Learning (Amazon Linux 2)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- AMI GPU du pilote Nvidia OSS ARM64 basé sur le Deep Learning (Amazon Linux 2) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2
- Architecture informatique : ARM64
- Noyau Linux : 5.10
- Pilote NVIDIA : 550.144.03
- Pile NVIDIA CUDA 12.1, 12.2, 12.3 :
  - Répertoires d'installation CUDA, NCCL et CudDN :

    - Exemple :/usr/local/cuda-12.1/ , /usr/local/cuda-12.1/
  - Version NCCL compilée :
    - Pour le répertoire CUDA de 12.3, compilé la version 2.21.5+ 4 de NCCL CUDA12
    - Pour les répertoires CUDA de 12.1, 12.2, version NCCL compilée 1.18.5+ 2. CUDA12
  - CUDA par défaut : 12.1
    - PATH/usr/local/cudapointe vers CUDA 12.1
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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
      - CHEMIN à avoir/usr/local/cuda-12.1/bin/:/usr/local/cuda-12.1/include/
      - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/bin/aws
- Type de volume EBS : GP3
- Boîte à outils pour conteneurs Nvidia : 1.16.2
  - Commande de version : nvidia-container-cli -V
- Docker : 26,12
- Python :/ usr/bin/python 3.10
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :

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

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est 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
```

Avis

Boîte à outils NVIDIA Container 1.17.4

Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Politique de support

Ces AMIs composants de cette AMI, tels que les versions CUDA, peuvent être supprimés et modifiés en fonction de la politique de support du framework ou pour optimiser les performances des conteneurs de deep learning ou pour réduire la taille de l'AMI dans une future version, sans préavis. Nous supprimons les versions CUDA AMIs si elles ne sont utilisées par aucune version du framework prise en charge.

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
sudo yum versionlock kernel*
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

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

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2) 20250214

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-17

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2) 20250117

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025

Date de sortie : 2024-10-22

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2) 20241022

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.90.07 à la version 550.127.05 pour corriger un problème CVEs présent dans le bulletin de sécurité d'affichage des GPU NVIDIA d'octobre 2024

Date de sortie : 2024-10-08

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2) 20241008

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.16.1 à la version 1.16.2, corrigeant la vulnérabilité de sécurité CVE-2024-0133.

Date de sortie : 2024-06-06

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2) 20240606

Mis à jour

- Version du pilote Nvidia mise à jour vers 535.183.01 à partir de 535.161.08

Date de sortie : 2024-05-14

Nom de l'AMI : Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux 2) 20240514

Ajouté

- Première version du DLAMI OSS (Deep Learning ARM64 Base) pour Amazon Linux 2

Notes de mise à jour pour Single Framework DLAMIs

Notes de mise à jour du DLAMI Single Framework

- PyTorch DLAMIs
- TensorFlow DLAMIs

PyTorch DLAMIs

Notes de mise à jour du DLAMI Multi Framework

- Notes de mise PyTorch à jour du DLAMI X86

- ARM64 PyTorch Notes de mise à jour du DLAMI

Notes de mise PyTorch à jour du DLAMI X86

Vous trouverez ci-dessous les notes de publication pour X86 PyTorch DLAMIs :

GPU

- AWS GPU AMI PyTorch 2.7 pour apprentissage profond (Amazon Linux 2023)
- AWS GPU AMI d'apprentissage profond PyTorch 2.7 (Ubuntu 22.04)
- AWS GPU AMI PyTorch 2.6 pour apprentissage profond (Amazon Linux 2023)
- AWS GPU AMI d'apprentissage profond PyTorch 2.6 (Ubuntu 22.04)
- AWS GPU AMI PyTorch 2.5 pour apprentissage profond (Amazon Linux 2023)
- AWS GPU AMI PyTorch 2.5 pour apprentissage profond (Ubuntu 22.04)
- AWS GPU AMI PyTorch 2.4 pour apprentissage profond (Ubuntu 22.04)

AWS Neurone

- Reportez-vous au guide de l'utilisateur du DLAMI Neuron

AWS Processeur graphique AMI OSS PyTorch 2.7 pour le Deep Learning (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI
- G4dn, G5, G5, Gr6, P4, P4de, P5, P5e, P5en, P6-B200

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2023

- Architecture de calcul : x86
- Noyau Linux : 6.1
- Pilote NVIDIA : 570.133.20
- Stack NVIDIA CUDA 12.8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-12.8/ usr/local/cuda
  - Lieu des tests du NCCL :
    - all_reduce, all_gather et reduce_scatter :

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

    - Pour exécuter des tests NCCL, LD_LIBRARY_PATH est déjà mis à jour avec les chemins nécessaires.
      - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :

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

      - LD_LIBRARY_PAT est mis à jour avec les chemins de version CUDA :

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

  - Version NCCL compilée :
    - Pour le répertoire CUDA de 12.8, compilé la version NCCL 2.26.2+ .8 CUDA12
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda/targets/x86_64-linux/lib
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
- Installateur EFA : 1.40.0
- Nvidia GDRCopy : 2,5
- AWS NCCL OFI : 1.14.2-aws
  - Le chemin d'installation :/opt/amazon/ofi-nccl/. Path /opt/amazon/ofi-nccl/libest ajouté à LD_LIBRARY_PATH

- AWS CLI v2 à/usr/local/bin/aws
- Type de volume EBS : GP3
- Boîte à outils pour conteneurs Nvidia : 1.17.7
  - Commande de version : nvidia-container-cli -V
- Docker : 25,0,8
- Python :/ usr/bin/python 3,12
- Requête AMI-ID avec le paramètre SSM (exemple de région : 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
```

- Requête AMI-ID avec AWSCLI (exemple de région : 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
```

Avis

Instances P6-B200

- Les instances P6-B200 nécessitent la version 12.8 ou supérieure de CUDA et le pilote NVIDIA 570 ou des pilotes plus récents.
- Le P6-B200 contient 8 cartes d'interface réseau et peut être lancé à l'aide de la commande CLI AWS suivante :

```
aws ec2 run-instances --region $REGION \
    --instance-type $INSTANCETYPE \
    --image-id $AMI --key-name $KEYNAME \
    --iam-instance-profile "Name=dlami-builder" \
    --tag-specifications "ResourceType=instanace,Tags=[{Key=Name,Value=$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"
```

Instances P5/P5e

- DeviceIndex est unique à chacun NetworkCard et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1. Vous trouverez ci-dessous un exemple de commande de lancement d'instance EC2 P5 utilisant awscli, s'affichant NetworkCardIndex entre 0 et DeviceIndex 31, 0 pour la première interface et 1 pour les 31 interfaces restantes.

```
aws ec2 run-instances --region $REGION \
    --instance-type $INSTANCETYPE \
    --image-id $AMI --key-name $KEYNAME \
    --iam-instance-profile "Name=dlami-builder" \
    --tag-specifications "ResourceType=instanace,Tags=[{Key=Name,Value=$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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
sudo dnf versionlock kernel*
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions

de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

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

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, le support de Conda est PyTorch devenu obsolète (voir l'annonce officielle). Par conséquent, les PyTorch versions 2.6 et supérieures utiliseront les environnements virtuels Python. Pour activer le PyTorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

Date de sortie : 2025-05-22

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Amazon Linux 2023) 20250520

Ajouté

- Version initiale de la série de GPU AMI Deep Learning PyTorch 2.7 (Amazon Linux 2023). Incluant un environnement virtuel Python pytorch (source/opt/pytorch/bin/activate) complété par le pilote NVIDIA R570, CUDA=12.8, cuDNN=9.10, NCCL=2.26.2 et EFA=1.40.0. PyTorch

AWS Processeur graphique AMI OSS pour apprentissage profond PyTorch 2.7 (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI
- G4dn, G5, G5, Gr6, P4, P4de, P5, P5e, P5en, P6-B200

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture de calcul : x86
- Noyau Linux : 6.8
- Pilote NVIDIA : 570.133.20
- Stack NVIDIA CUDA 12.8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-12.8/ usr/local/cuda
  - Lieu des tests du NCCL :
    - all_reduce, all_gather et reduce_scatter :

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

    - Pour exécuter des tests NCCL, LD_LIBRARY_PATH est déjà mis à jour avec les chemins nécessaires.
      - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :

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

      - LD_LIBRARY_PAT est mis à jour avec les chemins de version CUDA :

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

  - Version NCCL compilée :
    - Pour le répertoire CUDA de 12.8, compilé la version NCCL 2.26.2+ .8 CUDA12
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda/targets/x86_64-linux/lib
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
- Installateur EFA : 1.40.0

- Nvidia GDRCopy : 2,5
- Moteur Nvidia Transformer : 1.11.0
- AWS NCCL OFI : 1.14.2-aws
  - Le chemin d'installation :/opt/amazon/ofi-nccl/. Path /opt/amazon/ofi-nccl/libest ajouté à LD_LIBRARY_PATH
- AWS CLI v2 à/usr/local/bin/aws
- Type de volume EBS : GP3
- Boîte à outils pour conteneurs Nvidia : 1.17.7
  - Commande de version : nvidia-container-cli -V
- Docker : 28,2.2
- Python :/ usr/bin/python 3,12
- Requête AMI-ID avec le paramètre SSM (exemple de région : 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
```

- Requête AMI-ID avec AWSCLI (exemple de région : 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
```

Avis

Attention instantanée

- Flash attention n'a pas encore de version officielle pour la version PyTorch 2.7. Pour cette raison, il est temporairement supprimé de cette AMI. Dès qu'une version officielle de la version PyTorch 2.7 sera publiée, nous l'inclurons dans cette AMI.
- En l'absence de flash, le moteur du transformateur utilise par défaut l'attention fusionnée cuDNN. Il existe actuellement des problèmes connus liés à Fused Attention et Blackwell GPUs, tels que les instances P6-B200.

- « Avec la capacité de calcul sm10.0 (architecture Blackwell) GPUs, le FP8 type de données qui attire l'attention du point par point réduit contient un blocage qui provoque le blocage du noyau dans certaines circonstances, par exemple lorsque la taille du problème est importante ou lorsque le GPU exécute plusieurs noyaux simultanément. Un correctif est prévu pour une future version. » [Notes de mise à jour de cuDNN 9.10.0]
- Pour les utilisateurs qui souhaitent exécuter des instances P6-B200 avec FP8 des données et une attention accrue par point, pensez à installer Flash Attention manuellement.

Instances P6-B200

- Les instances P6-B200 nécessitent la version 12.8 ou supérieure de CUDA et le pilote NVIDIA 570 ou des pilotes plus récents.
- Le P6-B200 contient 8 cartes d'interface réseau et peut être lancé à l'aide de la commande CLI AWS suivante :

```
aws ec2 run-instances --region $REGION \
    --instance-type $INSTANCETYPE \
    --image-id $AMI --key-name $KEYNAME \
    --iam-instance-profile "Name=dlami-builder" \
    --tag-specifications "ResourceType=instanace,Tags=[{Key=Name,Value=$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"
```

Instances P5/P5e

- DeviceIndex est unique à chacun NetworkCard et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1. Vous trouverez ci-dessous un exemple de commande de lancement d'instance EC2 P5 utilisant awscli, s'affichant NetworkCardIndex entre 0 et DeviceIndex 31, 0 pour la première interface et 1 pour les 31 interfaces restantes.

```
aws ec2 run-instances --region $REGION \
    --instance-type $INSTANCETYPE \
    --image-id $AMI --key-name $KEYNAME \
    --iam-instance-profile "Name=dlami-builder" \
    --tag-specifications "ResourceType=instanace,Tags=[{Key=Name,Value=$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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

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

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

```
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
```

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, le support de Conda est PyTorch devenu obsolète (voir l'annonce officielle). Par conséquent, les PyTorch versions 2.6 et supérieures utiliseront les environnements virtuels Python. Pour activer le PyTorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

Date de sortie : 2025-06-03

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Ubuntu 22.04) 20250602

Ajouté

- Version initiale de la série de GPU AMI Deep Learning PyTorch 2.7 (Ubuntu 22.04). Incluant un environnement virtuel Python pytorch (source/opt/pytorch/bin/activate) complété par le pilote NVIDIA R570, CUDA = 12,8, cuDNN = 9,10, NCCL = 2,26,5 et EFA = 1,40,0. PyTorch

Problèmes connus

- « Avec la capacité de calcul sm10.0 (architecture Blackwell) GPUs, le FP8 type de données qui attire l'attention du point par point réduit contient un blocage qui provoque le blocage du noyau dans certaines circonstances, par exemple lorsque la taille du problème est importante ou lorsque le GPU exécute plusieurs noyaux simultanément. Un correctif est prévu pour une future version. » [Notes de mise à jour de cuDNN 9.10.0]
  - Pour les utilisateurs qui souhaitent exécuter des instances P6-B200 avec FP8 des données et une attention accrue par point, pensez à installer Flash Attention manuellement.

AWS GPU AMI PyTorch 2.6 pour apprentissage profond (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS NVIDIA Driver AMI GPU PyTorch 2.6.0 (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge :

- Reportez-vous à la section Modifications importantes apportées au DLAMI
- Deep Learning avec OSS Le pilote NVIDIA est compatible avec G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : EC2

- Système d'exploitation : Amazon Linux 2023
- Architecture de calcul : x86
- Stack NVIDIA CUDA12 1.6 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.6/ usr/local/cuda
  - CUDA par défaut : 12,6
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.6/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL compilée pour 12.6 : 2.24.3
- Lieu des tests du NCCL :
  - all_reduce, all_gather et reduce_scatter :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - Pour exécuter des tests NCCL, LD_LIBRARY_PATH est déjà mis à jour avec les chemins nécessaires.
    - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :
      - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
    - LD_LIBRARY_PATH est mis à jour avec les chemins de version CUDA
      - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- Installateur EFA : 1.38.0
- Nvidia GDRCopy : 2,4.1
- AWS NCCL OFI : 1.13.2-aws
  - AWS OFI NCCL prend désormais en charge plusieurs versions NCCL avec une seule version
  - Le chemin d'installation :/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
- Version de Python : 3.12
- Python :/opt/pytorch/bin/python
- Pilote NVIDIA : 570.86.15
- AWS CLI v2 à/usr/bin/aws
- Type de volume EBS : GP3

- NVMe Emplacement du magasin d'instances (sur les EC2 instances prises en charge) :/opt/dlami/nvme
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Avis

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, le support de Conda est PyTorch devenu obsolète (voir l'annonce officielle). Par conséquent, les PyTorch versions 2.6 et supérieures utiliseront les environnements virtuels Python. Pour activer le PyTorch venv, veuillez utiliser les instances opt/pytorch/bin/activate source/P5/P5e :

- DeviceIndex est unique à chacun NetworkCard et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1. Vous trouverez ci-dessous un exemple de commande de lancement d'instance EC2 P5 utilisant awscli, affiché NetworkCardIndex du numéro 0 à 31 et DeviceIndex sous la forme 0 pour la première interface et DeviceIndex sous la forme 1 pour les interfaces 31 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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
sudo dnf versionlock kernel*
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

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

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-02-21

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6.0 (Amazon Linux 2023) 20250220

Ajouté

- Version initiale du Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6 pour Amazon Linux 2023

- Depuis la version PyTorch 2.6, Pytorch n'est plus compatible avec Conda. Par conséquent, Pytorch 2.6 et versions ultérieures utiliseront les environnements virtuels Python. Pour activer le pytorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

AWS GPU AMI d'apprentissage profond PyTorch 2.6 (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6. $ {PATCH-VERSION} (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.
- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture de calcul : x86
- Python :/opt/pytorch/bin/python
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 570.86.15
- Stack NVIDIA CUDA12 8.1 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.6/ usr/local/cuda
  - CUDA par défaut : 12,6
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.6/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib

        - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
    - Version NCCL du système compilé présente à l'adresse/usr/local/cuda/: 2.24.3
    - PyTorch Version NCCL compilée à partir de l'environnement PyTorch conda : 2.21.5
- Lieu des tests NCCL :
    - all_reduce, all_gather et reduce_scatter :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
    - Pour exécuter les tests NCCL, LD_LIBRARY_PATH est déjà mis à jour avec les chemins nécessaires.
    - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :
    - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
    - LD_LIBRARY_PATH est mis à jour avec les chemins de version CUDA
    - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- Installateur EFA : 1.38.0
- Nvidia GDRCopy : 2,4.1
- Moteur Nvidia Transformer : v1.11.0
- AWS NCCL OFI : 1.13.2-aws
    - Le chemin d'installation :/opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
    - Remarque : le PyTorch package est également livré avec un plugin AWS OFI NCCL lié dynamiquement en tant que aws-ofi-nccl-dlc package conda et PyTorch utilisera ce package au lieu du système AWS OFI NCCL.
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Version de Python : 3.11
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
    - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :

- Pilote 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
```

Avis

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, Pytorch a abandonné le support de Conda (voir l'annonce officielle). Par conséquent, Pytorch 2.6 et versions ultérieures utiliseront les environnements virtuels Python. Pour activer le pytorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

Instances P5/P5e :

- DeviceIndex est unique à chacun NetworkCard et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1. Vous trouverez ci-dessous un exemple de commande de lancement d'instance EC2 P5 utilisant awscli, affiché NetworkCardIndex du numéro 0 à 31, 0 pour la première interface et 1 pour DeviceIndex les interfaces 31 restantes. DeviceIndex

```
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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
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
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

```
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
```

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-02-21

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6.0 (Ubuntu 22.04) 20250220

Ajouté

- Version initiale de la série de GPU AMI Deep Learning PyTorch 2.6 (Ubuntu 22.04). Incluant un environnement virtuel Python pytorch (source/opt/pytorch/bin/activate), complété par le pilote NVIDIA R570, CUDA = 12,6, cuDNN = 9,7, NCCL = 2,21,5 et EFA = 1,38,0. PyTorch
  - À partir de la PyTorch version 2.6, Pytorch a abandonné le support de Conda (voir l'annonce officielle). Par conséquent, Pytorch 2.6 et versions ultérieures utiliseront les environnements virtuels Python. Pour activer le pytorch venv, veuillez l'activer en utilisant la source/opt/pytorch/bin/activate

AWS GPU AMI PyTorch 2.5 pour apprentissage profond (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.
- Apprentissage profond avec OSS Le pilote Nvidia est compatible avec G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : EC2
- Système d'exploitation : Amazon Linux 2023
- Architecture informatique : x86
- Stack NVIDIA CUDA12 2.4 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.4/ usr/local/cuda
  - CUDA par défaut : 12,4
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.4/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL compilée pour 12.4 : 2.21.5
- Lieu des tests NCCL :
  - all_reduce, all_gather et reduce_scatter :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - Pour exécuter les tests NCCL, LD_LIBRARY_PATH est déjà mis à jour avec les chemins nécessaires.
    - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :
      - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
  - LD_LIBRARY_PATH est mis à jour avec les chemins de version CUDA
    - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- Installateur EFA : 1.38.0

- Nvidia GDRCopy : 2,4.1
- AWS NCCL OFI : 1.13.2-aws
  - AWS OFI NCCL prend désormais en charge plusieurs versions NCCL avec une seule version
  - Le chemin d'installation :/opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
  - Teste le chemin de la sonnerie, message_transfer :/opt/aws-ofi-nccl/tests
- Version de Python : 3.11
- Python :/opt/conda/envs/pytorch/bin/python
- Pilote NVIDIA : 560.35.03
- AWS CLI v2 à/usr/bin/aws
- Type de volume EBS : GP3
- NVMe Emplacement du magasin d'instances (sur les EC2 instances prises en charge) :/opt/dlami/nvme
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Avis

Instances P5/P5e :

- DeviceIndex est unique à chacun NetworkCard et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1. Vous trouverez ci-dessous un exemple de commande de lancement d'instance EC2 P5 utilisant awscli, affiché NetworkCardIndex du numéro 0 à 31, 0 pour la première interface et 1 pour DeviceIndex les interfaces 31 restantes. DeviceIndex

```
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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
sudo dnf versionlock kernel*
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

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

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) 20250216

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-08

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) 20250107

Ajouté

- Support ajouté pour les instances G4dn.

Date de sortie : 2024-11-21

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) 20241120

Ajouté

- Première version du Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5 pour Amazon Linux 2023

Problèmes connus

- Ce DLAMI ne prend pas en charge les instances G4dn et EC2 G5 pour le moment. AWS est conscient d'une incompatibilité susceptible d'entraîner des échecs d'initialisation de CUDA, affectant à la fois les familles d'instances G4dn et G5 lors de l'utilisation des pilotes NVIDIA open source avec un noyau Linux version 6.1 ou ultérieure. Ce problème concerne les distributions Linux telles qu'Amazon Linux 2023, Ubuntu 22.04 ou version ultérieure, ou SUSE Linux Enterprise Server 15 SP6 ou version ultérieure, entre autres.

AWS GPU AMI PyTorch 2.5 pour apprentissage profond (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5. $ {PATCH_VERSION} (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.
- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture de calcul : x86
- Python :/opt/conda/envs/pytorch/bin/python
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 550.144.03

- Stack NVIDIA CUDA12 2.4 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.4/ usr/local/cuda
  - CUDA par défaut : 12,4
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.4/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL du système compilé présente à l'adresse/usr/local/cuda/: 2.21.5
  - PyTorch Version NCCL compilée à partir de l'environnement PyTorch conda : 2.21.5
- Lieu des tests du NCCL :
  - all_reduce, all_gather et reduce_scatter :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - Pour exécuter des tests NCCL, LD_LIBRARY_PATH est déjà mis à jour avec les chemins nécessaires.
    - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :
      - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
  - LD_LIBRARY_PATH est mis à jour avec les chemins de version CUDA
    - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- Installateur EFA : 1.34.0
- Nvidia GDRCopy : 2,4.1
- Moteur Nvidia Transformer : v1.11.0
- AWS NCCL OFI : 1.11.0-aws
  - Le chemin d'installation :/opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
  - Teste le chemin de la sonnerie, message_transfer :/opt/aws-ofi-nccl/tests
  - Remarque : le PyTorch package est également livré avec un plugin AWS OFI NCCL lié dynamiquement en tant que aws-ofi-nccl-dlc package conda et PyTorch utilisera ce package au lieu du système AWS OFI NCCL.
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Version de Python : 3.11

- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Avis

Instances P5/P5e :

- DeviceIndex est unique à chacun NetworkCard et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1. Vous trouverez ci-dessous un exemple de commande de lancement d'instance EC2 P5 utilisant awscli, affiché NetworkCardIndex du numéro 0 à 31 et DeviceIndex sous la forme 0 pour la première interface et DeviceIndex sous la forme 1 pour les interfaces 31 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"
```

Noyau

- La version du noyau est épinglée à l'aide de la commande :

```
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
```

- Nous recommandons aux utilisateurs d'éviter de mettre à jour la version de leur noyau (sauf en cas de correctif de sécurité) afin de garantir la compatibilité avec les pilotes installés et les versions de package. Si les utilisateurs souhaitent toujours effectuer la mise à jour, ils peuvent exécuter les commandes suivantes pour déconnecter leur version du noyau :

```
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
```

- Pour chaque nouvelle version de DLAMI, le dernier noyau compatible disponible est utilisé.

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250216

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-21

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250119

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025.

Date de sortie : 2024-11-21

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20241121

Ajouté

- Version initiale de la série de GPU AMI Deep Learning PyTorch 2.4.1 (Ubuntu 22.04). Y compris un environnement Conda Pytorch complété par le pilote NVIDIA R550, CUDA=12.4.1, CUDNN=8.9.7, NCCL=2.21.5 et EFA=1.37.0. PyTorch

Fixe

- En raison d'une modification apportée au noyau Ubuntu pour corriger un défaut de la fonctionnalité KASLR (Kernel Address Space Layout Randomization), les instances G4Dn/G5 ne sont pas en mesure d'initialiser correctement CUDA sur le pilote OSS Nvidia. Afin d'atténuer ce problème, ce DLAMI inclut une fonctionnalité qui charge dynamiquement le pilote propriétaire pour les instances G4Dn et G5. Veuillez prévoir une brève période d'initialisation pour ce chargement afin de garantir le bon fonctionnement de vos instances.
  - Pour vérifier l'état et l'état de santé de ce service, vous pouvez utiliser les commandes suivantes :

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

```
active
```

AWS GPU AMI PyTorch 2.4 pour apprentissage profond (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4. $ {PATCH_VERSION} (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.
- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture de calcul : x86
- Python :/opt/conda/envs/pytorch/bin/python
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 550.144.03
- Stack NVIDIA CUDA12 8.1 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.4/ usr/local/cuda
  - CUDA par défaut : 12,4
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.4/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL du système compilé présente à l'adresse/usr/local/cuda/: 2.21.5
  - PyTorch Version NCCL compilée à partir de l'environnement PyTorch conda : 2.20.5

- Lieu des tests du NCCL :
  - all_reduce, all_gather et reduce_scatter :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - Pour exécuter des tests NCCL, LD_LIBRARY_PATH est déjà mis à jour avec les chemins nécessaires.
    - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :
      - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
  - LD_LIBRARY_PATH est mis à jour avec les chemins de version CUDA
    - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- Installateur EFA : 1.34.0
- Nvidia GDRCopy : 2,4.1
- Moteur Nvidia Transformer : v1.11.0
- AWS NCCL OFI : 1.11.0-aws
  - Le chemin d'installation :/opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/libest ajouté à LD_LIBRARY_PATH.
  - Teste le chemin de la sonnerie, message_transfer :/opt/aws-ofi-nccl/tests
  - Remarque : le PyTorch package est également livré avec un plugin AWS OFI NCCL lié dynamiquement en tant que aws-ofi-nccl-dlc package conda et PyTorch utilisera ce package au lieu du système AWS OFI NCCL.
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Version de Python : 3.11
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Avis

Instances P5/P5e

- DeviceIndex est unique à chacun NetworkCard et doit être un entier non négatif inférieur à la limite de ENIs par. NetworkCard Sur P5, le nombre de ENIs par NetworkCard est 2, ce qui signifie que les seules valeurs valides pour DeviceIndex sont 0 ou 1. Vous trouverez ci-dessous un exemple de commande de lancement d'instance EC2 P5 utilisant awscli, affiché NetworkCardIndex du numéro 0 à 31, 0 pour la première interface et 1 pour DeviceIndex les interfaces 31 restantes. DeviceIndex

```
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"
```

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20250216

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4

- Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
- Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Date de sortie : 2025-01-21

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20250119

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 afin de remédier aux problèmes CVE présents dans le bulletin de sécurité des pilotes d'affichage pour GPU NVIDIA de janvier 2025.

Date de sortie : 2024-11-18

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20241116

Fixe

- En raison d'une modification apportée au noyau Ubuntu pour corriger un défaut de la fonctionnalité KASLR (Kernel Address Space Layout Randomization), les instances G4Dn/G5 ne sont pas en mesure d'initialiser correctement CUDA sur le pilote OSS Nvidia. Afin d'atténuer ce problème, ce DLAMI inclut une fonctionnalité qui charge dynamiquement le pilote propriétaire pour les instances G4Dn et G5. Veuillez prévoir une brève période d'initialisation pour ce chargement afin de garantir le bon fonctionnement de vos instances.
  - Pour vérifier l'état et l'état de santé de ce service, vous pouvez utiliser les commandes suivantes :

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

Date de sortie : 2024-10-16

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20241016

Ajouté

- Ajout de la TransformerEngine version Nvidia v1.11.0 pour accélérer les modèles Transformer (pour plus de détails, veuillez consulter https://docs.nvidia.com/deeplearning/transformer-.html) engine/user-guide/index

Date de sortie : 2024-09-30

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20240929

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.16.1 à la version 1.16.2, corrigeant la vulnérabilité de sécurité CVE-2024-0133.

Date de sortie : 2024-09-26

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20240925

Ajouté

- Version initiale de la série de GPU AMI Deep Learning PyTorch 2.4.1 (Ubuntu 22.04). Y compris un environnement Conda Pytorch complété par le pilote NVIDIA R550, CUDA=12.4.1, CUDNN=8.9.7, NCCL=2.20,5 et EFA=1.34.0. PyTorch

ARM64 PyTorch Notes de mise à jour du DLAMI

Vous trouverez ci-dessous les notes de publication pour ARM64 PyTorch DLAMIs :

GPU

- AWS GPU ARM64 AMI PyTorch 2.7 pour apprentissage profond (Amazon Linux 2023)
- AWS GPU ARM64 AMI d'apprentissage profond PyTorch 2.7 (Ubuntu 22.04)
- AWS GPU ARM64 AMI PyTorch 2.6 pour apprentissage profond (Amazon Linux 2023)
- AWS GPU ARM64 AMI d'apprentissage profond PyTorch 2.6 (Ubuntu 22.04)

- AWS GPU ARM64 AMI PyTorch 2.5 pour apprentissage profond (Ubuntu 22.04)
- AWS GPU ARM64 AMI PyTorch 2.4 pour apprentissage profond (Ubuntu 22.04)

AWS Neurone

- Reportez-vous au guide de l'utilisateur du DLAMI Neuron

AWS Processeur graphique ARM64 AMI OSS PyTorch 2.7 pour le Deep Learning (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Pilote graphique Nvidia OSS d' ARM64 AMI Deep Learning PyTorch 2.7 (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2023
- Architecture informatique : ARM64
- Noyau Linux : 6.12
- Pilote NVIDIA : 570.133.20
- Stack NVIDIA CUDA 12.8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-12.8/ usr/local/cuda
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/- usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa

    - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
- AWS CLI v2 à/usr/local/bin/aws
- Type de volume EBS : GP3
- Boîte à outils pour conteneurs Nvidia : 1.17.7
  - Commande de version : nvidia-container-cli -V
- Docker : 25,0,8
- Python :/ usr/bin/python 3,12
- Requête AMI-ID avec le paramètre SSM (exemple de région : 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
```

- Requête AMI-ID avec AWSCLI (exemple de région : 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
```

Avis

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, le support de Conda est PyTorch devenu obsolète (voir l'annonce officielle). Par conséquent, les PyTorch versions 2.6 et supérieures utiliseront les environnements virtuels Python. Pour activer le PyTorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

Date de sortie : 2025-05-22

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Amazon Linux 2023) 20250521

Ajouté

- Version initiale de la série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Amazon Linux 2023). Incluant un environnement virtuel Python pytorch (source/opt/pytorch/bin/

activate) complété par le pilote NVIDIA R570, CUDA = 12,8, cuDNN = 9.10 et NCCL = 2.26.2 PyTorch

AWS GPU OSS avec ARM64 AMI d'apprentissage profond PyTorch 2.7 (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- ARM64 AMI d'apprentissage profond OSS Nvidia Driver GPU PyTorch 2.7 (Ubuntu 22.04) ${YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture informatique : ARM64
- Noyau Linux : 6.8
- Pilote NVIDIA : 570.133.20
- Stack NVIDIA CUDA 12.8 :
  - Répertoires d'installation CUDA, NCCL et CudDN :/-12.8/ usr/local/cuda
  - CUDA par défaut : 12,8
    - PATH//usr/local/cudapointe vers CUDA 12.8
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/- usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
- AWS CLI v2 à/usr/local/bin/aws
- Type de volume EBS : GP3
- Boîte à outils pour conteneurs Nvidia : 1.17.7

  - Commande de version : nvidia-container-cli -V
- Docker : 28,2.2
- Python :/ usr/bin/python 3,12
- Requête AMI-ID avec le paramètre SSM (exemple de région : 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
```

- Requête AMI-ID avec AWSCLI (exemple de région : 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
```

Avis

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, le support de Conda est PyTorch devenu obsolète (voir l'annonce officielle). Par conséquent, les PyTorch versions 2.6 et supérieures utiliseront les environnements virtuels Python. Pour activer le PyTorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

Date de sortie : 2025-05-22

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Ubuntu 22.04) 20250521

Ajouté

- Version initiale de la série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Ubuntu 22.04). Incluant un environnement virtuel Python pytorch (source/opt/pytorch/bin/activate) complété par le pilote NVIDIA R570, CUDA = 12,8, cuDNN = 9.10 et NCCL = 2.26.2 PyTorch

AWS GPU ARM64 AMI PyTorch 2.6 pour apprentissage profond (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6. $ {PATCH-VERSION} (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : EC2
- Système d'exploitation : Amazon Linux 2023
- Architecture informatique : ARM64
- Python :/opt/pytorch/bin/python
- Version de Python : 3.12
- Pilote NVIDIA :
  - pilote OSS Nvidia : 570.86.15
- Stack NVIDIA CUDA12 1.6 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.6/ usr/local/cuda
  - CUDA par défaut : 12,6
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.6/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL du système compilé présente à l'adresse/usr/local/cuda/: 2.24.3
  - PyTorch Version NCCL compilée à partir de l'environnement PyTorch conda : 2.21.5
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3

- Requête AMI-ID avec le paramètre SSM (exemple : la région est 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est 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
```

Avis

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, le support de Conda est PyTorch devenu obsolète (voir l'annonce officielle). Par conséquent, les PyTorch versions 2.6 et supérieures utiliseront les environnements virtuels Python. Pour activer le PyTorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

Date de sortie : 2025-02-21

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Amazon Linux 2023) 20250221

Ajouté

- Version initiale de la série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Amazon Linux 2023). Incluant un environnement virtuel Python pytorch (source/opt/pytorch/bin/activate/), complété par le pilote NVIDIA R570, CUDA = 12,6, cuDNN = 9,7, NCCL = 2,21,5. PyTorch

AWS GPU ARM64 AMI d'apprentissage profond PyTorch 2.6 (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning ARM64 AMI OSS NVIDIA Driver GPU PyTorch 2.6. $ {PATCH-VERSION} (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture informatique : ARM64
- Python :/opt/pytorch/bin/python
- Version de Python : 3.12
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 570.86.15
- Stack NVIDIA CUDA12 1.6 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.6/ usr/local/cuda
  - CUDA par défaut : 12,6
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.6/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL du système compilé présente à l'adresse/usr/local/cuda/: 2.24.3
  - PyTorch Version NCCL compilée à partir de l'environnement PyTorch venv : 2.21.5
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple de région : 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
```

- Requête AMI-ID avec AWSCLI (exemple de région : 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
```

Avis

PyTorch Obsolète d'Anaconda Channel

À partir de la PyTorch version 2.6, le support de Conda est PyTorch devenu obsolète (voir l'annonce officielle). Par conséquent, les PyTorch versions 2.6 et supérieures utiliseront les environnements virtuels Python. Pour activer le PyTorch venv, veuillez utiliser la source/opt/pytorch/bin/activate

Date de sortie : 2025-02-21

Nom de l'AMI : Deep Learning ARM64 AMI OSS NVIDIA Driver GPU PyTorch 2.6.0 (Ubuntu 22.04) 20250221

Ajouté

- Version initiale de la série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Ubuntu 22.04). Incluant un environnement virtuel Python pytorch (source/opt/pytorch/bin/activate), complété par le pilote NVIDIA R570, CUDA = 12,6, cuDNN = 9,7, NCCL = 2,21,5. PyTorch

AWS GPU ARM64 AMI PyTorch 2.5 pour apprentissage profond (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- ARM64 AMI d'apprentissage profond OSS Nvidia Driver GPU PyTorch 2.5. $ {PATCH-VERSION} (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture informatique : ARM64
- Python :/opt/conda/envs/pytorch/bin/python
- Version de Python : 3.11
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 550.144.03
- Stack NVIDIA CUDA12 2.4 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.4/ usr/local/cuda
  - CUDA par défaut : 12,4
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.4/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL du système compilé présente à l'adresse/usr/local/cuda/: 2.21.5
  - PyTorch Version NCCL compilée à partir de l'environnement PyTorch conda : 2.21.5
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple : la région est 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est 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
```

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250215

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-21

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250117

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025.

Date de sortie : 2024-11-22

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04) 20241122

Ajouté

- Version initiale de la série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04). Incluant un environnement Conda PyTorch complété par le pilote NVIDIA R550, CUDA = 12,4, CUDNN = 8.9.7, NCCL = 2.21,5. PyTorch

AWS GPU ARM64 AMI PyTorch 2.4 pour apprentissage profond (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4. $ {PATCH-VERSION} (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- G5g

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture informatique : ARM64
- Python :/opt/conda/envs/pytorch/bin/python
- Version de Python : 3.11

- Pilote NVIDIA :
  - Pilote OSS Nvidia : 550.144.03
- Stack NVIDIA CUDA12 8.1 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.4/ usr/local/cuda
  - CUDA par défaut : 12,4
    - CHEMIN/usr/local/cuda points to /usr/local/cuda-12.4/
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH doit avoir/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
      - CHEMIN à avoir/usr/local/cuda/bin/:/usr/local/cuda/include/
  - Version NCCL du système compilé présente à l'adresse/usr/local/cuda/: 2.21.5
  - PyTorch Version NCCL compilée à partir de l'environnement PyTorch conda : 2.20.5
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple : la région est 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est 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
```

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) 20250215

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-21

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) 20250117

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025.

Date de sortie : 2024-09-30

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) 20240927

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.16.1 à la version 1.16.2, corrigeant la vulnérabilité de sécurité CVE-2024-0133.

Date de sortie : 2024-09-26

Nom de l'AMI : Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) 20240926

Ajouté

- Version initiale de la série Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04). Incluant un environnement Conda PyTorch complété par le pilote NVIDIA R550, CUDA = 12,4, CUDNN = 8.9.7, NCCL = 2.20,5. PyTorch

TensorFlow DLAMIs

TensorFlow Notes de mise à jour du DLAMI

- Notes de mise TensorFlow à jour du DLAMI X86

Notes de mise TensorFlow à jour du DLAMI X86

Vous trouverez ci-dessous les notes de mise à jour du TensorFlow DLAMI X86 :

GPU

- AWS Processeur graphique AMI TensorFlow 2.18 pour apprentissage profond (Amazon Linux 2023)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.18 (Ubuntu 22.04)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.17 (Ubuntu 22.04)

AWS Neurone

- Reportez-vous au guide DLAMI de Neuron.

Processeur graphique AMI TensorFlow 2.18 pour apprentissage profond (Amazon Linux 2023)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Apprentissage profond avec OSS Le pilote Nvidia est compatible avec G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Amazon Linux 2023
- Architecture de calcul : x86
- Python :/ opt/tensorflow/bin/python 3,12
- TensorFlow version : 2.18
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 560.35.03
- CUDA12 Stack NVIDIA :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.5/ usr/local/cuda
- Installateur EFA : 1.37.0
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023) 20241215

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2024-12-09

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023) 20241206

Ajouté

- Version initiale de la série de processeurs graphiques Deep Learning OSS Nvidia Driver AMI TensorFlow 2.18 (Amazon Linux 2023).
  - Le logiciel inclut les éléments suivants :
    - « Nvidia-driver = 560,35,03 »
    - « fabric-manager=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 »
- L'environnement virtuel Tensorflow (source de commande d'activation/opt/tensorflow/bin/activate) inclut les éléments suivants :
    - « tensorflow=2,18,0 »

GPU AMI d'apprentissage profond TensorFlow 2.18 (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en.

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture informatique : x86
- Python :/ opt/tensorflow/bin/python 3,12
- TensorFlow version : 2.18
- Pilote NVIDIA :
    - Pilote OSS Nvidia : 550.144.03
- CUDA12 Stack NVIDIA :
    - Chemin d'installation de CUDA, NCCL et CudDN :/-12.5/ usr/local/cuda
- Installateur EFA : 1.37.0
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws

- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04) 20250215

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-20

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04) 20250118

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.90.07 à la version 550.127.05 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025

Date de sortie : 2024-12-09

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04) 20241206

Ajouté

- Version initiale de la série Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04).
  - Le logiciel inclut les éléments suivants :
    - « nvidia-driver=550,127,05 »
    - « fabric-manager=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 »
- L'environnement virtuel Tensorflow (source de commande d'activation/opt/tensorflow/bin/activate) inclut les éléments suivants :
  - « tensorflow=2,18,0 »

Fixe

- En raison d'une modification apportée au noyau Ubuntu pour corriger un défaut de la fonctionnalité KASLR (Kernel Address Space Layout Randomization), les instances G4Dn/G5 ne sont pas en mesure d'initialiser correctement CUDA sur le pilote OSS Nvidia. Afin d'atténuer ce problème, ce DLAMI inclut une fonctionnalité qui charge dynamiquement le pilote propriétaire pour les instances G4Dn et G5. Veuillez prévoir une brève période d'initialisation pour ce chargement afin de garantir le bon fonctionnement de vos instances.
  - Pour vérifier l'état et l'intégrité de ce service, vous pouvez utiliser les commandes suivantes :

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

GPU AMI d'apprentissage profond TensorFlow 2.17 (Ubuntu 22.04)

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Instances prises en charge

- Apprentissage profond avec OSS Le pilote Nvidia prend en charge les modèles G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e. Stylo 5.

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2
- Système d'exploitation : Ubuntu 22.04
- Architecture de calcul : x86
- Python :/ opt/tensorflow/bin/python 3,12
- TensorFlow version : 2.17
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 550.144.03
- CUDA12 Stack NVIDIA :

  - Chemin d'installation de CUDA, NCCL et CudDN :/-12.3/ usr/local/cuda
- Installateur EFA : 1.34.0
- AWS CLI v2 en tant qu'aws2 et AWS CLI v1 en tant qu'aws
- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple : la région est us-east-1) :
  - Pilote 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
```

- Interrogez l'AMI-ID avec AWSCLI (par exemple, la région est us-east-1) :
  - Pilote 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
```

Date de sortie : 2025-02-17

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20250215

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, veillez à mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel Si vous utilisez une couche de compatibilité CUDA.

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-01-20

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20250118

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025

La version 2.17.1

Date de sortie : 2024-11-18

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20241115

Fixe

- En raison d'une modification apportée au noyau Ubuntu pour corriger un défaut de la fonctionnalité KASLR (Kernel Address Space Layout Randomization), les instances G4Dn/G5 ne sont pas en mesure d'initialiser correctement CUDA sur le pilote OSS Nvidia. Afin d'atténuer ce problème, ce DLAMI inclut une fonctionnalité qui charge dynamiquement le pilote propriétaire pour les instances G4Dn et G5. Veuillez prévoir une brève période d'initialisation pour ce chargement afin de garantir le bon fonctionnement de vos instances.
  - Pour vérifier l'état et l'état de santé de ce service, vous pouvez utiliser les commandes suivantes :

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

La version 2.17.0

Date de sortie : 2024-09-25

Nom de l'AMI : Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20240924

Ajouté

- Version initiale de la série Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04).
  - Le logiciel inclut les éléments suivants :
    - « Nvidia-driver=550,90,07 »
    - « fabric-manager=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 »
- L'environnement virtuel Tensorflow (source de commande d'activation/opt/tensorflow/bin/activate) inclut les éléments suivants :
  - « tensorflow=2,17.0 »

Notes de mise à jour pour Multi-Framework DLAMIs

> **Tip**
>
> Si vous n'utilisez qu'un seul framework d'apprentissage automatique, nous vous recommandons d'utiliser un DLAMI à structure unique.

Notes de mise à jour du DLAMI Multi Framework

- Notes de mise à jour du DLAMI multi-framework

Notes de mise à jour du DLAMI multi-framework

Vous trouverez ci-dessous les notes de mise à jour du DLAMI X86 Multi-Framework :

GPU

- AWS AMI d'apprentissage profond (Amazon Linux 2)

AWS Neurone

- Reportez-vous au guide de l'utilisateur du DLAMI Neuron

AWS AMI d'apprentissage profond (Amazon Linux 2)

> **Tip**
>
> Les clients utilisant un framework unique comme PyTorch ou TensorFlow sont encouragés à utiliser le framework unique DLAMIs mentionné ici

Pour obtenir de l'aide pour démarrer, consultezCommencer à utiliser le DLAMI.

Format du nom de l'AMI

- Version $ {XX.X} du pilote Nvidia propriétaire pour le Deep Learning (Amazon Linux 2)
- Version $ {XX.X} du pilote Nvidia pour le Deep Learning OSS (Amazon Linux 2)

EC2 Instances prises en charge

- Reportez-vous à la section Modifications importantes apportées au DLAMI.
- Apprentissage profond avec OSS Le pilote Nvidia est compatible avec G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5
- Le Deep Learning avec pilote propriétaire Nvidia prend en charge les formats G3 (G3.16x non pris en charge), P3, P3dn

L'AMI inclut les éléments suivants :

- AWS Service pris en charge : Amazon EC2

- Système d'exploitation : Amazon Linux 2
- Architecture de calcul : x86
- Framework d'environnements Conda et versions python :
  - AMI du pilote Nvidia pour le Deep Learning OSS (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 pilote Nvidia propriétaire pour le Deep Learning (Amazon Linux 2) :
    - python3 : Python 3.10
    - tensorflow2_p310 : 2,16, Python 3,10 TensorFlow
    - pytorch_p310 : 2,2, Python 3,10 PyTorch
- Pilote NVIDIA :
  - Pilote OSS Nvidia : 550.163.01
  - Pilote Nvidia propriétaire : 550.163.01
- Stack NVIDIA CUDA12 1.1-12.4 :
  - Chemin d'installation de CUDA, NCCL et CudDN :/-xx.x/ usr/local/cuda
  - CUDA par défaut : 12.1
    - PATH//usr/local/cudapointe vers CUDA12 1.
    - Mise à jour des variables d'environnement ci-dessous :
      - LD_LIBRARY_PATH à avoir/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
      - CHEMIN à avoir/usr/local/cuda-12.1/bin/:/usr/local/cuda-11.8/include/
    - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
  - Version NCCL compilée pour CUDA 12.1-12.4 : 2.22.3
  - Lieu des tests du NCCL :
    - all_reduce, all_gather et reduce_scatter :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
    - Pour exécuter les tests NCCL, LD_LIBRARY_PATH doit réussir avec les mises à jour ci-dessous.
      - PATHs Des éléments communs sont déjà ajoutés à LD_LIBRARY_PATH :
        - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib

    - Pour toute autre version de CUDA, veuillez mettre à jour LD_LIBRARY_PATH en conséquence.
- Installateur EFA : 1.38.0
- GDRCopy: 2,4
- AWS NFC OFI : 1.13.2
  - Emplacement du système :/usr/local/cuda-xx.x/efa
  - Ceci est ajouté pour exécuter les tests NCCL situés à l'adresse/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - De plus, le PyTorch package est également livré avec un plugin AWS OFI NCCL lié dynamiquement en tant que aws-ofi-nccl-dlc package conda et PyTorch utilisera ce package au lieu du système AWS OFI NCCL.
- Emplacement des tests NCCL :/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
- AWS CLI v2 à/usr/local/bin/aws2 et AWS CLI v1 à/usr/local/bin/aws
- Type de volume EBS : GP3
- Requête AMI-ID avec le paramètre SSM (exemple de région : us-east-1) :
  - Pilote 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
```

  - Pilote Nvidia propriétaire :

```
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
```

- Requête AMI-ID avec AWSCLI (exemple de région : us-east-1) :
  - Pilote 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
```

  - Pilote Nvidia propriétaire :

```
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
```

Avis

Mises à jour EFA de 1.37 à 1.38 (sortie le 05/02/2025)

- EFA intègre désormais le plugin AWS OFI NCCL, qui se trouve désormais dans/-ofi-nccl/. opt/amazon/ofi-nccl rather than the original /opt/aws Si vous mettez à jour votre variable LD_LIBRARY_PATH, assurez-vous de modifier correctement l'emplacement NCCL de votre OFI.

Suppression de l'environnement Neuron Conda

- Le pilote Nvidia propriétaire pour le Deep Learning AMIs publié après le 18 juillet 2024 sera expédié sans les environnements Neuron Conda pour PyTorch et. TensorFlow Veuillez plutôt utiliser le Neuron DLAMIs on the DLAMI Release Notes pour utiliser les environnements neuronaux.

Suppression du package d'audit

- Les DLAMI publiés entre le 26 mars 2024 (2024-03-26) et le 12 avril 2024 (2024-04-12) ont été expédiés sans le package d'audit. Si vous avez besoin de ce package spécifique pour vos besoins de journalisation et de surveillance, veuillez migrer vos flux de travail vers le DLAMI le plus récent afin de les utiliser avec le package d'audit installé.

Horovod

- Horovod est supprimé des environnements conda pytorch_p310 et tensorflow2_p310 actuels sur le DLAMI. Les clients pourront installer les bibliothèques Horovod en suivant les directives Horovod et les installer sur leurs bibliothèques DLAMIs pour leurs tâches de formation distribuées.

Date de sortie : 2025-04-22

Noms des AMI

- Pilote AMI Nvidia OSS pour apprentissage profond (Amazon Linux 2) Version 81.2
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 81.2

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.144.03 à la version 550.163.01 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA d'avril 2025

Date de sortie : 2025-02-17

Noms des AMI

- Pilote AMI Nvidia OSS pour apprentissage profond (Amazon Linux 2) Version 80.6
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 80.4

Mis à jour

- Mise à jour de NVIDIA Container Toolkit de la version 1.17.3 à la version 1.17.4
  - Consultez la page des notes de publication ici pour plus d'informations : https://github.com/NVIDIA/nvidia-container-toolkit/releases/tag/v1.17.4
  - Dans la version 1.17.4 de Container Toolkit, le montage des bibliothèques de compatibilité CUDA est désormais désactivé. Afin de garantir la compatibilité avec plusieurs versions de CUDA sur les flux de travail de conteneurs, assurez-vous de mettre à jour votre LD_LIBRARY_PATH pour inclure vos bibliothèques de compatibilité CUDA, comme indiqué dans le didacticiel « Si vous utilisez une couche de compatibilité CUDA » ici - -gpu-drivers.html# https://docs.aws.amazon.com/sagemaker/ latest/dg/inference collapsible-cuda-compat

Supprimé

- Suppression des bibliothèques d'espace utilisateur cuobj et nvdisasm fournies par le kit d'outils NVIDIA CUDA pour remédier à un problème CVEs présent dans le bulletin de sécurité du kit d'outils NVIDIA CUDA du 18 février 2025

Date de sortie : 2025-02-05

Noms des AMI

- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 80.2
- Pilote AMI Nvidia OSS pour apprentissage profond (Amazon Linux 2) Version 80.4

Mis à jour

- Version EFA mise à niveau de 1.37.0 à 1.38.0
  - EFA intègre désormais le plugin AWS OFI NCCL, qui se trouve désormais dans/-ofi-nccl/. opt/amazon/ofi-nccl rather than the original /opt/aws Si vous mettez à jour votre variable LD_LIBRARY_PATH, assurez-vous de modifier correctement l'emplacement NCCL de votre OFI.

Date de sortie : 2025-01-15

Noms des AMI

- Pilote AMI Nvidia OSS pour apprentissage profond (Amazon Linux 2) Version 80.3
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 80.1

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.127.05 à la version 550.144.03 pour corriger un problème CVEs présent dans le bulletin de sécurité du pilote d'affichage pour GPU NVIDIA de janvier 2025

Date de sortie : 2024-12-09

Noms des AMI

- Pilote AMI Nvidia OSS pour apprentissage profond (Amazon Linux 2) Version 80.1
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 79.9

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.17.0 à la version 1.17.3

Date de sortie : 2024-11-11

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 79.9
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 79.7

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.16.2 à la version 1.17.0, corrigeant la vulnérabilité de sécurité CVE-2024-0134.

Date de sortie : 2024-10-22

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 79.6
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 79.6

Mis à jour

- Mise à niveau du pilote Nvidia de la version 550.90.07 à la version 550.127.05 pour corriger un problème CVEs présent dans le bulletin de sécurité d'affichage des GPU NVIDIA d'octobre 2024

Date de sortie : 2024-10-03

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 79.3
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 79.3

Mis à jour

- Mise à niveau de Nvidia Container Toolkit de la version 1.16.1 à la version 1.16.2, corrigeant la vulnérabilité de sécurité CVE-2024-0133.

Date de sortie : 2024-07-18

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 78.6
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 78.7

Mis à jour

- Suppression des environnements conda aws_neuron_pytorch_p38 et aws_neuron_tensorflow_p38 de l'AMI du pilote Nvidia propriétaire pour le Deep Learning.
- Suppression de la prise en charge de la famille d'instances Inf1 de l'AMI du pilote Nvidia propriétaire pour le Deep Learning.

Date de sortie : 2024-06-06

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 78.5
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 78.5

Mis à jour

- Version du pilote Nvidia mise à jour vers 535.183.01 à partir de 535.161.08

Date de sortie : 2024-05-17

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 78.1
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 78.1

Mis à jour

- Torchserve mis à jour de v0.8.2 à v0.11.0 dans l'environnement pytorch_p310.

Date de sortie : 2024-05-07

Noms des AMI

- Pilote AMI Nvidia OSS pour apprentissage profond (Amazon Linux 2) Version 78.0
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 78.0

Mis à jour

- TensorFlow version mise à jour de 2.15 à 2.16 dans l'environnement tensorflow2_p310.
- Version EFA mise à jour de la version 1.30 à la version 1.32
- Plugin AWS OFI NCCL mis à jour de la version 1.7.4 à la version 1.9.1
- Boîte à outils de conteneurs Nvidia mise à jour de la version 1.13.5 à la version 1.15.0
  - REMARQUE : La version 1.15.0 n'inclut PAS les packages nvidia-docker2 nvidia-container-runtime et nvidia-docker2. Il est recommandé d'utiliser les nvidia-container-toolkit packages directement en suivant la documentation du Nvidia Container Toolkit.

Ajouté

- Ajout d'une pile CUDA12 .3 avec CUDA12 .3, NCCL 2.21.5, cuDNN 8.9.7

Supprimé

- Suppression des piles de CUDA11 0,7, CUDA12 ,0 présentes à +/- 12,0 usr/local/cuda-11.7 and / usr/local/cuda
- Suppression du package nvidia-docker2 et de sa commande nvidia-docker dans le cadre de la mise à jour de la boîte à outils de conteneurs Nvidia de la version 1.13.5 à la version 1.15.0, qui n'inclut PAS les packages nvidia-docker2 et nvidia-docker2. nvidia-container-runtime

Date de sortie : 2024-04-04

Noms des AMI

- Pilote AMI Nvidia OSS pour apprentissage profond (Amazon Linux 2) Version 77.0
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 77.0

Mis à jour

- PyTorch version mise à jour de 2.1 à 2.2 dans l'environnement pytorch_p310.
- Pour le pilote OSS Nvidia DLAMIs, ajout du support des EC2 instances G6 et Gr6. Reportez-vous à la page de sélection des EC2 instances pour plus d'informations.

Date de sortie : 2024-03-29

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 76.8
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 76.9

Mis à jour

- Mise à jour du pilote Nvidia de 535.104.12 à 535.161.08 dans le pilote Nvidia propriétaire et OSS. DLAMIs
- Les nouvelles instances prises en charge pour chaque DLAMI sont les suivantes :
  - Le Deep Learning avec pilote propriétaire Nvidia prend en charge les formats G3 (G3.16x non pris en charge), P3, P3dn, Inf1
  - Apprentissage profond avec OSS Le pilote Nvidia est compatible avec G4dn, G5, P4d, P4de.

Supprimé

- Suppression de la prise en charge des EC2 instances G4dn, G5, G3.16x par le pilote propriétaire Nvidia DLAMI.

La version 76.8

Date de sortie : 2024-03-20

Noms des AMI

- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 76.8

Ajouté

- Ajout d'awscliv2 dans l'AMI en tant que//usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awssur l'AMI propriétaire du pilote Nvidia

La version 76.7

Date de sortie : 2024-03-20

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2), version 76.7

Ajouté

- Ajout de awscliv2 dans l'AMI en tant que/usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awssur l'AMI du pilote OSS Nvidia
- Pilote OSS Nvidia DLAMI mis à jour avec support G4dn et G5. Sur cette base, le support actuel ressemble à ce qui suit :
  - L'AMI de pilote Nvidia propriétaire de Deep Learning Base (Amazon Linux 2) prend en charge les formats P3, P3dn, G3, G5 et G4dn.
  - L'AMI du pilote Nvidia Deep Learning Base OSS (Amazon Linux 2) est compatible avec G4dn, G5, P4, P5.
- Il est recommandé d'utiliser le pilote DLAMIs OSS Nvidia pour G4dn, G5, P4, P5.

La version 76.3

Date de sortie : 2024-02-14

Mis à jour

- Mis à jour TensorFlow de la version 2.13.0 à la version 2.15.0
- EFA mis à jour de 1.29.0 à 1.30.0
- Mise à jour de AWS-OFI-NCCL de 1.7.3-aws à 1.7.4-aws
- Mise à jour du pilote Nvidia vers la version 535.104.12 sur l'AMI de pilote Nvidia propriétaire basée sur le Deep Learning
- Mise à jour du pilote Nvidia vers 535.154.05 sur Deep Learning OSS Nvidia Driver AMI

La version 76.2

Date de sortie : 2024-02-02

Noms des AMI

- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 76.2
- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2), version 76.4

Sécurité

- Version du package runc mise à jour pour utiliser le correctif pour CVE-2024-21626.

La version 76.1

Date de sortie : 2023-12-27

Mis à jour

- Mise à jour PyTorch de 2.0.1 à 2.1.0

La version 75.1

Date de sortie : 2023-11-17

Reportez-vous à la section Modifications importantes apportées au DLAMI

Noms des AMI

- Pilote AMI Nvidia pour le Deep Learning OSS (Amazon Linux 2) Version 75.1
- Pilote AMI propriétaire Nvidia pour le Deep Learning (Amazon Linux 2), version 75.1

Ajouté

- AWS L'AMI d'apprentissage profond (DLAMI) est divisée en deux groupes distincts :
  - DLAMI utilisant le pilote propriétaire Nvidia (compatible avec P3, P3dn, G3, G5, G4dn).
  - DLAMI qui utilise le pilote Nvidia OSS pour activer EFA (pour prendre en charge les formats P4, P5).
- Veuillez vous référer à l'annonce publique pour plus d'informations sur la division du DLAMI.

- AWS les requêtes cli ci-dessus se trouvent dans les notes de publication sous bullet point Query AMI-ID with AWSCLI (exemple, la région est us-east-1)

Mis à jour

- EFA mis à jour de 1.26.1 à 1.29.0
- GDRCopy mis à jour de 2.3 à 2.4

La version 74.4

Date de sortie : 2023-10-27

Mis à jour

- AWS Plugin OFI NCCL mis à jour de la version 1.7.2 à la version 1.7.3
- Répertoires CUDA 12.0-12.1 mis à jour avec la version 2.18.5 de NCCL
- CUDA12.1 mise à jour en tant que version CUDA par défaut
  - LD_LIBRARY_PATH a été mis à jour pour avoir//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
  - Pour les clients qui souhaitent passer à une autre version de CUDA, veuillez définir les variables LD_LIBRARY_PATH et PATH en conséquence.
- Pillow mis à jour de la version 9.4.0 à la version 10.1.0 pour corriger SNYK-PYTHON-PILLOW-5918878 dans tous les environnements Conda
- Mise à jour d'opencv-python de 4.8.0.74 à 4.8.1.78 pour corriger SNYK-PYTHON-OPENCVPYTHON-5926695 dans tous les environnements conda

Ajouté

- Kernel Live Patching est désormais activé. Les correctifs en temps réel permettent aux clients d'appliquer des correctifs de failles de sécurité et de bogues critiques à un noyau Linux en cours d'exécution, sans redémarrage ni interruption de l'exécution des applications.
  - Veuillez noter que la prise en charge des correctifs en direct pour le noyau 5.10.192 prendra fin le 30 novembre 23.
  - Pour plus d'informations, veuillez consulter les AWS documents officiels ici - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/al2-live-patching.html

La version 74.0

Date de sortie : 2023-07-19

Mis à jour

- Mis à jour TensorFlow de la version 2.12 à la version 2.13
  - Horovod a été supprimé de l'environnement conda dans cette version. Voir la notice pour plus de détails sur l'installation d'horovod.

La version 73.1

Date de sortie : 2023-06-12

Mis à jour

- Mise à jour PyTorch de la version 2.0.0 à la version 2.0.1

Commencer à utiliser le DLAMI

Ce guide contient des conseils pour choisir le DLAMI qui vous convient, sélectionner un type d'instance adapté à votre cas d'utilisation et à votre budget, Informations connexes sur DLAMI et décrit les configurations personnalisées susceptibles de vous intéresser.

Si vous utilisez AWS ou utilisez Amazon pour la première fois EC2, commencez par leAMI d'apprentissage profond avec Conda. Si vous connaissez Amazon EC2 et d'autres AWS services tels qu'Amazon EMR, Amazon EFS ou Amazon S3, et que vous souhaitez intégrer ces services à des projets nécessitant une formation ou une inférence distribuée, vérifiez si l'un d'entre eux correspond Informations connexes sur DLAMI à votre cas d'utilisation.

Mais d'abord, nous vous recommandons de consulter Choisir un DLAMI pour avoir une idée de ce qui pourrait être le type d'instance le mieux adapté à votre application.

Étape suivante

Choisir un DLAMI

Choisir un DLAMI

Nous proposons une gamme d'options DLAMI, comme indiqué dans les notes de mise à jour du DLAMI GPU. Pour vous aider à sélectionner le DLAMI adapté à votre cas d'utilisation, nous regroupons les images selon le type de matériel ou les fonctionnalités pour lesquelles elles ont été développées. Nos principaux groupes sont les suivants :

- Type de DLAMI : base, structure unique, infrastructure multiple (DLAMI Conda)
- Architecture de calcul : Graviton basé sur x86, basé sur ARM64 AWS
- Type de processeur : GPU, CPU, Inferentia, Trainium
- SDK : CUDA, Neuron AWS
- Système d'exploitation : Amazon Linux, Ubuntu

Les autres rubriques de ce guide vous permettront de mieux vous informer et d'entrer dans plus de détails.

Rubriques

Suivant

AMI d'apprentissage profond avec Conda

Installations de CUDA et liaisons d'infrastructures

Bien que le deep learning soit assez avant-gardiste, chaque framework propose des versions « stables ». Ces versions stables peuvent ne pas fonctionner avec la dernière implémentation et les fonctionnalités CUDA ou cuDNN. Votre cas d'utilisation et les fonctionnalités dont vous avez besoin peuvent vous aider à choisir un framework. En cas de doute, utilisez la dernière AMI de Deep Learning avec Conda. Il contient `pip` des binaires officiels pour tous les frameworks avec CUDA, en utilisant la version la plus récente prise en charge par chaque framework. Si vous souhaitez disposer des dernières versions et personnaliser votre environnement d'apprentissage profond, utilisez l'AMI Deep Learning Base.

Consultez notre guide sur Candidats stables et candidats à la sortie pour obtenir des informations supplémentaires.

Choisissez un DLAMI avec CUDA

Toutes les AMI de base de Deep Learning séries de versions CUDA sont disponibles

Toutes les AMI d'apprentissage profond avec Conda séries de versions CUDA sont disponibles

> **Note**
>
> Nous n'incluons plus les environnements MXNet CNTK, Caffe, Caffe2, Theano, Chainer ou Keras Conda dans le. AWS Apprentissage profond (deep learning) AMIs

Pour les numéros de version spécifiques du framework, consultez le Notes de AMIs mise à jour du Deep Learning

Choisissez ce type de DLAMI ou découvrez-en plus sur les DLAMIs différents types grâce à l'option Next Up.

Choisissez l'une des versions de CUDA et consultez la liste complète de celles DLAMIs qui contiennent cette version dans l'annexe, ou apprenez-en plus sur les différentes versions DLAMIs avec l'option Next Up.

Suivant

AMI de base de Deep Learning

Rubriques connexes

- Pour plus d'informations sur le basculement entre les versions CUDA, consultez le didacticiel Utilisation de l'AMI Deep Learning Base.

AMI de base de Deep Learning

L'AMI Deep Learning Base est comme un canevas vide pour le deep learning. Il est livré avec tout ce dont vous avez besoin jusqu'au moment de l'installation d'un framework particulier, et vous permet de choisir les versions de CUDA.

Pourquoi choisir le DLAMI de base

Ce groupe d'AMI est utile pour les contributeurs de projets qui souhaitent entreprendre un projet d'apprentissage profond et créer le dernier. Il est destiné à quiconque souhaite déployer son propre environnement en étant sûr que les derniers logiciels NVIDIA sont installés et fonctionnent, afin de se concentrer sur le choix d'infrastructures et de versions à installer.

Choisissez ce type de DLAMI ou découvrez-en plus sur les DLAMIs différents types avec l'option Next Up.

Suivant

DLAMI avec Conda

Rubriques connexes

- Utilisation de l'AMI Deep Learning Base

AMI d'apprentissage profond avec Conda

Le DLAMI Conda conda utilise des environnements virtuels, ils sont présents soit en multi-framework, soit en framework unique. DLAMIs Ces environnements sont configurés de manière à séparer les différentes installations de framework et à rationaliser le passage d'un framework à l'autre. C'est un excellent moyen d'apprendre et d'expérimenter avec tous les cadres proposés par le DLAMI. La plupart des utilisateurs trouvent que la nouvelle AMI Deep Learning avec Conda est parfaite pour eux.

Ils sont souvent mis à jour avec les dernières versions des frameworks et disposent des derniers pilotes et logiciels GPU. Ils sont généralement désignés comme « les » AWS Apprentissage profond (deep learning) AMIs dans la plupart des documents. Ils DLAMIs prennent en charge les systèmes d'exploitation Ubuntu 20.04, Ubuntu 22.04, Amazon Linux 2, Amazon Linux 2023. La prise en charge des systèmes d'exploitation dépend du support fourni par le système d'exploitation en amont.

Candidats stables et candidats à la sortie

Le Conda AMIs utilise des binaires optimisés des versions formelles les plus récentes de chaque framework. Il n'est pas prévu de versions candidates et de fonctions expérimentales. Les optimisations dépendent de la prise en charge par le framework de technologies d'accélération telles que le MKL DNN d'Intel, qui accélère l'entraînement et l'inférence sur les types d'instances de processeurs C5 et C4. Les fichiers binaires sont également compilés pour prendre en charge les ensembles d'instructions Intel avancés, notamment AVX, AVX-2, SSE4 .1 et .2. SSE4 Ils accélèrent les opérations vectorielles et à virgule flottante sur les architectures d'UC Intel. De plus, pour les types d'instances GPU, le CUDA et le cuDNN sont mis à jour avec la version prise en charge par la dernière version officielle.

L'AMI Deep Learning avec Conda installe automatiquement la version la plus optimisée du framework pour votre EC2 instance Amazon lors de la première activation du framework. Pour plus d'informations, consultez Utilisation de l'AMI Deep Learning avec Conda.

Si vous souhaitez effectuer une installation à partir des sources, en utilisant des options de construction personnalisées ou optimisées, le AMI de base de Deep Learning s pourrait être une meilleure option pour vous.

Obsolescence de Python 2

La communauté open source Python a officiellement mis fin au support de Python 2 le 1er janvier 2020. La PyTorch communauté TensorFlow et la communauté ont annoncé que les

versions TensorFlow 2.1 et PyTorch 1.4 sont les dernières à supporter Python 2. Les versions précédentes du DLAMI (v26, v25, etc.) contenant des environnements Python 2 Conda sont toujours disponibles. Cependant, nous fournissons des mises à jour des environnements Python 2 Conda sur les versions DLAMI publiées précédemment uniquement si des correctifs de sécurité ont été publiés par la communauté open source pour ces versions. Les versions DLAMI contenant les dernières versions des frameworks PyTorch et ne contiennent pas TensorFlow les environnements Python 2 Conda.

CUDA Support

Les numéros de version spécifiques de CUDA se trouvent dans les notes de mise à jour du DLAMI du GPU.

Suivant

Options d'architecture DLAMI

Rubriques connexes

- Pour un didacticiel sur l'utilisation d'une AMI de Deep Learning avec Conda, consultez le Utilisation de l'AMI Deep Learning avec Conda didacticiel.

Options d'architecture DLAMI

AWS Apprentissage profond (deep learning) AMIss sont proposés avec des architectures Graviton2 basées sur x86 ou AWS ARM64.

Pour plus d'informations sur la prise en main du ARM64 DLAMI GPU, consultez. Le ARM64 DLAMI Pour plus de détails sur les types d'instances disponibles, consultezChoix d'un type d'instance DLAMI.

Suivant

Options du système d'exploitation DLAMI

Options du système d'exploitation DLAMI

DLAMIs sont proposés dans les systèmes d'exploitation suivants.

- Amazon Linux 2
- Amazon Linux 2023

- Ubuntu 20.04
- Ubuntu 22.04

Les anciennes versions des systèmes d'exploitation sont disponibles sur les versions obsolètes DLAMIs. Pour plus d'informations sur la dépréciation du DLAMI, voir Dépréciations du DLAMI

Avant de choisir un DLAMI, évaluez le type d'instance dont vous avez besoin et identifiez votre région. AWS

Suivant

Choix d'un type d'instance DLAMI

Choix d'un type d'instance DLAMI

Plus généralement, tenez compte des points suivants lorsque vous choisissez un type d'instance pour un DLAMI.

- Si vous débutez dans le domaine du deep learning, une instance dotée d'un seul GPU peut répondre à vos besoins.
- Si vous êtes soucieux de votre budget, vous pouvez utiliser des instances utilisant uniquement le processeur.
- Si vous souhaitez optimiser les performances et la rentabilité pour l'inférence de modèles d'apprentissage profond, vous pouvez utiliser des instances avec des puces AWS Inferentia.
- Si vous recherchez une instance de GPU hautes performances avec une architecture de processeur basée sur ARM64, vous pouvez utiliser le type d'instance G5g.
- Si vous souhaitez exécuter un modèle préentraîné pour les inférences et les prédictions, vous pouvez associer une Amazon Elastic Inference à votre instance Amazon. EC2 Amazon Elastic Inference vous donne accès à un accélérateur doté d'une fraction de GPU.
- Pour les services d'inférence à volume élevé, une instance de processeur unique avec beaucoup de mémoire, ou un cluster de telles instances, peut être une meilleure solution.
- Si vous utilisez un modèle volumineux contenant beaucoup de données ou un lot de grande taille, vous avez besoin d'une instance plus grande avec plus de mémoire. Vous pouvez également distribuer votre modèle à un cluster de GPUs. Vous pouvez trouver que l'utilisation d'une instance avec moins de mémoire est une meilleure solution pour vous si vous réduisez la taille de votre lot. Cela peut avoir une incidence sur votre précision et votre vitesse de formation.

- Si vous souhaitez exécuter des applications d'apprentissage automatique à l'aide de la bibliothèque NCCL (NVIDIA Collective Communications Library) nécessitant des niveaux élevés de communications entre nœuds à grande échelle, vous pouvez utiliser Elastic Fabric Adapter (EFA).

Pour plus de détails sur les instances, voir d'instances.

Les rubriques suivantes fournissent des informations sur les considérations relatives aux types d'instance.

> **Important**
>
> Le Deep Learning AMIs inclut les pilotes, les logiciels ou les boîtes à outils développés, détenus ou fournis par NVIDIA Corporation. Vous acceptez d'utiliser ces pilotes, logiciels ou boîtes à outils NVIDIA uniquement sur les EC2 instances Amazon qui incluent du matériel NVIDIA.

Rubriques

Tarification du DLAMI

Les frameworks d'apprentissage profond inclus dans le DLAMI sont gratuits et chacun possède ses propres licences open source. Bien que le logiciel inclus dans le DLAMI soit gratuit, vous devez toujours payer pour le matériel d'instance Amazon EC2 sous-jacent.

Certains types d' EC2 instances Amazon sont étiquetés comme gratuits. Il est possible d'exécuter le DLAMI sur l'une de ces instances gratuites. Cela signifie que l'utilisation du DLAMI est totalement gratuite lorsque vous utilisez uniquement la capacité de cette instance. Si vous avez besoin d'une instance plus puissante avec plus de cœurs de processeur, plus d'espace disque, plus de RAM, ou

une ou plusieurs instances GPUs, vous avez besoin d'une instance qui ne fait pas partie de la classe d'instance de niveau libre.

Pour plus d'informations sur le choix et la tarification des instances, consultez EC2 les tarifs Amazon.

Disponibilité de la région DLAMI

Chaque région prend en charge une gamme différente de types d'instances et, souvent, le coût d'un type d'instance varie légèrement d'une région à l'autre. DLAMIs ne sont pas disponibles dans toutes les régions, mais il est possible de les DLAMIs copier dans la région de votre choix. Consultez Copier une AMI pour plus d'informations. Consultez la liste de sélection des régions et assurez-vous de choisir une région proche de vous ou de vos clients. Si vous prévoyez d'utiliser plusieurs DLAMI et de créer éventuellement un cluster, veillez à utiliser la même région pour tous les nœuds du cluster.

Pour plus d'informations sur les régions, rendez-vous sur Amazon EC2 Service endpoints.

Suivant

Instances GPU recommandées

Instances GPU recommandées

Nous recommandons une instance de GPU pour la plupart des applications de deep learning. L'entraînement de nouveaux modèles est plus rapide sur une instance GPU que sur une instance CPU. Vous pouvez effectuer une mise à l'échelle sous-linéaire lorsque vous avez des instances multi-GPU ou si vous utilisez une formation distribuée sur de nombreuses instances avec. GPUs

Les types d'instances suivants prennent en charge le DLAMI. Pour plus d'informations sur les options de type d'instance GPU et leurs utilisations, consultez instances Types d'instances et sélectionnez Calcul accéléré.

> **Note**
>
> La taille de votre modèle doit être prise en compte dans le choix d'une instance. Si votre modèle dépasse la RAM disponible d'une instance, choisissez un autre type d'instance avec suffisamment de mémoire pour votre application.

- Les instances Amazon EC2 P6 disposent d'un maximum de 8 cartes NVIDIA Blackwell B200. GPUs

- Les instances Amazon EC2 P5e peuvent contenir jusqu'à 8 NVIDIA Tesla GPUs H200.
- Les instances Amazon EC2 P5 peuvent contenir jusqu'à 8 NVIDIA Tesla GPUs H100.
- Les instances Amazon EC2 P4 peuvent contenir jusqu'à 8 NVIDIA Tesla GPUs A100.
- Les instances Amazon EC2 P3 peuvent contenir jusqu'à 8 NVIDIA Tesla GPUs V100.
- Les instances Amazon EC2 G3 peuvent contenir jusqu'à 4 NVIDIA Tesla GPUs M60.
- Les instances Amazon EC2 G4 disposent d'un maximum de 4 cartes NVIDIA GPUs T4.
- Les instances Amazon EC2 G5 disposent d'un maximum de 8 cartes NVIDIA GPUs A10G.
- Les instances Amazon EC2 G6 disposent d'un maximum de 8 cartes NVIDIA GPUs L4.
- Les instances Amazon EC2 G6e possèdent jusqu'à 8 cœurs NVIDIA L40S Tensor. GPUs
- Les instances Amazon EC2 G5g sont équipées de processeurs Graviton2 basés sur ARM64 AWS .

Les instances DLAMI fournissent des outils pour surveiller et optimiser vos processus GPU. Pour plus d'informations sur la surveillance de vos processus GPU, consultezOptimisation et surveillance des GPU.

Pour des didacticiels spécifiques sur l'utilisation des instances G5g, consultezLe ARM64 DLAMI.

Suivant

Instances d'UC recommandées

Instances d'UC recommandées

Que vous ayez un budget restreint, que vous débutiez en apprentissage profond ou que vous souhaitiez seulement exécuter un service de prévision, vous avez plusieurs options abordables dans la catégorie UC. Certains frameworks tirent parti du MKL DNN d'Intel, qui accélère l'entraînement et l'inférence sur les types d'instances de processeur C5 (non disponible dans toutes les régions). Pour plus d'informations sur les types d'instances de processeur, voir d'instances et sélectionnez Optimisé pour le calcul.

> **Note**
>
> La taille de votre modèle doit être prise en compte dans le choix d'une instance. Si votre modèle dépasse la RAM disponible d'une instance, choisissez un autre type d'instance avec suffisamment de mémoire pour votre application.

- Les instances Amazon EC2 C5 possèdent jusqu'à 72 processeurs Intel v. CPUs Les instances C5 excellent dans les domaines de la modélisation scientifique, du traitement par lots, de l'analyse distribuée, du calcul haute performance (HPC) et de l'inférence par machine et par apprentissage profond.

Suivant

Instances d'inférence recommandées

Instances d'inférence recommandées

AWS Les instances Inferentia sont conçues pour fournir des performances élevées et une rentabilité élevées pour les charges de travail d'inférence de modèles d'apprentissage profond. Plus précisément, les types d'instances Inf2 utilisent les puces AWS Inferentia et le SDK AWS Neuron, qui est intégré aux frameworks d'apprentissage automatique populaires tels que et. TensorFlow PyTorch

Les clients peuvent utiliser les instances Inf2 pour exécuter des applications d'inférence d'apprentissage automatique à grande échelle, telles que la recherche, les moteurs de recommandation, la vision par ordinateur, la reconnaissance vocale, le traitement du langage naturel, la personnalisation et la détection des fraudes, au moindre coût dans le cloud.

> **Note**
>
> La taille de votre modèle doit être prise en compte dans le choix d'une instance. Si votre modèle dépasse la RAM disponible d'une instance, choisissez un autre type d'instance avec suffisamment de mémoire pour votre application.

- Les instances Amazon EC2 Inf2 possèdent jusqu'à 16 puces AWS Inferentia et un débit réseau de 100 Gbit/s.

Pour plus d'informations sur la prise en main d' AWS Inferentia DLAMIs, consultezLa puce AWS Inferentia avec DLAMI.

Suivant

Instances Trainium recommandées

Instances Trainium recommandées

AWS Les instances Trainium sont conçues pour fournir des performances élevées et une rentabilité élevées pour les charges de travail d'inférence de modèles de deep learning. Plus précisément, les types d'instances Trn1 utilisent AWS des puces Trainium et le SDK AWS Neuron, qui est intégré aux frameworks d'apprentissage automatique populaires tels que et. TensorFlow PyTorch

Les clients peuvent utiliser les instances Trn1 pour exécuter des applications d'inférence d'apprentissage automatique à grande échelle, telles que la recherche, les moteurs de recommandation, la vision par ordinateur, la reconnaissance vocale, le traitement du langage naturel, la personnalisation et la détection des fraudes, au moindre coût dans le cloud.

> **Note**
>
> La taille de votre modèle doit être prise en compte dans le choix d'une instance. Si votre modèle dépasse la RAM disponible d'une instance, choisissez un autre type d'instance avec suffisamment de mémoire pour votre application.

- Les instances Amazon EC2 Trn1 possèdent jusqu'à 16 puces AWS Trainium et un débit réseau de 100 Gbit/s.

Configuration d'une instance DLAMI

Après avoir choisi un DLAMI et le type d'instance Amazon Elastic Compute Cloud (EC2Amazon) que vous souhaitez utiliser, vous êtes prêt à configurer votre nouvelle instance DLAMI.

Si vous n'avez pas encore choisi de DLAMI EC2 et de type d'instance, consultez. Commencer à utiliser le DLAMI

Rubriques

Trouver l'identifiant d'un DLAMI

Chaque DLAMI possède un identifiant (ID) unique. Lorsque vous lancez une instance DLAMI à l'aide de la console EC2 Amazon, vous pouvez éventuellement utiliser l'ID DLAMI pour rechercher le DLAMI que vous souhaitez utiliser. Lorsque vous lancez une instance DLAMI à l'aide AWS Command Line Interface du AWS CLI(), cet ID est requis.

Vous pouvez trouver l'ID du DLAMI de votre choix en utilisant AWS CLI une commande pour EC2 Amazon ou Parameter Store, une fonctionnalité de. AWS Systems Manager Pour obtenir des instructions sur l'installation et la configuration du AWS CLI, voir Commencer avec le AWS CLI dans le guide de AWS Command Line Interface l'utilisateur.

Using Parameter Store

Pour trouver un identifiant DLAMI à l'aide de ssm get-parameter

Dans la ssm get-parametercommande suivante, pour l'`--name`option, le format du nom du paramètre est*`/aws/service/deeplearning/ami/$architecture/$ami_type/latest/ami-id`*. Dans ce format de nom, il *`architecture`* peut s'agir de l'un **`x86_64`** ou de l'autre**`arm64`**. *`ami_type`*Spécifiez-le en prenant le nom du DLAMI et en supprimant les mots clés « deep », « learning » et « ami ». Le nom de l'AMI se trouve dansNotes de AMIs mise à jour du Deep Learning.

> **Important**
>
> Pour utiliser cette commande, le principal AWS Identity and Access Management (IAM) que vous utilisez doit disposer de l'`ssm:GetParameter`autorisation. Pour plus d'informations sur les principaux IAM, consultez la section Ressources supplémentaires sur les rôles IAM dans le Guide de l'utilisateur 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
  ```

  La sortie doit ressembler à ce qui suit :

  ```
  ami-09ee1a996ac214ce7
  ```

  > **Tip**
  >
  > Pour certains frameworks DLAMI actuellement pris en charge, vous trouverez des exemples de commandes plus spécifiques ssm get-parameter dans. Notes de AMIs mise à jour du Deep Learning Cliquez sur le lien vers les notes de version du DLAMI de votre choix, puis recherchez sa requête d'identification dans les notes de version.

Using Amazon EC2 CLI

Pour trouver un identifiant DLAMI à l'aide de ec2 describe-images

Dans la ec2 describe-imagescommande suivante, pour la valeur du filtre`Name=name`, entrez le nom du DLAMI. Vous pouvez spécifier une version de version pour un framework donné, ou vous pouvez obtenir la dernière version en remplaçant le numéro de version par un point d'interrogation (?).

- ```
  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, &amp;CreationDate))[:1].ImageId' --output text
  ```

La sortie doit ressembler à ce qui suit :

```
ami-09ee1a996ac214ce7
```

> **Tip**
>
> Pour un exemple de ec2 describe-images commande spécifique au DLAMI de votre choix, consultez. Notes de AMIs mise à jour du Deep Learning Cliquez sur le lien vers les notes de version du DLAMI de votre choix, puis recherchez sa requête d'identification dans les notes de version.

Étape suivante

Lancement d'une instance DLAMI

Lancement d'une instance DLAMI

Une fois que vous avez trouvé l'ID du DLAMI que vous souhaitez utiliser pour lancer une instance DLAMI, vous êtes prêt à lancer l'instance. Pour le lancer, vous pouvez utiliser la EC2 console Amazon ou le AWS Command Line Interface (AWS CLI).

> **Note**
>
> Pour cette présentation, nous pouvons faire des références spécifiques à l'AMI GPU du pilote Nvidia Deep Learning Base OSS (Ubuntu 22.04). Même si vous sélectionnez un autre DLAMI, vous devriez pouvoir suivre ce guide.

EC2 console

> **Note**
>
> Pour accélérer les applications de calcul haute performance (HPC) et d'apprentissage automatique, vous pouvez lancer votre instance DLAMI avec un Elastic Fabric Adapter (EFA). Pour des instructions spécifiques, voirLancement d'une AWS Apprentissage profond (deep learning) AMIs instance avec EFA.

1. Ouvrez la EC2 console.
2. Notez votre actuel Région AWS dans la barre de navigation supérieure. Si ce n'est pas la région que vous souhaitez, modifiez cette option avant de continuer. Pour plus d'informations, consultez Amazon EC2 Service endpoints dans le Référence générale d'Amazon Web Services.
3. Choisissez Launch Instances (Lancer les instances).
4. Entrez un nom pour votre instance et sélectionnez le DLAMI qui vous convient.

   a. Recherchez un DLAMI existant dans AMIs Mon ou choisissez Quick Start.

   b. Recherche par ID DLAMI. Parcourez les options, puis sélectionnez votre choix.
5. Choisissez un type d'instance. Vous trouverez les familles d'instances recommandées pour votre DLAMI dans. Notes de AMIs mise à jour du Deep Learning Pour des recommandations générales sur les types d'instances DLAMI, consultez. Choix d'un type d'instance DLAMI
6. Choisissez Launch Instances (Lancer les instances).

AWS CLI

- Pour utiliser le AWS CLI, vous devez disposer de l'ID du DLAMI que vous souhaitez utiliser, du type d'instance EC2 et Région AWS des informations relatives à votre jeton de sécurité. Vous pouvez ensuite lancer l'instance à l'aide de la ec2 run-instances AWS CLI commande.

  Pour obtenir des instructions sur l'installation et la configuration du AWS CLI, voir Commencer avec le AWS CLI dans le guide de AWS Command Line Interface l'utilisateur. Pour plus d'informations, y compris des exemples de commandes, consultez Lancer, lister et fermer des EC2 instances Amazon pour le AWS CLI.

Après avoir lancé votre instance à l'aide de la EC2 console Amazon ou AWS CLI attendez qu'elle soit prête. Cela ne prend généralement que quelques minutes. Vous pouvez vérifier le statut de l'instance dans la EC2 console Amazon. Pour plus d'informations, consultez la section Contrôles de statut pour les EC2 instances Amazon dans le guide de EC2 l'utilisateur Amazon.

Étape suivante

Connexion à une instance DLAMI

Connexion à une instance DLAMI

Une fois que vous avez lancé une instance DLAMI et que celle-ci est en cours d'exécution, vous pouvez vous y connecter à partir d'un client (Windows, macOS ou Linux) via SSH. Pour obtenir des instructions, consultez Connect to your Linux instance using SSH dans le Amazon EC2 User Guide.

Conservez une copie de la commande de connexion SSH à portée de main au cas où vous souhaiteriez configurer un serveur Jupyter Notebook après vous être connecté. Pour vous connecter à la page Web de Jupyter, vous devez utiliser une variante de cette commande.

Étape suivante

Configuration d'un serveur Jupyter Notebook sur une instance DLAMI

Configuration d'un serveur Jupyter Notebook sur une instance DLAMI

Avec un serveur Jupyter Notebook, vous pouvez créer et exécuter des blocs-notes Jupyter à partir de votre instance DLAMI. Avec les blocs-notes Jupyter, vous pouvez réaliser des expériences d'apprentissage automatique (ML) à des fins d'entraînement et d'inférence tout en utilisant l' AWS infrastructure et en accédant aux packages intégrés au DLAMI. Pour plus d'informations sur les blocs-notes Jupyter, consultez The Jupyter Notebook sur le site Web de documentation utilisateur de Jupyter.

Pour configurer un serveur Jupyter Notebook, vous devez :

- Configurez le serveur Jupyter Notebook sur votre instance DLAMI.
- Configurez votre client pour qu'il se connecte au serveur Jupyter Notebook. Nous fournissons des instructions de configuration pour les clients Windows, macOS et Linux.
- Testez la configuration en vous connectant au serveur Jupyter Notebook.

Pour effectuer ces étapes, suivez les instructions des rubriques suivantes. Après avoir configuré un serveur Jupyter Notebook, vous pouvez exécuter les exemples de didacticiels pour bloc-notes fournis dans le. DLAMIs Pour de plus amples informations, veuillez consulter Exécution des didacticiels blocs-notes Jupyter.

Rubriques

- Sécurisation du serveur Jupyter Notebook sur une instance DLAMI

- Démarrage du serveur Jupyter Notebook sur une instance DLAMI
- Connexion d'un client au serveur Jupyter Notebook sur une instance DLAMI
- Connexion au serveur Jupyter Notebook sur une instance DLAMI

Sécurisation du serveur Jupyter Notebook sur une instance DLAMI

Pour garantir la sécurité de votre serveur Jupyter Notebook, nous vous recommandons de définir un mot de passe et de créer un certificat SSL pour le serveur. Pour configurer un mot de passe et un protocole SSL, connectez-vous d'abord à votre instance DLAMI, puis suivez ces instructions.

Pour sécuriser le serveur Jupyter Notebook

1. Jupyter fournit un mot de passe utilitaire. Exécutez la commande suivante et entrez le mot de passe de votre choix à l'invite.

   ```
   $ jupyter notebook password
   ```

   Le résultat doit se présenter comme suit :

   ```
   Enter password:
        Verify password:
        [NotebookPasswordApp] Wrote hashed password to /home/ubuntu/.jupyter/
   jupyter_notebook_config.json
   ```

2. Pour créer un certificat auto-signé Suivez les instructions pour remplir votre localité selon vos besoins. Vous devez entrer `.` si vous souhaitez qu'une invite reste vide. Vos réponses n'a pas d'impact sur les fonctionnalités du certificat.

   ```
   $ cd ~
        $ mkdir ssl
        $ cd ssl
        $ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout mykey.key -out
    mycert.pem
   ```

> **Note**
>
> Vous pourriez être intéressé par la création d'un certificat SSL standard signé par un tiers et qui n'entraîne pas d'avertissement de sécurité par le navigateur. Ce processus est nettement

plus complexe. Pour plus d'informations, consultez la section Sécurisation d'un serveur de bloc-notes dans la documentation utilisateur de Jupyter Notebook.

Étape suivante

Démarrage du serveur Jupyter Notebook sur une instance DLAMI

Démarrage du serveur Jupyter Notebook sur une instance DLAMI

Après avoir sécurisé votre serveur Jupyter Notebook avec un mot de passe et un protocole SSL, vous pouvez démarrer le serveur. Connectez-vous à votre instance DLAMI et exécutez la commande suivante qui utilise le certificat SSL que vous avez créé précédemment.

```
$ jupyter notebook --certfile=~/ssl/mycert.pem --keyfile ~/ssl/mykey.key
```

Une fois le serveur démarré, vous pouvez vous y connecter via un tunnel SSH à partir de votre ordinateur client. Lorsque le serveur fonctionne, un message de Jupyter vous le confirme. À ce stade, ignorez l'avertissement indiquant que vous pouvez accéder au serveur via une URL d'hôte local, car cela ne fonctionnera pas tant que vous n'aurez pas créé le tunnel.

> **Note**
>
> Jupyter s'en occupera lorsque vous changerez d'infrastructure à l'aide de l'interface web Jupyter. Pour de plus amples informations, veuillez consulter Changer d'environnement avec Jupyter.

Étape suivante

Connexion d'un client au serveur Jupyter Notebook sur une instance DLAMI

Connexion d'un client au serveur Jupyter Notebook sur une instance DLAMI

Après avoir démarré le serveur Jupyter Notebook sur votre instance DLAMI, configurez votre client Windows, macOS ou Linux pour qu'il se connecte au serveur. Lorsque vous vous connectez, vous pouvez créer et accéder à des blocs-notes Jupyter sur le serveur de votre espace de travail et exécuter votre code de deep learning sur le serveur.

Prérequis

Assurez-vous de disposer des éléments suivants, dont vous avez besoin pour configurer un tunnel SSH :

- Le nom DNS public de votre EC2 instance Amazon. Pour plus d'informations, consultez les types de noms d'hôte des EC2 instances Amazon dans le guide de EC2 l'utilisateur Amazon.
- La paire de clés pour le fichier de clé privée. Pour plus d'informations sur l'accès à votre paire de clés, consultez la section relative aux paires de EC2 clés Amazon et aux EC2 instances Amazon dans le guide de EC2 l'utilisateur Amazon.

Connect depuis un client Windows, macOS ou Linux

Pour vous connecter à votre instance DLAMI depuis un client Windows, macOS ou Linux, suivez les instructions du système d'exploitation de votre client.

Windows

Pour vous connecter à votre instance DLAMI à partir d'un client Windows via SSH

1. Utilisez un client SSH pour Windows, tel que PuTTY. Pour obtenir des instructions, consultez Connect to your Linux instance using PuTTY dans le guide de EC2 l'utilisateur Amazon. Pour les autres options de connexion SSH, consultez Se connecter à votre instance Linux à l'aide de SSH.
2. (Facultatif) Créez un tunnel SSH vers un serveur Jupyter en cours d'exécution. Installez Git Bash sur votre client Windows, puis suivez les instructions de connexion pour les clients macOS et Linux.

macOS or Linux

Pour vous connecter à votre instance DLAMI à partir d'un client macOS ou Linux à l'aide de SSH

1. Ouvrez un terminal .
2. Exécutez la commande suivante pour transférer toutes les demandes sur le port local 8888 vers le port 8888 de votre instance Amazon EC2 distante. Mettez à jour la commande en remplaçant l'emplacement de votre clé pour accéder à l' EC2 instance Amazon et le nom DNS public de votre EC2 instance Amazon. Notez que pour une AMI Amazon Linux, le nom d'utilisateur est `ec2-user`au lieu de `ubuntu`.

```
$ ssh -i ~/mykeypair.pem -N -f -L 8888:localhost:8888 ubuntu@ec2-###-##-##-
###.compute-1.amazonaws.com
```

Cette commande ouvre un tunnel entre votre client et l' EC2 instance Amazon distante qui exécute le serveur Jupyter Notebook.

Étape suivante

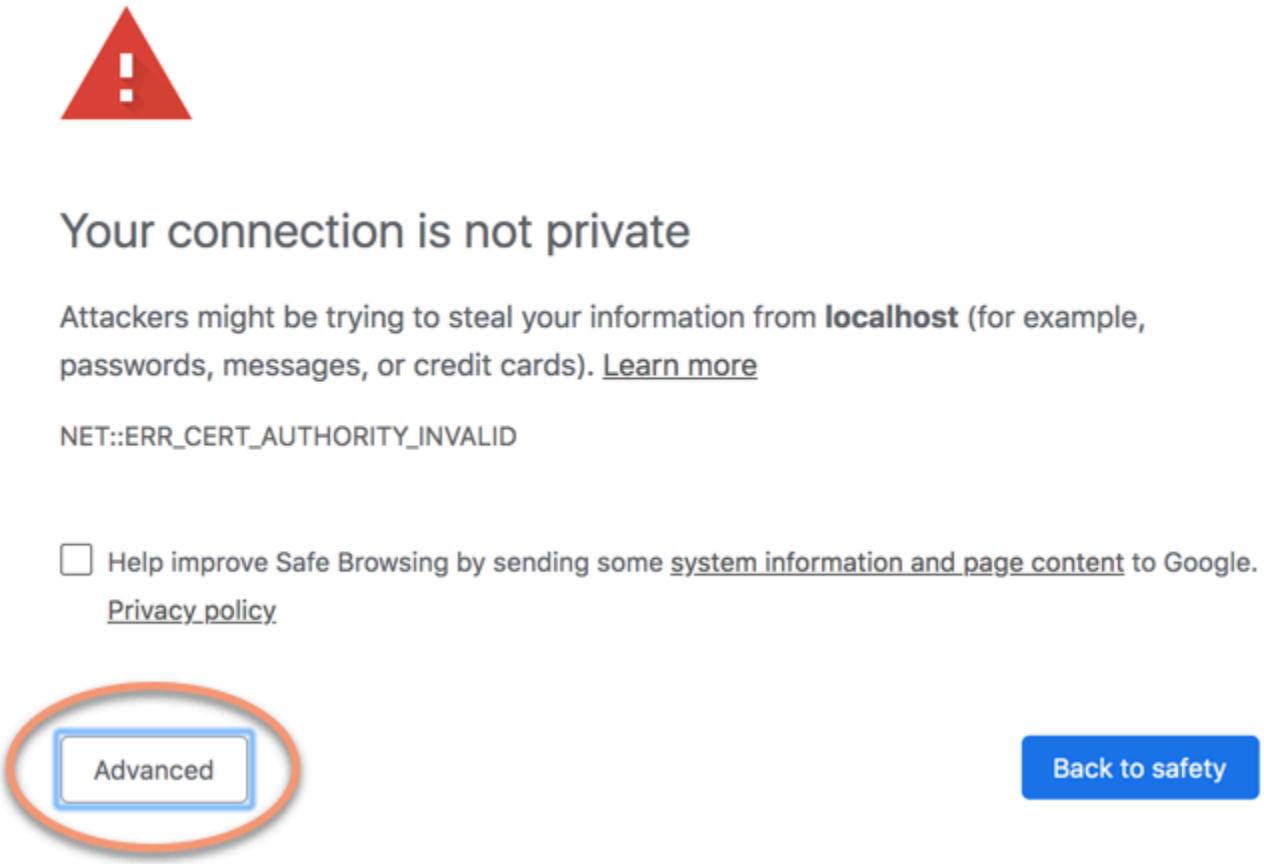
Connexion au serveur Jupyter Notebook sur une instance DLAMI

Connexion au serveur Jupyter Notebook sur une instance DLAMI

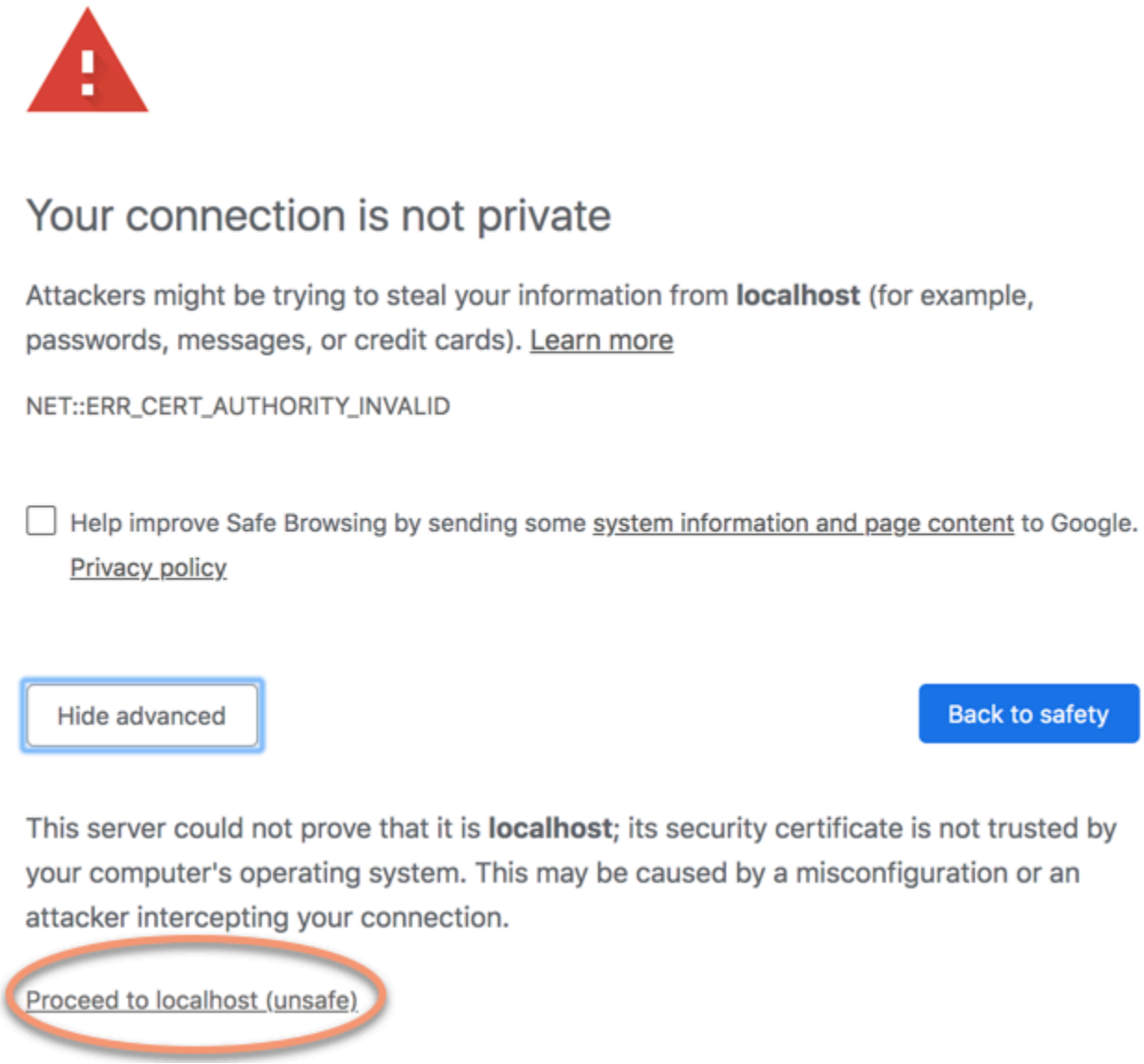
Après avoir connecté votre client au serveur Jupyter Notebook sur votre instance DLAMI, vous pouvez vous connecter au serveur.

Pour vous connecter au serveur dans votre navigateur

1. Dans la barre d'adresse de votre navigateur, saisissez l'URL suivante ou cliquez sur ce lien : https://localhost:8888
2. Avec un certificat SSL autosigné, votre navigateur vous avertira et vous demandera d'éviter de continuer à visiter le site Web.



Dans la mesure où vous avez configuré cela vous-même, la procédure peut se poursuivre. En fonction de votre navigateur, vous obtenez un « advanced », « afficher les détails », ou similaire.



Cliquez sur, puis cliquez sur le lien « passez à localhost ». Si la connexion est établie, la page Web du serveur Jupyter Notebook s'affiche. À ce stade, le mot de passe que vous avez configuré précédemment vous sera demandé.

Vous avez maintenant accès au serveur Jupyter Notebook qui s'exécute sur l'instance DLAMI. Vous pouvez créer de nouveaux blocs-notes ou exécuter les Didacticiels.

Nettoyage d'une instance DLAMI

Lorsque vous n'avez plus besoin de votre instance DLAMI, vous pouvez l'arrêter ou la résilier sur EC2 Amazon pour éviter des frais imprévus.

Si vous arrêtez une instance, vous pouvez la conserver et la redémarrer ultérieurement lorsque vous souhaiterez l'utiliser à nouveau. Vos configurations, fichiers et autres informations non volatiles sont stockés dans un volume sur Amazon Simple Storage Service (Amazon S3). Lorsque votre instance est arrêtée, vous devez payer des frais S3 pour conserver le volume, mais pas pour les ressources de calcul. Lorsque vous redémarrez l'instance, elle monte ce volume de stockage avec vos données.

Si vous mettez fin à une instance, elle disparaît et vous ne pouvez pas la redémarrer. Bien entendu, vous n'aurez plus à payer de frais pour les ressources de calcul en cas de résiliation d'une instance. Cependant, vos données se trouvent toujours sur Amazon S3 et vous pouvez continuer à payer des frais S3. Pour éviter tous frais supplémentaires liés à la résiliation de votre instance, vous devez également supprimer le volume de stockage sur Amazon S3. Pour obtenir des instructions, consultez la section Résiliation d' EC2 instances Amazon dans le guide de EC2 l'utilisateur Amazon.

Pour plus d'informations sur les états des EC2 instances Amazon, tels que `stopped` et`terminated`, consultez les modifications de l'état des EC2 instances Amazon dans le guide de EC2 l'utilisateur Amazon.

Utilisation d'un DLAMI

Rubriques

Les sections suivantes décrivent comment l'AMI Deep Learning avec Conda peut être utilisée pour changer d'environnement, exécuter des exemples de code à partir de chacun des frameworks et exécuter Jupyter afin que vous puissiez essayer différents didacticiels sur les blocs-notes.

Utilisation de l'AMI Deep Learning avec Conda

Rubriques

Présentation de l'AMI d'apprentissage profond avec Conda

Conda est un système de gestion de packages Open Source et système de gestion d'environnement qui s'exécute sur Windows, macOS et Linux. Conda s'installe, s'exécute et met à jour rapidement des packages et leurs dépendances. Conda crée, enregistre, charge et bascule facilement entre les différents environnements sur votre ordinateur local.

L'AMI Deep Learning avec Conda a été configurée pour que vous puissiez facilement passer d'un environnement d'apprentissage profond à un autre. Les instructions suivantes vous guident pour certaines commandes élémentaires de `conda`. Elles vous permettent également de vérifier que l'importation de base de l'infrastructure fonctionne et que vous pouvez lancer quelques opérations simples avec l'infrastructure. Vous pouvez ensuite passer à des didacticiels plus approfondis fournis avec le DLAMI ou aux exemples de frameworks trouvés sur le site de projet de chaque framework.

Connectez-vous à votre DLAMI

Une fois que vous êtes connecté à votre serveur, vous verrez un « message du jour » (MOTD) du serveur décrivant diverses commandes Conda que vous pouvez utiliser pour basculer entre les différentes infrastructures d'apprentissage profond (deep learning). Vous trouverez ci-dessous un exemple de MOTD. Votre MOTD spécifique peut varier à mesure que de nouvelles versions du DLAMI sont publiées.

```
=============================================================================
        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
    =============================================================================
```

Démarrez l' TensorFlow environnement

> **Note**
>
> Lorsque vous lancez votre premier environnement Conda, soyez patient pendant qu'il se charge. L'AMI Deep Learning avec Conda installe automatiquement la version la plus

optimisée du framework pour votre EC2 instance lors de la première activation du framework. Des retards ultérieurs sont peu probables.

1. Activez l'environnement TensorFlow virtuel pour Python 3.

```
$ source activate tensorflow2_p310
```

2. Démarrez le terminal iPython.

```
(tensorflow2_p310)$ ipython
```

3. Lancez un TensorFlow programme rapide.

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Vous devez voir « Hello, Tensorflow ! »

Suivant

Exécution des didacticiels blocs-notes Jupyter

Passez à l'environnement PyTorch Python 3

Si vous êtes toujours dans la console IPython, `quit()` utilisez-la, puis préparez-vous à changer d'environnement.

- Activez l'environnement PyTorch virtuel pour Python 3.

```
$ source activate pytorch_p310
```

Testez PyTorch du code

Pour tester votre installation, utilisez Python pour écrire PyTorch du code qui crée et imprime un tableau.

1. Démarrez le terminal iPython.

```
(pytorch_p310)$ ipython
```

2. Importer PyTorch.

```
import torch
```

Vous pouvez voir un message d'avertissement concernant un package tiers. Vous pouvez l'ignorer.

3. Créez une matrice 5x3 avec les éléments initialisés de manière aléatoire. Imprimez le tableau.

```
x = torch.rand(5, 3)
print(x)
```

Vérifiez le résultat.

```
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]])
```

Suppression d’environnements

Si vous manquez d'espace sur le DLAMI, vous pouvez choisir de désinstaller les packages Conda que vous n'utilisez pas :

```
conda env list
conda env remove --name <env_name>
```

Utilisation de l'AMI Deep Learning Base

Utilisation de l'AMI Deep Learning Base

L'AMI de base est fournie avec une plate-forme de base de pilotes et de bibliothèques d'accélération de GPU pour déployer votre propre environnement d'apprentissage profond personnalisé. Par défaut,

l'AMI est configurée avec n'importe quel environnement de version NVIDIA CUDA. Vous pouvez également passer d'une version de CUDA à une autre. Consultez les instructions suivantes pour savoir comment procéder.

Configuration des versions CUDA

Vous pouvez vérifier la version CUDA en exécutant le `nvcc` programme NVIDIA.

```
nvcc --version
```

Vous pouvez sélectionner et vérifier une version particulière de CUDA à l'aide de la commande bash suivante :

```
sudo rm /usr/local/cuda
sudo ln -s /usr/local/cuda-12.0 /usr/local/cuda
```

Pour plus d'informations, consultez les notes de mise à jour du DLAMI de base.

Exécution des didacticiels blocs-notes Jupyter

Les didacticiels et les exemples sont fournis avec la source de chacun des projets de deep learning et, dans la plupart des cas, ils s'exécuteront sur n'importe quel DLAMI. Si vous avez choisi l'AMI d'apprentissage profond avec Conda, vous bénéficiez de quelques didacticiels déjà configurés et prêts à tester.

> **Important**
>
> Pour exécuter les didacticiels du bloc-notes Jupyter installés sur le DLAMI, vous devez. Configuration d'un serveur Jupyter Notebook sur une instance DLAMI

Une fois le serveur Jupyter en cours d'exécution, vous pouvez exécuter les didacticiels via votre navigateur Web. Si vous exécutez l'AMI Deep Learning avec Conda ou si vous avez configuré des environnements Python, vous pouvez changer de noyau Python depuis l'interface du bloc-notes Jupyter. Sélectionnez le noyau approprié avant de tenter d'exécuter un didacticiel spécifique d'une infrastructure. D'autres exemples sont fournis aux utilisateurs de l'AMI Deep Learning avec Conda.

> **Note**
>
> De nombreux didacticiels nécessitent des modules Python supplémentaires qui ne sont peut-être pas configurés sur votre DLAMI. Si un message d'erreur s'affiche`"xyz module not found"`, par exemple, connectez-vous au DLAMI, activez l'environnement comme décrit ci-dessus, puis installez les modules nécessaires.

> **Tip**
>
> Les didacticiels et les exemples de deep learning s'appuient souvent sur un ou plusieurs d'entre eux GPUs. Si votre type d'instance n'a pas de GPU, il est possible que vous deviez modifier une partie du code des exemples pour qu'ils fonctionnent.

Navigation dans les didacticiels installés

Une fois que vous êtes connecté au serveur Jupyter et que vous pouvez voir le répertoire des tutoriels (sur l'AMI Deep Learning avec Conda uniquement), des dossiers de didacticiels portant le nom de chaque framework vous seront présentés. Si aucun framework n'est répertorié, cela signifie qu'aucun didacticiel n'est disponible pour ce framework sur votre DLAMI actuel. Cliquez sur le nom de l'infrastructure pour voir les didacticiels répertoriés, puis cliquez sur un didacticiel pour le lancer.

La première fois que vous exécuterez un bloc-notes sur l'AMI Deep Learning avec Conda, il voudra savoir quel environnement vous souhaitez utiliser. Vous serez invité à le sélectionner dans une liste. Chaque environnement est nommé selon ce modèle :

`Environment (conda_framework_python-version)`

Par exemple, vous pourriez voir`Environment (conda_mxnet_p36)`, ce qui signifie que l'environnement possède MXNet Python 3. L'autre variante serait`Environment (conda_mxnet_p27)`, ce qui signifie que l'environnement possède MXNet Python 2.

> **Tip**
>
> Si vous vous demandez quelle version de CUDA est active, vous pouvez la voir dans le MOTD lorsque vous vous connectez pour la première fois au DLAMI.

Changer d'environnement avec Jupyter

Si vous décidez de tester un didacticiel pour une infrastructure différente, assurez-vous de vérifier le noyau en cours d'exécution. Cette information est visible dans le coin supérieur droit de l'interface Jupyter, juste sous le bouton de déconnexion. Vous pouvez modifier le noyau sur n'importe quel bloc-note ouvert, en cliquant sur l'option de menu Jupyter Kernel, puis sur Change Kernel, puis en cliquant sur l'environnement qui correspond au bloc-notes que vous exécutez.

À ce stade, vous devrez réexécuter toutes les cellules, parce qu'une modification du noyau effacera l'état de tout ce que vous avez exécuté précédemment.

> **Tip**
>
> Il peut être amusant et instructif de passer d'une infrastructure à une autre, mais vous pouvez manquer de mémoire. Si vous commencez à voir des erreurs, consultez la fenêtre du terminal sur lequel le serveur Jupyter est en cours d'exécution. Vous y trouverez des messages utiles et un journal d'erreurs, et il se peut qu'une out-of-memory erreur s'affiche. Pour corriger ce problème, vous pouvez accéder à la page d'accueil de votre serveur Jupyter, cliquez sur l'onglet En cours d'exécution, puis sur Fermeture pour chacun des didacticiels qui sont probablement toujours en cours d'exécution en arrière-plan et qui consomment toute votre mémoire.

Didacticiels

Vous trouverez ci-dessous des didacticiels sur l'utilisation de l'AMI Deep Learning avec le logiciel Conda.

Rubriques

Activation des infrastructures

Les frameworks d'apprentissage profond installés sur l'AMI Deep Learning avec Conda sont les suivants. Cliquez sur une infrastructure pour découvrir comment l'activer.

Rubriques

PyTorch

Activation PyTorch

Lorsqu'un package Conda stable d'un framework est publié, il est testé et préinstallé sur le DLAMI. Si vous voulez exécuter la dernière génération nocturne non testée, vous pouvez PyTorchInstall's Nightly Build (expérimental) manuellement.

Pour activer le framework actuellement installé, suivez ces instructions sur votre AMI Deep Learning avec Conda.

Pour PyTorch Python 3 avec CUDA et MKL-DNN, exécutez cette commande :

```
$ source activate pytorch_p310
```

Démarrez le terminal iPython.

```
(pytorch_p310)$ ipython
```

Lancez un PyTorch programme rapide.

```
import torch
x = torch.rand(5, 3)
print(x)
print(x.size())
y = torch.rand(5, 3)
print(torch.add(x, y))
```

Vous devriez voir le tableau aléatoire initial imprimé, puis sa taille, puis l'ajout d'un autre tableau aléatoire.

PyTorchInstall's Nightly Build (expérimental)

Comment procéder à l'installation PyTorch à partir d'une version nocturne

Vous pouvez installer la dernière PyTorch version dans l'un des environnements PyTorch Conda ou dans les deux sur votre AMI Deep Learning avec Conda.

1. • (Option pour Python 3) - Activez l' PyTorch environnement Python 3 :

```
$ source activate pytorch_p310
```

2. Les étapes restantes partent du principe que vous utilisez l'environnement `pytorch_p310`. Supprimez le fichier actuellement installé PyTorch :

```
(pytorch_p310)$ pip uninstall torch
```

3. • (Option pour les instances GPU) - Installez la dernière version nocturne de PyTorch CUDA.0 :

```
(pytorch_p310)$ pip install torch_nightly -f https://download.pytorch.org/whl/
nightly/cu100/torch_nightly.html
```

   • (Option pour les instances de processeur) - Installez la dernière version nocturne de PyTorch pour les instances sans GPUs :

```
(pytorch_p310)$ pip install torch_nightly -f https://download.pytorch.org/whl/
nightly/cpu/torch_nightly.html
```

4. Pour vérifier que vous avez correctement installé la dernière version nocturne, démarrez le IPython terminal et vérifiez la version de PyTorch.

```
(pytorch_p310)$ ipython
```

```
import torch
print (torch.__version__)
```

   Vous devez obtenir une sortie imprimée similaire à `1.0.0.dev20180922`

5. Pour vérifier que la version PyTorch nocturne fonctionne correctement avec l'exemple MNIST, vous pouvez exécuter un script de test à partir PyTorch du référentiel d'exemples :

```
(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
```

Autres didacticiels

Pour d'autres didacticiels et exemples, reportez-vous à la documentation officielle du framework, à la PyTorch documentation et au PyTorchsite Web.

TensorFlow 2

Ce didacticiel explique comment activer TensorFlow 2 sur une instance exécutant l'AMI Deep Learning avec Conda (DLAMI sur Conda) et exécuter un programme 2. TensorFlow

Lorsqu'un package Conda stable d'un framework est publié, il est testé et préinstallé sur le DLAMI.

Activation de TensorFlow 2

Pour utiliser TensorFlow le DLAMI avec Conda

1. Pour activer TensorFlow 2, ouvrez une instance Amazon Elastic Compute Cloud (Amazon EC2) du DLAMI avec Conda.
2. Pour TensorFlow 2 et Keras 2 sur Python 3 avec CUDA 10.1 et MKL-DNN, exécutez cette commande :

   ```
   $ source activate tensorflow2_p310
   ```
3. Démarrez le terminal iPython:

   ```
   (tensorflow2_p310)$ ipython
   ```
4. Exécutez un programme TensorFlow 2 pour vérifier qu'il fonctionne correctement :

   ```
   import tensorflow as tf
   hello = tf.constant('Hello, TensorFlow!')
   tf.print(hello)
   ```

`Hello, TensorFlow!` doit apparaître sur votre écran.

Autres didacticiels

Pour plus de didacticiels et d'exemples, consultez la TensorFlow documentation de l'API TensorFlow Python ou consultez le TensorFlowsite Web.

Entraînement distribué à l'aide d'Elastic Fabric Adapter

Un adaptateur Elastic Fabric (EFA) est un périphérique réseau que vous pouvez connecter à votre instance DLAMI pour accélérer les applications de calcul haute performance (HPC). EFA vous permet d'atteindre les performances applicatives d'un cluster HPC sur site, grâce à l'évolutivité, à la flexibilité et à l'élasticité offertes par le cloud. AWS

Les rubriques suivantes expliquent comment commencer à utiliser l'EFA avec le DLAMI.

> **Note**
>
> Choisissez votre DLAMI dans cette liste de DLAMI pour GPU de base

Rubriques

- Lancement d'une AWS Apprentissage profond (deep learning) AMIs instance avec EFA
- Utilisation de l'EFA sur le DLAMI

Lancement d'une AWS Apprentissage profond (deep learning) AMIs instance avec EFA

Le dernier DLAMI de base est prêt à être utilisé avec EFA et est fourni avec les pilotes requis, les modules de noyau, libfabric, openmpi et le plugin NCCL OFI pour les instances GPU.

Vous trouverez les versions CUDA prises en charge d'un DLAMI de base dans les notes de publication.

Remarque :

- Lorsque vous exécutez une application NCCL sous `mpirun` EFA, vous devez spécifier le chemin complet vers l'installation prise en charge par EFA comme suit :

```
/opt/amazon/openmpi/bin/mpirun <command>
```

- Pour permettre à votre application d'utiliser EFA, ajoutez `FI_PROVIDER="efa"` à la commande `mpirun` comme indiqué dans Utilisation de l'EFA sur le DLAMI.

Rubriques

- Préparer un groupe de sécurité compatible EFA
- Lancer votre instance
- Vérifiez la pièce jointe EFA

Préparer un groupe de sécurité compatible EFA

L'EFA nécessite un groupe de sécurité qui autorise tout le trafic entrant et sortant à destination et en provenance du groupe de sécurité lui-même. Pour plus d'informations, consultez la documentation EFA.

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le panneau de navigation, choisissez Groupes de sécurité, puis Créer un groupe de sécurité.
3. Dans la fenêtre Créer un groupe de sécurité, procédez comme suit :
   - Pour Nom du groupe de sécurité, saisissez un nom descriptif pour le groupe de sécurité, tel que `EFA-enabled security group`.
   - (Facultatif) Pour Description, saisissez une brève description du groupe de sécurité.
   - Pour VPC, sélectionnez le VPC dans lequel vous prévoyez de lancer vos instances activées pour EFA.
   - Choisissez Créer.
4. Sélectionnez le groupe de sécurité que vous avez créé et dans l'onglet Description, copiez l'ID du groupe.
5. Dans les onglets Entrant et Sortant, procédez comme suit :
   - Choisissez Modifier.
   - Pour Type, sélectionnez Tout le trafic.
   - Pour Source, choisissez Personnalisé.

- Collez l'ID de groupe de sécurité que vous avez copié dans le champ.
- Choisissez Enregistrer.

6. Activez le trafic entrant en vous référant à Autorisation du trafic entrant pour vos instances Linux. Si vous ignorez cette étape, vous ne pourrez pas communiquer avec votre instance DLAMI.

Lancer votre instance

L'EFA sur le AWS Apprentissage profond (deep learning) AMIs est actuellement pris en charge avec les types d'instances et systèmes d'exploitation suivants :

- 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

La section suivante explique comment lancer une instance DLAMI compatible EFA. Pour plus d'informations sur le lancement d'une instance compatible EFA, voir Lancer des instances compatibles EFA dans un groupe de placement de clusters.

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Choisissez Launch Instances (Lancer les instances).
3. Sur la page Choisir une AMI, sélectionnez une DLAMI prise en charge sur la page des notes de mise à jour de la DLAMI
4. Sur la page Choisir un type d'instance, sélectionnez l'un des types d'instance pris en charge suivants, puis choisissez Suivant : Configurer les détails de l'instance. Consultez ce lien pour obtenir la liste des instances prises en charge : Commencez avec EFA et MPI
5. Sur la page Configurer les détails de l’instance, procédez de la façon suivante :
   - Pour Nombre d’instances, entrez le nombre d’instances activées pour EFA que vous voulez lancer.
   - Pour Réseau et Sous-réseau, sélectionnez le VPC et le sous-réseau dans lesquels lancer les instances.
   - [Facultatif] Pour Groupe de placement, sélectionnez Ajouter une instance au groupe de placement. Pour de meilleures performances, lancez les instances au sein d'un groupe de placement.

- [Facultatif] Pour le nom du groupe de placement, sélectionnez Ajouter à un nouveau groupe de placement, entrez un nom descriptif pour le groupe de placement, puis pour Stratégie du groupe de placement, sélectionnez cluster.
- Assurez-vous d'activer le « Elastic Fabric Adapter » sur cette page. Si cette option est désactivée, remplacez le sous-réseau par un sous-réseau qui prend en charge le type d'instance sélectionné.
- Dans la section Interfaces réseau, pour l'appareil eth0, choisissez Nouvelle interface réseau. Vous pouvez éventuellement spécifier une IPv4 adresse principale et une ou plusieurs IPv4 adresses secondaires. Si vous lancez l'instance dans un sous-réseau auquel est associé un bloc IPv6 CIDR, vous pouvez éventuellement spécifier une IPv6 adresse principale et une ou plusieurs adresses secondaires IPv6 .
- Choisissez Suivant : Ajouter un stockage.

6. Sur la page Ajouter un stockage, spécifiez les volumes à attacher aux instances, outre ceux spécifiés par l'AMI (par exemple, le volume du périphérique racine), puis sélectionner Suivant : Ajouter des balises.
7. Sur la page Ajouter des balises, spécifiez des balises pour l'instance, par exemple un nom évocateur, puis sélectionnez Suivant : Configurer le groupe de sécurité.
8. Sur la page Configurer le groupe de sécurité, pour Attribuer un groupe de sécurité, sélectionnez Sélectionner un groupe de sécurité existant, puis sélectionnez le groupe de sécurité que vous avez créé précédemment.
9. Choisissez Vérifier et lancer.
10. Sur la page Examiner le lancement de l'instance, vérifiez les paramètres, puis choisissez Lancer pour sélectionner une paire de clés et lancer votre instance.

Vérifiez la pièce jointe EFA

À partir de la console

Après avoir lancé l'instance, vérifiez les détails de l'instance dans la AWS console. Pour ce faire, sélectionnez l'instance dans la EC2 console et consultez l'onglet Description dans le volet inférieur de la page. Recherchez le paramètre « Interfaces réseau : eth0 » et cliquez sur eth0 pour ouvrir une fenêtre contextuelle. Assurez-vous que « Elastic Fabric Adapter » est activé.

Si EFA n'est pas activé, vous pouvez résoudre ce problème soit en :

- Mettre fin à l' EC2 instance et en lancer une nouvelle en suivant les mêmes étapes. Assurez-vous que l'EFA est joint.
- Attachez EFA à une instance existante.

1. Dans la EC2 console, accédez à Interfaces réseau.
2. Cliquez sur Create a Network Interface (Créer une interface réseau).
3. Sélectionnez le sous-réseau dans lequel se trouve votre instance.
4. Assurez-vous d'activer « Elastic Fabric Adapter » et de cliquer sur Créer.
5. Retournez à l'onglet EC2 Instances et sélectionnez votre instance.
6. Accédez à Actions : État de l'instance et arrêtez l'instance avant d'attacher EFA.
7. Dans Actions, sélectionnez Mise en réseau : Attacher l'interface réseau.
8. Sélectionnez l'interface que vous venez de créer et cliquez sur Attacher.
9. Redémarrez votre instance.

À partir de l'instance

Le script de test suivant est déjà présent sur le DLAMI. Exécutez-le pour vous assurer que les modules de noyau sont correctement chargés.

```
$ fi_info -p efa
```

Votre sortie doit ressembler à ce qui suit :

```
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
```

Vérifier la configuration du groupe de sécurité

Le script de test suivant est déjà présent sur le DLAMI. Exécutez-le pour vérifier que le groupe de sécurité que vous avez créé est correctement configuré.

```
$ cd /opt/amazon/efa/test/
$ ./efa_test.sh
```

Votre sortie doit ressembler à ce qui suit :

```
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
```

S'il cesse de répondre ou s'il ne se termine pas, assurez-vous que votre groupe de sécurité dispose des inbound/outbound règles correctes.

Utilisation de l'EFA sur le DLAMI

La section suivante décrit comment utiliser EFA pour exécuter des applications à nœuds multiples sur le AWS Apprentissage profond (deep learning) AMIs.

Exécution d'applications multi-nœuds avec EFA

Pour exécuter une application sur un cluster de nœuds, la configuration suivante est requise

Rubriques

Activer SSH sans mot de passe

Sélectionnez un nœud de votre cluster comme nœud principal. Les autres nœuds sont appelés nœuds de membre.

1. Sur le nœud principal, générez la paire de clés RSA.

   ```
   ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
   ```

2. Modifiez les autorisations de la clé privée sur le nœud principal.

   ```
   chmod 600 ~/.ssh/id_rsa
   ```

3. Copiez la clé ~/.ssh/id_rsa.pub publique et ajoutez-la à l'~/.ssh/authorized_keysun des nœuds membres du cluster.
4. Vous devriez maintenant pouvoir vous connecter directement aux nœuds de membre du nœud principal en utilisant l'adresse IP privée.

   ```
   ssh <member private ip>
   ```

5. Désactivez la strictHostKey vérification et activez le transfert d'agent sur le nœud principal en ajoutant ce qui suit au fichier ~/.ssh/config sur le nœud principal :

   ```
   Host *
       ForwardAgent yes
   Host *
       StrictHostKeyChecking no
   ```

6. Sur les instances Amazon Linux 2, exécutez la commande suivante sur le nœud principal pour fournir les autorisations correctes au fichier de configuration :

   ```
   chmod 600 ~/.ssh/config
   ```

Créer un fichier hosts.

Sur le nœud principal, créez un fichier hosts pour identifier les nœuds du cluster. Le fichier hosts doit avoir une entrée pour chaque nœud du cluster. Créez un fichier ~/hosts et ajoutez chaque nœud en utilisant l'adresse IP privée comme suit :

```
localhost slots=8
```

```
<private ip of node 1> slots=8
<private ip of node 2> slots=8
```

Tests NCCL

> **Note**
>
> Ces tests ont été exécutés à l'aide de la version 1.38.0 d'EFA et du plugin OFI NCCL 1.13.2.

Vous trouverez ci-dessous un sous-ensemble de tests NCCL fournis par Nvidia pour tester à la fois les fonctionnalités et les performances sur plusieurs nœuds de calcul

Instances prises en charge : P3dn, P4, P5, P5e, P5en

Tests de performance

Test de performance NCCL à nœuds multiples sur P4D.24xlarge

Pour vérifier les performances NCCL avec EFA, exécutez le test de performance NCCL standard disponible sur le référentiel officiel des tests NCCL. Le DLAMI est fourni avec ce test déjà conçu pour CUDA XX.X. Vous pouvez également exécuter votre propre script avec EFA.

Lors de la construction de votre propre script, suivez les instructions suivantes :

- Utilisez le chemin complet vers mpirun comme indiqué dans l'exemple lors de l'exécution d'applications NCCL avec EFA.
- Modifiez les paramètres np et N en fonction du nombre d'instances et GPUs de votre cluster.
- Ajoutez l'indicateur NCCL_DEBUG=INFO et assurez-vous que les journaux indiquent l'utilisation de l'EFA sous la forme « Le fournisseur sélectionné est EFA ».
- Définissez l'emplacement du journal d'entraînement à analyser pour validation

```
TRAINING_LOG="testEFA_$(date +"%N").log"
```

Utilisez la commande `watch nvidia-smi` sur n'importe quel nœud de membre pour surveiller l'utilisation des GPU. Les `watch nvidia-smi` commandes suivantes concernent une version générique de CUDA xx.x et dépendent du système d'exploitation de votre instance. Vous pouvez exécuter les commandes pour n'importe quelle version CUDA disponible dans votre EC2 instance Amazon en remplaçant la version CUDA dans le 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 20.04 :

```
$ /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}
```

Le résultat doit être similaire à ce qui suit :

```
# 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/ofi-nccl/share/aws-ofi-
nccl/xml/p4d-24xl-topo.xml
...
-----------------------------some output truncated-----------------------------------
#                                                              out-of-place
          in-place
#       size         count      type   redop    root     time   algbw   busbw #wrong
  time   algbw   busbw #wrong
#        (B)    (elements)                               (us)  (GB/s)  (GB/s)
  (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
```

Tests de validation

Pour vérifier que les tests EFA ont renvoyé un résultat valide, veuillez utiliser les tests suivants pour confirmer :

- Obtenez le type d'instance à l'aide des métadonnées d' EC2 instance :

```
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)
```

- Exécutez le Tests de performance
- Définissez les paramètres suivants

```
CUDA_VERSION
CUDA_RUNTIME_VERSION
NCCL_VERSION
```

- Validez les résultats comme indiqué :

```
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
```

- Pour accéder aux données de référence, nous pouvons analyser la dernière ligne du résultat du tableau issu du test all_reduce à nœuds multiples :

```
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}"
```

Optimisation et surveillance des GPU

La section suivante vous guide à travers les options d'optimisation et de surveillance des GPU. Cette section est conçue comme un flux de travail classique, la surveillance supervisant le prétraitement et la formation.

- Surveillance
  - Moniteur GPUs avec CloudWatch
- Optimisation
  - Prétraitement
  - Entraînement

Surveillance

Votre DLAMI est préinstallé avec plusieurs outils de surveillance du GPU. Ce manuel indique également les outils disponibles afin d'être téléchargés et installés.

- Moniteur GPUs avec CloudWatch- un utilitaire préinstallé qui communique les statistiques d'utilisation du GPU à Amazon CloudWatch.
- Interface de ligne de commande nvidia-smi - utilitaire permettant de l'utilisation globale de mémoire et des fonctions de calcul des GPU. Il est préinstallé sur votre AWS Apprentissage profond (deep learning) AMIs (DLAMI).

- Bibliothèque C NVML - API basée sur C permettant d'accéder directement aux fonctions de surveillance et de gestion des GPU. Elle est utilisée par l'interface de ligne de commande nvidia-smi en arrière-plan et est préinstallée sur vos DLAMI. Elle comporte également les liaisons Perl et Python pour faciliter le développement dans ces langages. L'utilitaire gpumon.py préinstallé sur votre DLAMI utilise le package pynvml de. nvidia-ml-py
- NVIDIA DCGM - outil de gestion des clusters. Visitez la page destinée aux développeurs pour apprendre à installer et à configurer cet outil.

> **Tip**
>
> Consultez le blog des développeurs de NVIDIA pour obtenir les dernières informations sur l'utilisation des outils CUDA installés sur votre DLAMI :
>
> - Surveillance de TensorCore l'utilisation à l'aide de Nsight IDE et de nvprof.

Moniteur GPUs avec CloudWatch

Lorsque vous utilisez vos DLAMI avec un GPU, vous pouvez avoir envie d'effectuer le suivi de l'utilisation lors de la formation ou de l'inférence. Cela peut s'avérer utile pour optimiser votre pipeline de données et régler votre réseau de deep learning.

Il existe deux manières de configurer les métriques du GPU avec CloudWatch :

- Configurer les métriques avec l' AWS CloudWatch agent (recommandé)
- Configurer les métriques avec le script préinstallé `gpumon.py`

Configurer les métriques avec l' AWS CloudWatch agent (recommandé)

Intégrez votre DLAMI à l'agent CloudWatch unifié pour configurer les métriques du GPU et surveiller l'utilisation des coprocessus du GPU dans les instances accélérées Amazon. EC2

Il existe quatre méthodes pour configurer les métriques du GPU avec votre DLAMI :

- Configurer des métriques GPU minimales
- Configurer des métriques partielles du GPU
- Configurer toutes les métriques GPU disponibles

- Configurer des métriques GPU personnalisées

Pour plus d'informations sur les mises à jour et les correctifs de sécurité, voir Correctifs de sécurité pour l'agent AWS CloudWatch

Prérequis

Pour commencer, vous devez configurer les autorisations IAM de l' EC2 instance Amazon qui permettent à votre instance d'envoyer des métriques à CloudWatch. Pour connaître les étapes détaillées, voir Création de rôles et d'utilisateurs IAM à utiliser avec l' CloudWatch agent.

Configurer des métriques GPU minimales

Configurez des métriques GPU minimales à l'aide du `dlami-cloudwatch-agent@minimal` `systemd` service. Ce service configure les métriques suivantes :

- `utilization_gpu`
- `utilization_memory`

Vous pouvez trouver le `systemd` service pour les métriques GPU préconfigurées minimales à l'emplacement suivant :

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-minimal.json
```

Activez et démarrez le `systemd` service à l'aide des commandes suivantes :

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

Configurer des métriques partielles du GPU

Configurez des métriques partielles du GPU à l'aide du `dlami-cloudwatch-agent@partial` `systemd` service. Ce service configure les métriques suivantes :

- `utilization_gpu`
- `utilization_memory`
- `memory_total`
- `memory_used`

- memory_free

Vous pouvez trouver le systemd service pour les métriques GPU partiellement préconfigurées à l'emplacement suivant :

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-partial.json
```

Activez et démarrez le systemd service à l'aide des commandes suivantes :

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

Configurer toutes les métriques GPU disponibles

Configurez toutes les métriques GPU disponibles à l'aide du dlami-cloudwatch-agent@all systemd service. Ce service configure les métriques suivantes :

- 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`

Vous pouvez trouver le `systemd` service pour toutes les métriques GPU préconfigurées disponibles à l'emplacement suivant :

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-all.json
```

Activez et démarrez le `systemd` service à l'aide des commandes suivantes :

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

Configurer des métriques GPU personnalisées

Si les métriques préconfigurées ne répondent pas à vos exigences, vous pouvez créer un fichier de configuration d' CloudWatch agent personnalisé.

Création d'un fichier de configuration personnalisé

Pour créer un fichier de configuration personnalisé, reportez-vous aux étapes détaillées de la section Création ou modification manuelle du fichier de configuration de l' CloudWatch agent.

Pour cet exemple, supposons que la définition du schéma se trouve à`/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`.

Configurez les métriques avec votre fichier personnalisé

Exécutez la commande suivante pour configurer l' CloudWatch agent en fonction de votre fichier personnalisé :

```
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
```

Correctifs de sécurité pour l'agent AWS CloudWatch

Les nouvelles versions DLAMIs sont configurées avec les derniers correctifs de sécurité des AWS CloudWatch agents disponibles. Reportez-vous aux sections suivantes pour mettre à jour votre DLAMI actuel avec les derniers correctifs de sécurité en fonction du système d'exploitation de votre choix.

Amazon Linux 2

yumÀ utiliser pour obtenir les derniers correctifs de sécurité des AWS CloudWatch agents pour un DLAMI Amazon Linux 2.

```
sudo yum update
```

Ubuntu

Pour obtenir les derniers correctifs AWS CloudWatch de sécurité pour un DLAMI avec Ubuntu, il est nécessaire de réinstaller l'agent à AWS CloudWatch l'aide d'un lien de téléchargement Amazon S3.

```
wget https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/
amazon-cloudwatch-agent.deb
```

Pour plus d'informations sur l'installation de l' AWS CloudWatch agent à l'aide des liens de téléchargement Amazon S3, consultez Installation et exécution de l' CloudWatch agent sur vos serveurs.

Configurer les métriques avec le script préinstallé **`gpumon.py`**

Un utilitaire appelée gpumon.py est préinstallé sur vos DLAMI. Il s'intègre CloudWatch et prend en charge la surveillance de l'utilisation par processeur graphique : mémoire du processeur graphique, température du processeur graphique et puissance du processeur graphique. Le script envoie régulièrement les données surveillées à CloudWatch. Vous pouvez configurer le niveau de granularité des données envoyées CloudWatch en modifiant quelques paramètres dans le script. Avant de démarrer le script, vous devez toutefois vous configurer CloudWatch pour recevoir les métriques.

Comment configurer et exécuter la surveillance du GPU avec CloudWatch

1. Créez un utilisateur IAM ou modifiez un utilisateur existant pour disposer d'une politique de publication de la métrique sur. CloudWatch Si vous créez un nouvel utilisateur, notez les informations d'identification. Vous en aurez besoin à l'étape suivante.

   La politique IAM à rechercher est « cloudwatch : PutMetricData ». La stratégie qui est ajoutée est la suivante :

   ```
   {
      "Version": "2012-10-17",
   ```

```
   "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
   ]
}
```

> **Tip**
>
> Pour plus d'informations sur la création d'un utilisateur IAM et l'ajout de politiques pour CloudWatch, consultez la CloudWatch documentation.

2. Sur votre DLAMI, AWS exécutez configure et spécifiez les informations d'identification de l'utilisateur IAM.

```
$ aws configure
```

3. Vous devrez peut-être modifier l'utilitaire gpumon avant de l'exécuter. Vous trouverez l'utilitaire gpumon et le fichier README à l'emplacement défini dans le bloc de code suivant. Pour plus d'informations sur le `gpumon.py` script, consultez l'emplacement du script sur Amazon S3.

```
Folder: ~/tools/GPUCloudWatchMonitor
Files:  ~/tools/GPUCloudWatchMonitor/gpumon.py
       ~/tools/GPUCloudWatchMonitor/README
```

   Options :

   - Si votre instance N'est PAS située dans la région us-east-1, modifiez la région dans gpumon.py.
   - Modifiez d'autres paramètres tels que le CloudWatch `namespace` ou la période de référence avec`store_reso`.

4. Actuellement, le script prend uniquement en charge Python 3. Activez l'environnement Python 3 de votre framework préféré ou activez l'environnement Python 3 général DLAMI.

```
$ source activate python3
```

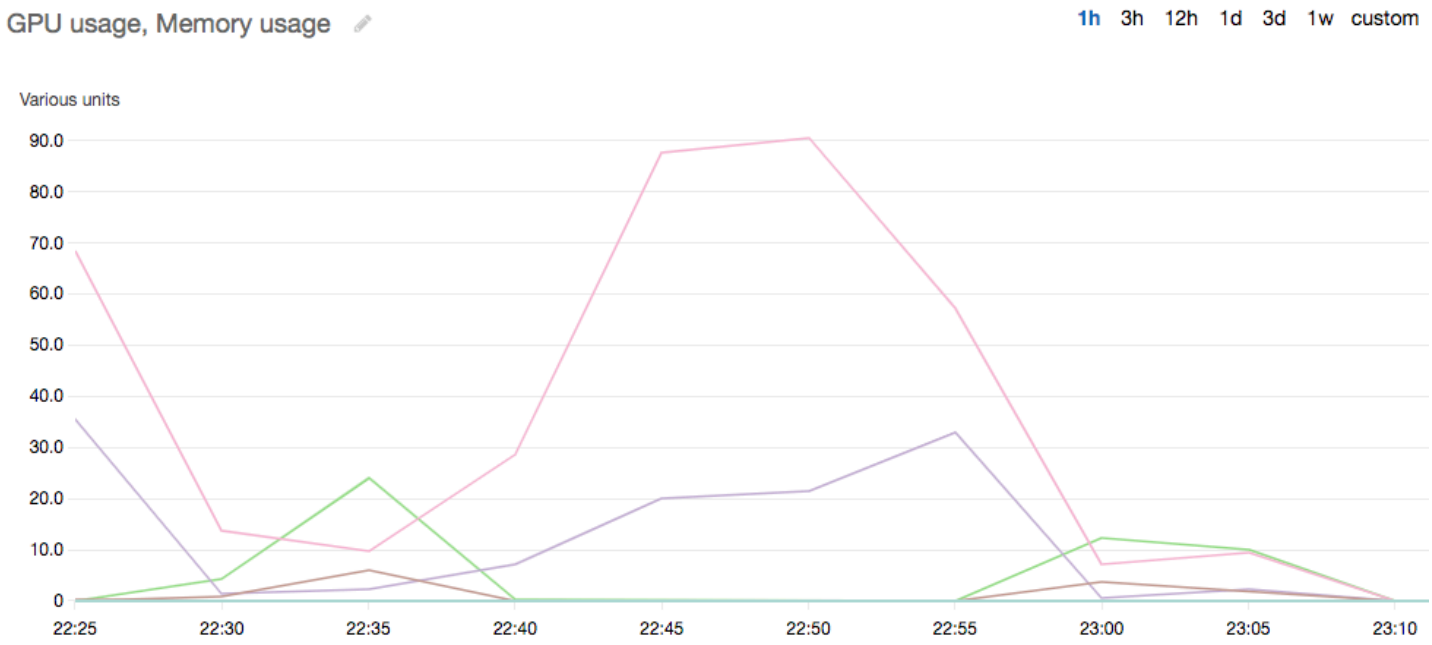
5. Exécutez l'utilitaire gpumon en arrière-plan.

```
(python3)$ python gpumon.py &
```

6. Ouvrez votre navigateur à l'adresse https://console.aws.amazon.com/cloudwatch/, puis sélectionnez la métrique. Il aura un espace de noms « DeepLearningTrain ».

> **Tip**
>
> Vous pouvez modifier l'espace de noms en modifiant gpumon.py. Vous pouvez également modifier l'intervalle de création des rapports en ajustant `store_reso`.

Voici un exemple de CloudWatch graphique illustrant une exécution de gpumon.py surveillant une tâche de formation sur une instance p2.8xlarge.



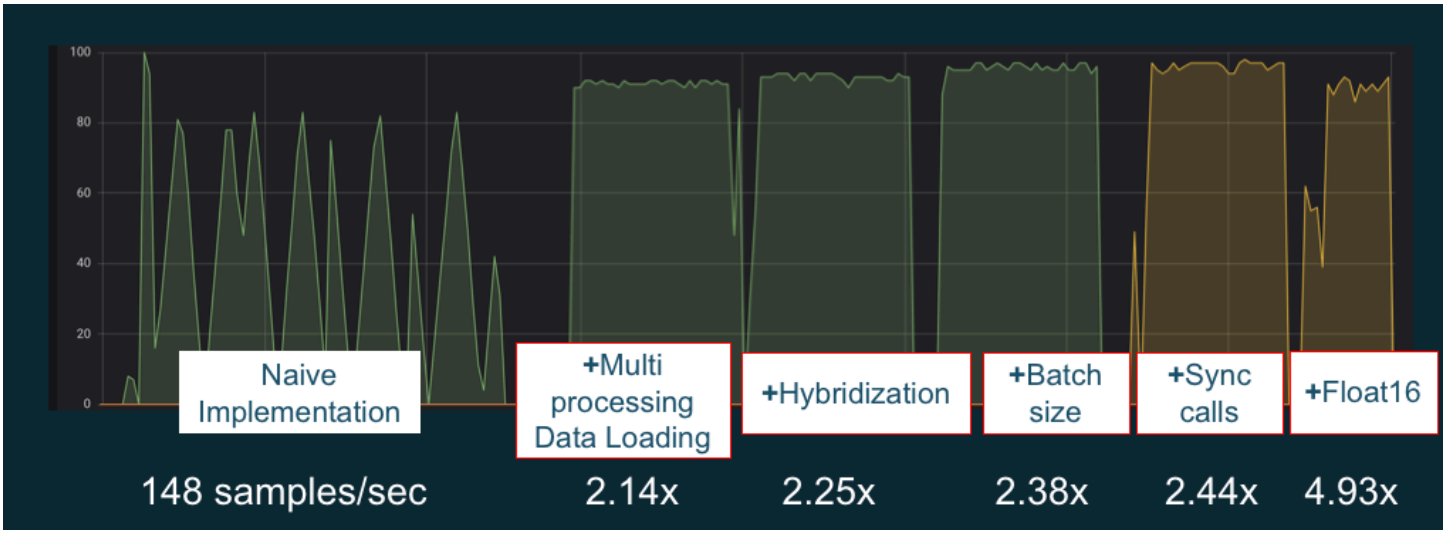
Les rubriques suivantes relatives à la surveillance et l'optimisation des GPU peuvent vous intéresser :

- Surveillance
  - Moniteur GPUs avec CloudWatch
- Optimisation
  - Prétraitement
  - Entraînement

Optimisation

Pour en tirer le meilleur parti GPUs, vous pouvez optimiser votre pipeline de données et ajuster votre réseau de deep learning. Comme décrit dans le graphique suivant, une implémentation naïve ou élémentaire d'un réseau neuronal risque d'utiliser le GPU de manière incohérente, et ne pas exploiter pleinement son potentiel. Lorsque vous optimisez le prétraitement et le chargement des données, vous pouvez réduire le goulot d'étranglement de votre CPU à votre GPU. Vous pouvez ajuster le réseau neuronal lui-même, en utilisant un réseau hybride (lorsqu'il est pris en charge par l'infrastructure), en ajustant la taille des lots et en synchronisant les appels. Vous pouvez également utiliser la formation avec précision multiple (float16 ou int8) dans la plupart des infrastructures, ce qui peut améliorer considérablement le débit.

Le graphique suivant illustre les gains de performance cumulés lorsque différentes optimisations sont appliquées. Vos résultats dépendront des données vous traitez et du réseau que vous optimisez.



Exemples d'optimisations des performances des GPU. Source du graphique : Performance Tricks with MXNet Gluon

Les guides suivants présentent les options qui fonctionneront avec votre DLAMI et vous aideront à améliorer les performances du GPU.

Rubriques

- Prétraitement
- Entraînement

Prétraitement

Le processus de prétraitement des données via des transformations ou des augmentations peut souvent être lié AU CPU. Il peut donc constituer le goulot d'étranglement dans votre pipeline. Les infrastructures comportent des opérateurs pour le traitement des images, mais la bibliothèque DALI (Data Augmentation Library) affiche de meilleures performances que les options intégrées aux infrastructures.

- Bibliothèque NVIDIA Data Augmentation Library (DALI) : DALI décharge l'augmentation de données dans le GPU. Il n'est pas préinstallé sur le DLAMI, mais vous pouvez y accéder en l'installant ou en chargeant un conteneur de framework pris en charge sur votre DLAMI ou sur une autre instance Amazon Elastic Compute Cloud. Reportez-vous à la page de projet DALI sur le site web NVIDIA pour plus de détails. Pour un exemple de cas d'utilisation et pour télécharger des exemples de code, consultez l'exemple de performance d'entraînement en matière de SageMaker prétraitement.
- nvJPEG : bibliothèque de décodage JPEG avec accélération GPU pour les programmeurs C. Elle prend en charge le décodage d'images uniques ou de lots, ainsi que les opérations de transformation suivantes qui sont communes dans le deep learning. nvJPEG est intégré à DALI. Vous pouvez également le télécharger à partir de la page nvjpeg du site web NVIDIA et l'utiliser séparément.

Les rubriques suivantes relatives à la surveillance et l'optimisation des GPU peuvent vous intéresser :

- Surveillance
  - Moniteur GPUs avec CloudWatch
- Optimisation
  - Prétraitement
  - Entraînement

Entraînement

Avec la formation avec précision mixte, vous pouvez déployer de plus grands réseaux avec la même quantité de mémoire, ou réduire l'utilisation de la mémoire par rapport à votre réseau simple ou double précision. Vous obtiendrez aussi de meilleures performances de calcul. Vous pouvez également tirer parti de transferts de données plus petits et plus rapides, qui est un facteur important dans la formation distribuée à plusieurs nœuds. Pour profiter de la formation avec précision mixte,

vous devez ajuster la conversion et la mise à l'échelle de la perte des données. Les guides suivants décrivent la procédure pour les infrastructures qui prennent en charge la précision mixte.

- SDK NVIDIA Deep Learning : documentation sur le site Web de NVIDIA décrivant l'implémentation en précision mixte pour MXNet, et PyTorch. TensorFlow

> **Tip**
>
> Recherchez l'infrastructure de votre choix sur le site web, puis recherchez « précision mixte » ou « fp16 » pour obtenir les dernières techniques d'optimisation. Voici quelques guides relatifs à la précision mixte qui peuvent vous être utiles :
>
> - Entraînement de précision mixte avec TensorFlow (vidéo) - sur le site du blog NVIDIA.
> - Entraînement à précision mixte avec float16 avec MXNet - un article de FAQ sur le site Web. MXNet
> - NVIDIA Apex : un outil pour un entraînement facile à précision mixte grâce PyTorch à un article de blog sur le site Web de NVIDIA.

Les rubriques suivantes relatives à la surveillance et l'optimisation des GPU peuvent vous intéresser :

- Surveillance
  - Moniteur GPUs avec CloudWatch
- Optimisation
  - Prétraitement
  - Entraînement

La puce AWS Inferentia avec DLAMI

AWS Inferentia est une puce d'apprentissage automatique personnalisée conçue AWS que vous pouvez utiliser pour des prédictions d'inférence de haute performance. Pour utiliser la puce, configurez une instance Amazon Elastic Compute Cloud et utilisez le kit de développement logiciel (SDK) AWS Neuron pour appeler la puce Inferentia. Pour offrir aux clients la meilleure expérience Inferentia, Neuron a été intégré au (AWS Apprentissage profond (deep learning) AMIs DLAMI).

Les rubriques suivantes vous montrent comment commencer à utiliser Inferentia avec le DLAMI.

Table des matières

Lancement d'une instance DLAMI avec Neuron AWS

Le dernier DLAMI est prêt à être utilisé AWS avec Inferentia et est fourni avec AWS le package Neuron API. Pour lancer une instance DLAMI, reportez-vous à la section Lancement et configuration d'une instance DLAMI. Une fois que vous avez un DLAMI, suivez les étapes décrites ici pour vous assurer que AWS votre puce Inferentia AWS et vos ressources Neuron sont actives.

Table des matières

Vérification de votre instance

Avant d'utiliser votre instance, vérifiez qu'elle est correctement configurée et configurée avec Neuron.

Identification des AWS dispositifs d'inférence

Pour identifier le nombre d'appareils Inferentia sur votre instance, utilisez la commande suivante :

```
neuron-ls
```

Si des périphériques Inferentia sont attachés à votre instance, votre sortie ressemblera à ce qui suit :

```
+--------+--------+--------+-----------+--------------+
| 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 |
+--------+--------+--------+-----------+--------------+
```

La sortie fournie provient d'une instance INF1.6xLarge et inclut les colonnes suivantes :

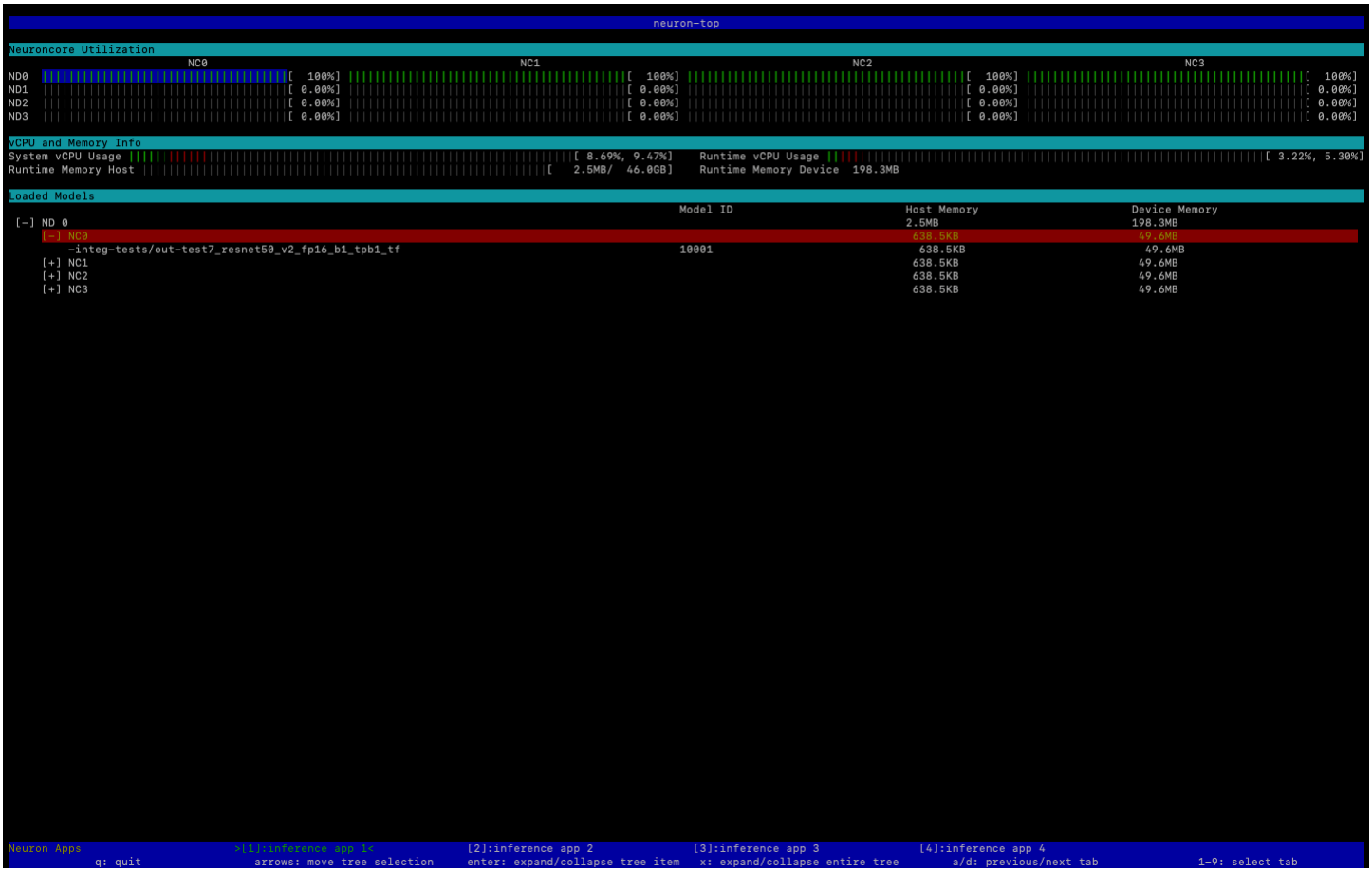
- DISPOSITIF NEURONAL : ID logique attribué au NeuronDevice. Cet ID est utilisé lors de la configuration de plusieurs environnements d'exécution pour en utiliser différents. NeuronDevices
- NOYAUX NEURONAUX : Le nombre de personnes NeuronCores présentes dans le NeuronDevice.
- MÉMOIRE NEURONALE : quantité de mémoire DRAM contenue dans le. NeuronDevice
- APPAREILS CONNECTÉS : Autres appareils NeuronDevices connectés au NeuronDevice.
- PCI BDF : ID de fonction du périphérique de bus PCI (BDF) du. NeuronDevice

Affichage de l'utilisation des ressources

Affichez des informations utiles sur l'utilisation NeuronCore des vCPU, l'utilisation de la mémoire, les modèles chargés et les applications Neuron à l'aide de la commande. `neuron-top` Le lancement `neuron-top` sans arguments affichera les données de toutes les applications d'apprentissage automatique qui les utilisent NeuronCores.

```
neuron-top
```

Lorsqu'une application en utilise quatre NeuronCores, le résultat doit ressembler à l'image suivante :



Pour plus d'informations sur les ressources permettant de surveiller et d'optimiser les applications d'inférence basées sur les neurones, consultez Neuron Tools.

Utilisation de Neuron Monitor (neuron-monitor)

Neuron Monitor collecte des métriques à partir des environnements d'exécution Neuron exécutés sur le système et diffuse les données collectées vers stdout au format JSON. Ces métriques sont organisées en groupes de métriques que vous configurez en fournissant un fichier de configuration. Pour plus d'informations sur Neuron Monitor, consultez le guide de l'utilisateur de Neuron Monitor.

Mise à niveau du logiciel Neuron

Pour plus d'informations sur la mise à jour du logiciel Neuron SDK dans le DLAMI, consultez le guide de configuration de Neuron. AWS

Étape suivante

Utilisation du DLAMI avec Neuron AWS

Utilisation du DLAMI avec Neuron AWS

Un flux de travail typique avec le SDK AWS Neuron consiste à compiler un modèle d'apprentissage automatique préalablement entraîné sur un serveur de compilation. Ensuite, distribuez les artefacts aux instances Inf1 pour exécution. AWS Apprentissage profond (deep learning) AMIs (DLAMI) est préinstallé avec tout ce dont vous avez besoin pour compiler et exécuter l'inférence dans une instance Inf1 qui utilise Inferentia.

Les sections suivantes décrivent comment utiliser le DLAMI avec Inferentia.

Table des matières

Utilisation de TensorFlow -Neuron et du compilateur Neuron AWS

Ce didacticiel montre comment utiliser le compilateur AWS Neuron pour compiler le modèle Keras ResNet -50 et l'exporter en tant que modèle enregistré au format. SavedModel Ce format est un format interchangeable typique des TensorFlow modèles. Vous apprendrez également à exécuter l'inférence sur une instance Inf1 avec un exemple d'entrée.

Pour plus d'informations sur le SDK Neuron, consultez la documentation du SDK AWS Neuron.

Table des matières

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancement d'une instance DLAMI avec Neuron AWS. Vous devez également être familiarisé avec le deep learning et l'utilisation du DLAMI.

Activation de l'environnement Conda

Activez l'environnement TensorFlow -Neuron conda à l'aide de la commande suivante :

```
source activate aws_neuron_tensorflow_p36
```

Pour quitter l'environnement Conda actuel, exécutez la commande suivante :

```
source deactivate
```

Compilation Resnet50

Créez un script Python appelé **`tensorflow_compile_resnet50.py`** ayant le contenu suivant. Ce script Python compile le modèle Keras ResNet 50 et l'exporte en tant que modèle enregistré.

```
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
tfn.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')
```

Compilez le modèle à l'aide de la commande suivante :

```
python tensorflow_compile_resnet50.py
```

Le processus de compilation prendra quelques minutes. Une fois celui-ci terminée, votre sortie devrait ressembler à ce qui suit :

```
...
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
...
```

Après la compilation, le modèle enregistré est compressé dans **`ws_resnet50/resnet50_neuron.zip`**. Décompressez le modèle et téléchargez l'exemple d'image pour l'inférence à l'aide des commandes suivantes :

```
unzip ws_resnet50/resnet50_neuron.zip -d .
curl -O https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg
```

ResNet50 Inférence

Créez un script Python appelé **`tensorflow_infer_resnet50.py`** ayant le contenu suivant. Ce script exécute l'inférence sur le modèle téléchargé à l'aide d'un modèle d'inférence compilé précédemment.

```
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])
```

Exécutez l'inférence sur le modèle à l'aide de la commande suivante :

```
python tensorflow_infer_resnet50.py
```

Le résultat doit être similaire à ce qui suit :

```
...
[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
 'snow_leopard', 0.009290541)]
```

Étape suivante

Utilisation de AWS Neuron Serving TensorFlow

Utilisation de AWS Neuron Serving TensorFlow

Ce didacticiel montre comment construire un graphe et ajouter une étape de compilation AWS Neuron avant d'exporter le modèle enregistré pour l'utiliser avec TensorFlow Serving. TensorFlow Le service est un système de service qui vous permet d'étendre l'inférence sur un réseau. Neuron TensorFlow Serving utilise la même API que le TensorFlow Serving normal. La seule différence est qu'un modèle enregistré doit être compilé pour AWS Inferentia et que le point d'entrée est un nom `tensorflow_model_server_neuron` binaire différent. Le fichier binaire se trouve dans le `/usr/local/bin/tensorflow_model_server_neuron` DLAMI et y est préinstallé.

Pour plus d'informations sur le SDK Neuron, consultez la documentation du SDK AWS Neuron.

Table des matières

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancement d'une instance DLAMI avec Neuron AWS. Vous devez également être familiarisé avec le deep learning et l'utilisation du DLAMI.

Activation de l'environnement Conda

Activez l'environnement TensorFlow -Neuron conda à l'aide de la commande suivante :

```
source activate aws_neuron_tensorflow_p36
```

Si vous devez quitter l'environnement Conda actuel, exécutez :

```
source deactivate
```

Compilation et exportation du modèle enregistré

Créez un script Python appelé `tensorflow-model-server-compile.py` avec le contenu suivant. Ce script construit un graphe et le compile à l'aide de Neuron. Il exporte ensuite le graphe compilé en tant que modèle enregistré.

```
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
modeldir = "./resnet50/1"
tf.saved_model.simple_save(sess, modeldir, inputs, outputs)

# compile the model for Inferentia
neuron_modeldir = os.path.join(os.path.expanduser('~'), 'resnet50_inf1', '1')
tf.neuron.saved_model.compile(modeldir, neuron_modeldir, batch_size=1)
```

Compilez le modèle à l'aide de la commande suivante :

```
python tensorflow-model-server-compile.py
```

Le résultat doit être similaire à ce qui suit :

```
...
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
```

Servir le modèle enregistré

Une fois le modèle compilé, vous pouvez utiliser la commande suivante pour servir le modèle enregistré avec le binaire tensorflow_model_server_neuron :

```
tensorflow_model_server_neuron --model_name=resnet50_inf1 \
    --model_base_path=$HOME/resnet50_inf1/ --port=8500 &
```

Le résultat doit être similaire à ce qui suit. Le modèle compilé est installé dans la DRAM du dispositif Inferentia par le serveur pour préparer l'inférence.

```
...
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 ...
```

Génération de demandes d'inférence au serveur de modèle

Créez un script Python appelé `tensorflow-model-server-infer.py` avec le contenu suivant. Ce script exécute l'inférence via gRPC, qui est une infrastructure de service.

```
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/awslabs/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))
```

Exécutez l'inférence sur le modèle à l'aide de gRPC avec la commande suivante :

```
python tensorflow-model-server-infer.py
```

Le résultat doit être similaire à ce qui suit :

```
[[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
 'snow_leopard', 0.009290541)]]
```

Utilisation de MXNet -Neuron et du compilateur Neuron AWS

L'API de compilation MXNet -Neuron fournit une méthode pour compiler un modèle de graphe que vous pouvez exécuter sur un appareil AWS Inferentia.

Dans cet exemple, vous utilisez l'API pour compiler un modèle ResNet -50 et l'utiliser pour exécuter l'inférence.

Pour plus d'informations sur le SDK Neuron, consultez la documentation du SDK AWS Neuron.

Table des matières

- ResNet50 Inférence

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancement d'une instance DLAMI avec Neuron AWS. Vous devez également être familiarisé avec le deep learning et l'utilisation du DLAMI.

Activation de l'environnement Conda

Activez l'environnement MXNet -Neuron conda à l'aide de la commande suivante :

```
source activate aws_neuron_mxnet_p36
```

Pour quitter l'environnement Conda actuel, exécutez :

```
source deactivate
```

Compilation Resnet50

Créez un script Python appelé **`mxnet_compile_resnet50.py`** avec le contenu suivant. Ce script utilise l'API Python de compilation MXNet -Neuron pour compiler un modèle ResNet -50.

```
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)
```

Compilez le modèle à l'aide de la commande suivante :

```
python mxnet_compile_resnet50.py
```

La compilation prendra quelques minutes. Une fois la compilation terminée, les fichiers suivants se trouveront dans votre répertoire actuel :

```
resnet-50-0000.params
resnet-50-symbol.json
compiled_resnet50-0000.params
compiled_resnet50-symbol.json
```

ResNet50 Inférence

Créez un script Python appelé **`mxnet_infer_resnet50.py`** avec le contenu suivant. Ce script télécharge un exemple d'image qu'il utilise pour exécuter l'inférence avec le modèle compilé.

```
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]))
```

Exécutez l'inférence avec le modèle compilé à l'aide de la commande suivante :

```
python mxnet_infer_resnet50.py
```

Le résultat doit être similaire à ce qui suit :

```
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
```

Étape suivante

Utilisation de MXNet -Neuron Model Serving

Utilisation de MXNet -Neuron Model Serving

Dans ce didacticiel, vous apprendrez à utiliser un MXNet modèle préentraîné pour effectuer une classification d'images en temps réel avec Multi Model Server (MMS). Le MMS est un easy-to-use outil flexible destiné à servir des modèles d'apprentissage profond entraînés à l'aide de n'importe quel

framework d'apprentissage automatique ou d'apprentissage profond. Ce tutoriel inclut une étape de compilation à l'aide de AWS Neuron et une implémentation du MMS à l'aide de. MXNet

Pour plus d'informations sur le SDK Neuron, consultez la documentation du SDK AWS Neuron.

Table des matières

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancement d'une instance DLAMI avec Neuron AWS. Vous devez également être familiarisé avec le deep learning et l'utilisation du DLAMI.

Activation de l'environnement Conda

Activez l'environnement MXNet -Neuron conda à l'aide de la commande suivante :

```
source activate aws_neuron_mxnet_p36
```

Pour quitter l'environnement Conda actuel, exécutez :

```
source deactivate
```

Téléchargement de l'exemple de code

Pour exécuter cet exemple, téléchargez l'exemple de code à l'aide des commandes suivantes :

```
git clone https://github.com/awslabs/multi-model-server
cd multi-model-server/examples/mxnet_vision
```

Compilation du modèle.

Créez un script Python appelé `multi-model-server-compile.py` avec le contenu suivant. Ce script compile le modèle ResNet 50 sur la cible de l'appareil 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)
```

Pour compiler le modèle, utilisez la commande suivante :

```
python multi-model-server-compile.py
```

Le résultat doit être similaire à ce qui suit :

```
...
[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!
```

Créez un fichier nommé `signature.json` avec le contenu suivant pour configurer le nom et la forme de l'entrée :

```
{
  "inputs": [
    {
      "data_name": "data",
      "data_shape": [
        1,
        3,
        224,
        224
      ]
    }
  ]
}
```

Téléchargez le fichier `synset.txt` à l'aide de la commande suivante. Ce fichier est une liste de noms pour les classes de ImageNet prédiction.

```
curl -O https://s3.amazonaws.com/model-server/model_archive_1.0/examples/
squeezenet_v1.1/synset.txt
```

Créez une classe de service personnalisée en suivant le modèle figurant dans le dossier `model_server_template`. Copiez le modèle dans votre répertoire de travail actuel à l'aide de la commande suivante :

```
cp -r ../model_service_template/* .
```

Modifiez le module `mxnet_model_service.py` pour remplacer le contexte `mx.cpu()` par le contexte `mx.neuron()` comme suit. Vous devez également commenter la copie de données inutile `model_input` car MXNet -Neuron ne prend pas en charge le NDArray et 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]
```

Empaquetez le modèle avec l'archiveur de modèle à l'aide des commandes suivantes :

```
cd ~/multi-model-server/examples
model-archiver --force --model-name resnet-50_compiled --model-path mxnet_vision --
handler mxnet_vision_service:handle
```

Exécution de l'inférence

Démarrez le serveur multimodèle et chargez le modèle qui utilise l' RESTful API à l'aide des commandes suivantes. Assurez-vous que neuron-rtd s'exécute avec les paramètres par défaut.

```
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
```

Exécutez l'inférence à l'aide d'un exemple d'image avec les commandes suivantes :

```
curl -O https://raw.githubusercontent.com/awslabs/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
```

Le résultat doit être similaire à ce qui suit :

```
[
  {
    "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"
  }
]
```

Pour effectuer un nettoyage après le test, émettez une commande de suppression via l' RESTful API et arrêtez le serveur de modèles à l'aide des commandes suivantes :

```
curl -X DELETE http://127.0.0.1:8081/models/resnet-50_compiled

multi-model-server --stop
```

Vous devriez voir la sortie suivante :

```
{
  "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
```

Utilisation de PyTorch -Neuron et du compilateur Neuron AWS

L'API de compilation PyTorch -Neuron fournit une méthode pour compiler un modèle de graphe que vous pouvez exécuter sur un appareil AWS Inferentia.

Un modèle formé doit être compilé sur une cible Inferentia avant de pouvoir être déployé sur des instances Inf1. Le didacticiel suivant compile le modèle torchvision ResNet 50 et l'exporte en tant que module enregistré. TorchScript Ce modèle est ensuite utilisé pour exécuter l'inférence.

Pour plus de commodité, ce didacticiel utilise une instance Inf1 pour la compilation et l'inférence. En pratique, vous pouvez compiler votre modèle à l'aide d'un autre type d'instance, par exemple la famille d'instances c5. Vous devez ensuite déployer votre modèle compilé sur le serveur d'inférence Inf1. Pour plus d'informations, consultez la documentation du PyTorch SDK AWS Neuron.

Table des matières

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancement d'une instance DLAMI avec Neuron AWS. Vous devez également être familiarisé avec le deep learning et l'utilisation du DLAMI.

Activation de l'environnement Conda

Activez l'environnement PyTorch -Neuron conda à l'aide de la commande suivante :

```
source activate aws_neuron_pytorch_p36
```

Pour quitter l'environnement Conda actuel, exécutez :

```
source deactivate
```

Compilation Resnet50

Créez un script Python appelé **`pytorch_trace_resnet50.py`** avec le contenu suivant. Ce script utilise l'API Python de compilation PyTorch -Neuron pour compiler un modèle ResNet -50.

> **Note**
>
> Il existe une dépendance entre les versions de Torchvision et le package Torch dont vous devez tenir compte lorsque vous compilez des modèles Torchvision. Ces règles de dépendance peuvent être gérées via pip. Torchvision==0.6.1 correspond à la version torch==1.5.1, tandis que torchvision==0.8.2 correspond à la version 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")
```

Exécutez le script de compilation.

```
python pytorch_trace_resnet50.py
```

La compilation prendra quelques minutes. Lorsque la compilation est terminée, le modèle compilé est enregistré `resnet50_neuron.pt` dans le répertoire local.

ResNet50 Inférence

Créez un script Python appelé **`pytorch_infer_resnet50.py`** avec le contenu suivant. Ce script télécharge un exemple d'image qu'il utilise pour exécuter l'inférence avec le modèle compilé.

```
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/awslabs/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) )
```

Exécutez l'inférence avec le modèle compilé à l'aide de la commande suivante :

```
python pytorch_infer_resnet50.py
```

Le résultat doit être similaire à ce qui suit :

```
Top 5 labels:
 ['tiger', 'lynx', 'tiger_cat', 'Egyptian_cat', 'tabby']
```

Le ARM64 DLAMI

AWS ARM64 DLAMIs Les GPU sont conçus pour fournir des performances élevées et une rentabilité élevées pour les charges de travail liées au deep learning. Plus précisément, le type d'instance G5g intègre le processeur AWS Graviton2 basé sur ARM64, qui a été entièrement conçu AWS et optimisé pour la façon dont les clients exécutent leurs charges de travail dans le cloud. AWS ARM64 DLAMIs Les GPU sont préconfigurés avec Docker, NVIDIA Docker, NVIDIA Driver, CUDA, cuDNN, NCCL, ainsi que des frameworks d'apprentissage automatique populaires tels que et. TensorFlow PyTorch

Avec le type d'instance G5g, vous pouvez tirer parti des avantages en termes de prix et de performances de Graviton2 pour déployer des modèles d'apprentissage profond accélérés par GPU à un coût nettement inférieur à celui des instances x86 avec accélération GPU.

Sélectionnez un ARM64 DLAMI

Lancez une instance G5g avec le ARM64 DLAMI de votre choix.

Pour step-by-step obtenir des instructions sur le lancement d'un DLAMI, reportez-vous à la section Lancement et configuration d'un DLAMI.

Pour obtenir la liste des versions les plus récentes ARM64 DLAMIs, consultez les notes de mise à jour relatives au DLAMI.

Démarrer

Les rubriques suivantes expliquent comment commencer à utiliser le ARM64 DLAMI.

Table des matières

Utilisation du ARM64 PyTorch DLAMI du GPU

AWS Apprentissage profond (deep learning) AMIs Il est prêt à être utilisé avec le processeur Arm64 et est GPUs optimisé pour. PyTorch Le ARM64 PyTorch DLAMI GPU inclut un environnement Python préconfiguré PyTorchavec TorchVision, et pour les cas d'utilisation de l'apprentissage profond, de la formation TorchServeet de l'inférence.

Table des matières

Vérifier l'environnement PyTorch Python

Connectez-vous à votre instance G5g et activez l'environnement Conda de base à l'aide de la commande suivante :

```
source activate base
```

Votre invite de commande doit indiquer que vous travaillez dans l'environnement Conda de base, qui contient PyTorch TorchVision, et d'autres bibliothèques.

```
(base) $
```

Vérifiez les trajectoires d'outils par défaut de l' PyTorch environnement :

```
(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)
```

Exécutez un exemple d'entraînement avec PyTorch

Exécutez un exemple de tâche de formation MNIST :

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

Votre sortie doit ressembler à ce qui suit :

```
...
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%)
```

Exécutez un échantillon d'inférence avec PyTorch

Utilisez les commandes suivantes pour télécharger un modèle densenet161 préentraîné et exécuter l'inférence à l'aide de : 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
```

Votre sortie doit ressembler à ce qui suit :

```
{
  "tiger_cat": 0.4693363308906555,
  "tabby": 0.4633873701095581,
  "Egyptian_cat": 0.06456123292446136,
  "lynx": 0.0012828150065615773,
  "plastic_bag": 0.00023322898778133094
}
```

Utilisez les commandes suivantes pour annuler l'enregistrement du modèle densenet161 et arrêter le serveur :

```
curl -X DELETE http://localhost:8081/models/densenet161/1.0
torchserve --stop
```

Votre sortie doit ressembler à ce qui suit :

```
{
  "status": "Model \"densenet161\" unregistered"
}
TorchServe has stopped.
```

Inférence

Cette section propose des didacticiels sur la façon d'exécuter l'inférence à l'aide des frameworks et des outils du DLAMI.

Outils d'inférence

- TensorFlow Servir

Service de modèle

Les options de service de modèles installées sur l'AMI Deep Learning avec Conda sont les suivantes. Cliquez sur une option pour savoir comment l'utiliser.

Rubriques

- TensorFlow Servir
- TorchServe

TensorFlow Servir

TensorFlow Serving est un système de service flexible et performant pour les modèles d'apprentissage automatique.

Le `tensorflow-serving-api` DLAMI est préinstallé avec un seul cadre DLAMI. Pour utiliser le service Tensorflow, activez d'abord l' TensorFlow environnement.

```
$ source /opt/tensorflow/bin/activate
```

Utilisez ensuite votre éditeur de texte préféré pour créer un script avec le contenu suivant. Nommez-la `test_train_mnist.py`. Ce script est référencé à partir du TensorFlow didacticiel qui formera et évaluera un modèle d'apprentissage automatique par réseau neuronal qui classifie les images.

```
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)
```

À présent, exécutez le script en transmettant en paramètres l'emplacement et le port du serveur ainsi que le nom de fichier de la photo du husky.

```
$ /opt/tensorflow/bin/python3 test_train_mnist.py
```

Soyez patient, car le script peut prendre un certain temps avant de fournir une sortie. Une fois la formation terminée, vous devriez voir ce qui suit :

```
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
```

Autres exemples et fonctions

Si vous souhaitez en savoir plus sur TensorFlow Serving, consultez le TensorFlow site Web.

TorchServe

TorchServe est un outil flexible destiné à servir des modèles de deep learning exportés depuis PyTorch. TorchServe est préinstallé avec l'AMI Deep Learning avec Conda.

Pour plus d'informations sur l'utilisation TorchServe, consultez la PyTorchdocumentation de Model Server.

Rubriques

Servir un modèle de classification d'images sur TorchServe

Ce didacticiel montre comment utiliser un modèle de classification d'images avec TorchServe. Il utilise un modèle DenseNet -161 fourni par PyTorch. Une fois que le serveur est en cours d'exécution, il écoute les demandes de prédiction. Lorsque vous téléchargez une image, dans ce cas, l'image d'un chaton, le serveur renvoie une prédiction des 5 meilleures classes correspondantes parmi les classes sur lesquelles le modèle a été entraîné.

Pour servir un exemple de modèle de classification d'images sur TorchServe

1. Connectez-vous à une instance Amazon Elastic Compute Cloud (Amazon EC2) avec l'AMI Deep Learning avec Conda v34 ou version ultérieure.
2. Activez l'`pytorch_p310`environnement.

```
source activate pytorch_p310
```

3. Clonez le TorchServe référentiel, puis créez un répertoire pour stocker vos modèles.

```
git clone https://github.com/pytorch/serve.git
mkdir model_store
```

4. Archivez le modèle à l'aide de l'archiveur de modèles. Le `extra-files` paramètre utilise un fichier du `TorchServe` dépôt, donc mettez à jour le chemin si nécessaire. Pour plus d'informations sur l'archiveur de modèles, consultez la section Archiveur de modèles Torch pour. 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. Exécutez TorchServe pour démarrer un point de terminaison. L'ajout > `/dev/null` atténue la sortie du journal.

```
torchserve --start --ncs --model-store model_store --models densenet161.mar > /dev/
null
```

6. Téléchargez l'image d'un chaton et envoyez-la au terminal de TorchServe prédiction :

```
curl -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg
curl http://127.0.0.1:8080/predictions/densenet161 -T kitten.jpg
```

Le point de terminaison de prédiction renvoie une prédiction au format JSON similaire aux cinq principales prédictions suivantes, où l'image a une probabilité de 47 % de contenir un chat égyptien, suivie d'une probabilité de 46 % qu'elle ait un chat tigré.

```
{
 "tiger_cat": 0.46933576464653015,
 "tabby": 0.463387668132782,
 "Egyptian_cat": 0.0645613968372345,
 "lynx": 0.0012828196631744504,
 "plastic_bag": 0.00023323058849200606
}
```

7. Lorsque vous avez terminé le test, arrêtez le serveur :

```
torchserve --stop
```

Autres exemples

TorchServe contient de nombreux exemples que vous pouvez exécuter sur votre instance DLAMI. Vous pouvez les consulter sur la page d'exemples du référentiel de TorchServe projets.

Plus d'info

Pour plus de TorchServe documentation, notamment sur la configuration TorchServe avec Docker et les dernières TorchServe fonctionnalités, consultez la page du TorchServe projet sur GitHub.

Mise à niveau de votre DLAMI

Vous trouverez ici des informations sur la mise à niveau de votre DLAMI ainsi que des conseils sur la mise à jour logicielle de votre DLAMI.

Conservez toujours votre système d'exploitation et les autres logiciels installés à jour en appliquant les correctifs et les mises à jour dès qu'ils sont disponibles.

Si vous utilisez Amazon Linux ou Ubuntu, lorsque vous vous connectez à votre DLAMI, vous êtes averti si des mises à jour sont disponibles et consultez les instructions de mise à jour. Pour plus d'informations sur la maintenance d'Amazon Linux, consultez la section Mise à jour du logiciel de l'instance. Pour les instances Ubuntu, reportez-vous à la documentation Ubuntu officielle.

Sur Windows, consultez Windows Update régulièrement pour connaître mises à jour logicielles et de sécurité. Si vous préférez, vous pouvez demander à ce que les mises à jour soient appliquées automatiquement.

> **Important**
>
> Pour plus d'informations sur les vulnérabilités Meltdown et Spectre et sur les correctifs à apporter à votre système d'exploitation pour y remédier, consultez le bulletin AWS de sécurité -2018-013.

Rubriques

Mise à niveau vers une nouvelle version du DLAMI

Les images système de DLAMI sont régulièrement mises à jour pour tirer parti des nouvelles versions du framework d'apprentissage profond, des mises à jour de CUDA et d'autres logiciels, ainsi que de l'optimisation des performances. Si vous utilisez un DLAMI depuis un certain temps et que vous souhaitez profiter d'une mise à jour, vous devez lancer une nouvelle instance. Vous devez aussi

transférer manuellement les ensembles de données, les points de contrôle, ou les autres données précieuses. Vous pouvez plutôt utiliser Amazon EBS pour conserver vos données et les associer à un nouveau DLAMI. Ainsi, vous pouvez mettre à niveau souvent, tout en minimisant le temps nécessaire pour transférer vos données.

> **Note**
>
> Lorsque vous attachez et déplacez des volumes Amazon EBS entre DLAMIs eux, vous devez placer le volume DLAMIs et le nouveau volume dans la même zone de disponibilité.

1. Utilisez Amazon EC2console pour créer un nouveau volume Amazon EBS. Pour obtenir des instructions détaillées, consultez la section Création d'un volume Amazon EBS.
2. Joignez le volume Amazon EBS que vous venez de créer à votre DLAMI existant. Pour obtenir des instructions détaillées, consultez Joindre un volume Amazon EBS.
3. Transférez vos données, comme des ensembles de données, des points de contrôle et des fichiers de configuration.
4. Lancez un DLAMI. Pour obtenir des instructions complètes, consultez Configuration d'une instance DLAMI.
5. Détachez le volume Amazon EBS de votre ancien DLAMI. Pour obtenir des instructions détaillées, consultez la section Détacher un volume Amazon EBS.
6. Attachez le volume Amazon EBS à votre nouveau DLAMI. Suivez les instructions de l'étape 2 pour attacher le volume.
7. Après avoir vérifié que vos données sont disponibles sur votre nouveau DLAMI, arrêtez et résiliez votre ancien DLAMI. Pour obtenir des instructions de nettoyage, consultez Nettoyage d'une instance DLAMI.

Astuces pour les mises à jour de logiciels

De temps à autre, vous souhaiterez peut-être mettre à jour manuellement le logiciel de votre DLAMI. Il est généralement recommandé d'utiliser `pip` pour mettre à jour les packages Python. Vous devez également utiliser `pip` pour mettre à jour les packages dans un environnement Conda sur l'AMI Deep Learning avec Conda. Reportez-vous au site Web de l'infrastructure ou du logiciel pour obtenir des instructions pour la mise à niveau et l'installation.

> **Note**
>
> Nous ne pouvons pas garantir le succès de la mise à jour d'un package. Toute tentative de mise à jour d'un package dans un environnement présentant des dépendances incompatibles peut entraîner un échec. Dans ce cas, vous devez contacter le responsable de la bibliothèque pour voir s'il est possible de mettre à jour les dépendances du package. Vous pouvez également essayer de modifier l'environnement de manière à autoriser la mise à jour. Cependant, cette modification impliquera probablement la suppression ou la mise à jour de packages existants, ce qui signifie que nous ne pouvons plus garantir la stabilité de cet environnement.

Il AWS Apprentissage profond (deep learning) AMIs est livré avec de nombreux environnements Conda et de nombreux packages préinstallés. En raison du nombre de packages préinstallés, il est difficile de trouver un ensemble de packages dont la compatibilité est garantie. Vous pouvez voir un avertissement « L'environnement est incohérent, veuillez vérifier attentivement le plan de package ». Le DLAMI garantit que tous les environnements fournis par le DLAMI sont corrects, mais ne peut garantir que les packages installés par l'utilisateur fonctionneront correctement.

Recevez des notifications sur les nouvelles mises à jour

> **Note**
>
> AWS Le Deep Learning AMIs publie les correctifs de sécurité à une cadence hebdomadaire. Des notifications de publication seront envoyées pour ces correctifs de sécurité incrémentiels, mais ils peuvent ne pas être inclus dans les notes de publication officielles.

Vous pouvez recevoir des notifications chaque fois qu'un nouveau DLAMI est publié. Les notifications sont publiées sur Amazon SNS à l'aide de la rubrique suivante.

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

Des messages sont publiés ici lorsqu'un nouveau DLAMI est publié. La version, les métadonnées et les ID d'AMI régionaux de l'AMI seront inclus dans le message.

Ces messages peuvent être reçus à l'aide de différentes méthodes. Nous vous recommandons d'utiliser la méthode suivante.

1. Ouvrez la console Amazon SNS.
2. Dans la barre de navigation, remplacez la AWS région par US West (Oregon), si nécessaire. Vous devez sélectionner la région dans laquelle la notification SNS à laquelle vous êtes abonné a été créée.
3. Dans le volet de navigation, choisissez Abonnements, puis Créer un abonnement.
4. Dans la boîte de dialogue Créer un abonnement, procédez comme suit :

   a. Pour l'ARN du sujet, copiez et collez le nom de ressource Amazon (ARN) suivant : **`arn:aws:sns:us-west-2:767397762724:dlami-updates`**

   b. Pour le protocole, choisissez-en un parmi [Amazon SQS, AWS Lamda, Email, Email-JSON]

   c. Pour Endpoint, entrez l'adresse e-mail ou le nom de ressource Amazon (ARN) de la ressource que vous utiliserez pour recevoir les notifications.

   d. Choisissez Create subscription (Créer un abonnement).
5. Vous recevez un e-mail de confirmation dont l'objet est AWS Notification - Confirmation d'abonnement. Ouvrez l'e-mail et choisissez Confirm subscription (Confirmer l'abonnement) pour terminer votre abonnement.

Sécurité dans AWS Apprentissage profond (deep learning) AMIs

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui s'exécute Services AWS dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS Apprentissage profond (deep learning) AMIs, voir AWS Services concernés par programme de conformitéAWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par Service AWS ce que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre entreprise, ainsi que la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation du DLAMI. Les rubriques suivantes expliquent comment configurer le DLAMI pour répondre à vos objectifs de sécurité et de conformité. Vous apprenez également à utiliser d'autres outils Services AWS qui vous aident à surveiller et à sécuriser vos ressources DLAMI.

Pour plus d'informations, consultez la section Sécurité sur Amazon EC2 dans le guide de EC2 l'utilisateur Amazon.

Rubriques

- AWS Apprentissage profond (deep learning) AMIs Instances de surveillance

Protection des données dans AWS Apprentissage profond (deep learning) AMIs

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans AWS Apprentissage profond (deep learning) AMIs. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec DLAMI ou Services AWS autre à l'aide de la console, de l'API AWS CLI ou. AWS SDKs Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Gestion des identités et des accès pour AWS Apprentissage profond (deep learning) AMIs

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources DLAMI. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Pour plus d'informations sur la gestion des identités et des accès, consultez Gestion des identités et des accès pour Amazon EC2.

Rubriques

- Authentification par des identités
- Gestion des accès à l'aide de politiques
- IAM avec Amazon EMR

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec

des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez AWS Signature Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour plus d'informations, consultez Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Authentification multifactorielle AWS dans IAM dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez Tâches nécessitant les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme telles que des mots de passe

et des clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons d'effectuer une rotation des clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Pour assumer temporairement un rôle IAM dans le AWS Management Console, vous pouvez passer d'un rôle d'utilisateur à un rôle IAM (console). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .

- Autorisations d’utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d’autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d’un compte différent d’accéder aux ressources de votre compte. Les rôles constituent le principal moyen d’accorder l’accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l’accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l’utilisateur IAM.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d’appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d’autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d’accès.
  - Rôle de service : il s’agit d’un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d’IAM. Pour plus d’informations, consultez Création d’un rôle pour la délégation d’autorisations à un Service AWS dans le Guide de l’utilisateur IAM.
  - Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d’effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des demandes AWS CLI d' AWS API. Cela est préférable au stockage des

clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le guide de l'utilisateur IAM.

Gestion des accès à l’aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d’informations sur la structure et le contenu des documents de politique JSON, consultez Vue d’ensemble des politiques JSON dans le Guide de l’utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C’est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d’aucune autorisation. Pour octroyer aux utilisateurs des autorisations d’effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L’administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d’une action, quelle que soit la méthode que vous utilisez pour exécuter l’opération. Par exemple, supposons que vous disposiez d’une politique qui autorise l’action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l’identité

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un groupe d’utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur

l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les Organizations SCPs, voir Politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.
- Politiques de contrôle des ressources (RCPs) : RCPs politiques JSON que vous pouvez utiliser pour définir le maximum d'autorisations disponibles pour les ressources de vos comptes sans mettre à jour les politiques IAM associées à chaque ressource que vous possédez. Le RCP limite les autorisations pour les ressources des comptes membres et peut avoir un impact sur les autorisations effectives pour les identités, y compris Utilisateur racine d'un compte AWS, qu'elles appartiennent ou non à votre organisation. Pour plus d'informations sur les Organizations RCPs, y compris une liste de ces Services AWS supports RCPs, consultez la section Resource control policies (RCPs) dans le guide de AWS Organizations l'utilisateur.
- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations

peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

IAM avec Amazon EMR

Vous pouvez utiliser IAM avec Amazon EMR pour définir les utilisateurs AWS , les ressources, les groupes, les rôles et les politiques. Vous pouvez également contrôler les utilisateurs et rôles auxquels Services AWS ces utilisateurs et rôles peuvent accéder.

Pour plus d'informations sur l'utilisation d'IAM avec Amazon EMR,AWS Identity and Access Management consultez Amazon EMR.

Validation de conformité pour AWS Apprentissage profond (deep learning) AMIs

Des auditeurs tiers évaluent la sécurité et AWS Apprentissage profond (deep learning) AMIs la conformité de plusieurs programmes de AWS conformité. Pour plus d'informations sur les programmes de conformité pris en charge, consultez la section Validation de conformité pour Amazon EC2.

Pour obtenir la liste Services AWS des programmes de conformité spécifiques, voir AWS Services concernés par programme de conformitéAWS . Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact dans. AWS Artifact

Lorsque vous utilisez le DLAMI, votre responsabilité en matière de conformité dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise et des lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides démarrage rapide de la sécurité et de la conformité. Ces guides de déploiement traitent des considérations architecturales et fournissent des étapes pour déployer des environnements de base axés sur la sécurité et la conformité sur AWS.
- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- Évaluation des ressources à l'aide des AWS Config règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos AWS ressources et vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité.

Résilience dans AWS Apprentissage profond (deep learning) AMIs

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

Pour plus d'informations sur les EC2 fonctionnalités d'Amazon destinées à répondre à vos besoins en matière de résilience et de sauvegarde des données, consultez Resilience in Amazon EC2 dans le guide de EC2 l'utilisateur Amazon.

Sécurité de l'infrastructure dans AWS Apprentissage profond (deep learning) AMIs

La sécurité de l'infrastructure de AWS Apprentissage profond (deep learning) AMIs est soutenue par Amazon EC2. Pour plus d'informations, consultez la section Sécurité de l'infrastructure dans Amazon EC2 dans le guide de EC2 l'utilisateur Amazon.

AWS Apprentissage profond (deep learning) AMIs Instances de surveillance

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances de votre AWS Apprentissage profond (deep learning) AMIs instance et de vos autres AWS solutions. Votre instance DLAMI est fournie avec plusieurs outils de surveillance du GPU, notamment un utilitaire qui transmet les statistiques d'utilisation du GPU à Amazon. CloudWatch Pour plus d'informationsOptimisation et surveillance des GPU, consultez la section Surveiller les EC2 ressources Amazon dans le guide de EC2 l'utilisateur Amazon.

Désactiver le suivi de l'utilisation pour les instances DLAMI

Les distributions de systèmes AWS Apprentissage profond (deep learning) AMIs d'exploitation suivantes incluent du code qui permet de AWS collecter le type d'instance, l'ID d'instance, le type de DLAMI et les informations du système d'exploitation.

> **Note**
>
> AWS ne collecte ni ne conserve aucune autre information concernant le DLAMI, comme les commandes que vous utilisez dans le DLAMI.

- Amazon Linux 2
- Amazon Linux 2023
- Ubuntu 20.04
- Ubuntu 22.04

Pour désactiver le suivi de l'utilisation

Si vous le souhaitez, vous pouvez désactiver le suivi de l'utilisation pour une nouvelle instance DLAMI. Pour vous désinscrire, vous devez ajouter un tag à votre EC2 instance Amazon lors du lancement. La balise doit utiliser la clé dont `OPT_OUT_TRACKING` la valeur associée est définie sur`true`. Pour plus d'informations, consultez la section Marquer vos EC2 ressources Amazon dans le guide de EC2 l'utilisateur Amazon.

Politique de support DLAMI

Vous trouverez ici les détails de la politique de support pour AWS Apprentissage profond (deep learning) AMIs (DLAMI).

Pour obtenir la liste des frameworks DLAMI et des systèmes AWS d'exploitation actuellement pris en charge, consultez la page Politique de support DLAMI. La terminologie suivante s'applique à tout ce qui est DLAMIs mentionné sur la page de politique de support et sur cette page :

- La version actuelle spécifie la version du framework au format x.y.z. Dans ce format, x fait référence à la version majeure, y à la version mineure et z à la version du correctif. Par exemple, pour la version TensorFlow 2.10.1, la version majeure est 2, la version mineure est 10 et la version du correctif est 1.
- La fin du correctif indique la durée pendant laquelle une version de framework ou de système d'exploitation spécifique est prise en AWS charge.

Pour des informations détaillées sur des sujets spécifiques DLAMIs, voirNotes de AMIs mise à jour du Deep Learning.

Support DLAMI FAQs

- Quelles versions du framework reçoivent des correctifs de sécurité ?
- Quels systèmes d'exploitation reçoivent des correctifs de sécurité ?
- Quelles images sont AWS publiées lorsque de nouvelles versions du framework sont publiées ?
- Quelles images bénéficient de nouvelles AWS fonctionnalités d' SageMaker intelligence artificielle ?
- Comment est définie la version actuelle dans le tableau Supported Frameworks ?
- Et si j'utilise une version qui ne figure pas dans le tableau des versions prises en charge ?
- Les versions de correctif précédentes d'une version du framework sont-elles prises en DLAMIs charge ?
- Comment puis-je trouver la dernière image corrigée pour une version de framework prise en charge ?
- À quelle fréquence les nouvelles images sont-elles publiées ?

Quelles versions du framework reçoivent des correctifs de sécurité ?

Si la version du framework figure sous Versions de framework prises en charge dans le tableau des politiques de AWS Apprentissage profond (deep learning) AMIs support, elle reçoit des correctifs de sécurité.

Quels systèmes d'exploitation reçoivent des correctifs de sécurité ?

Si le système d'exploitation est répertorié sous Versions de système d'exploitation prises en charge dans le tableau des politiques de AWS Apprentissage profond (deep learning) AMIs support, il reçoit des correctifs de sécurité.

Quelles images sont AWS publiées lorsque de nouvelles versions du framework sont publiées ?

Nous publions de nouvelles DLAMIs peu de temps après la publication de nouvelles versions de TensorFlow et PyTorch de leur publication. Cela inclut les versions majeures, les versions majeures et mineures et les major-minor-patch versions des frameworks. Nous mettons également à jour les images lorsque de nouvelles versions de pilotes et de bibliothèques sont disponibles. Pour plus d'informations sur la maintenance des images, voir Quand le support actif pour ma version de framework prend-il fin ?

Quelles images bénéficient de nouvelles AWS fonctionnalités d' SageMaker intelligence artificielle ?

Les nouvelles fonctionnalités sont généralement publiées dans la dernière version de DLAMIs for PyTorch et TensorFlow. Reportez-vous aux notes de publication relatives à une image spécifique pour plus de détails sur les nouvelles AWS fonctionnalités ou l' SageMaker IA. Pour obtenir la liste des versions disponibles DLAMIs, consultez les notes de version relatives au DLAMI. Pour plus d'informations sur la maintenance des images, voir Quand le support actif pour ma version de framework prend-il fin ?

Comment est définie la version actuelle dans le tableau Supported Frameworks ?

La version actuelle du tableau AWS Apprentissage profond (deep learning) AMIs Support Policy fait référence à la dernière version du framework disponible sur GitHub. AWS Chaque dernière version inclut des mises à jour des pilotes, des bibliothèques et des packages pertinents du DLAMI. Pour plus d'informations sur la maintenance des images, voir Quand le support actif pour ma version de framework prend-il fin ?

Et si j'utilise une version qui ne figure pas dans le tableau des versions prises en charge ?

Si vous utilisez une version qui ne figure pas dans le tableau des politiques de AWS Apprentissage profond (deep learning) AMIs support, il se peut que vous ne disposiez pas des pilotes, bibliothèques et packages appropriés les plus récents. Pour une up-to-date version ultérieure, nous vous recommandons de passer à l'un des frameworks ou systèmes d'exploitation pris en charge disponibles à l'aide du DLAMI le plus récent de votre choix. Pour obtenir la liste des versions disponibles DLAMIs, consultez les notes de version relatives au DLAMI.

Les versions de correctif précédentes d'une version du framework sont-elles prises en DLAMIs charge ?

Non. Nous prenons en charge la dernière version de correctif de la dernière version majeure de chaque framework publiée 365 jours après sa GitHub publication initiale, comme indiqué dans le tableau des politiques de AWS Apprentissage profond (deep learning) AMIs support. Pour de plus amples informations, consultez Et si j'utilise une version qui ne figure pas dans le tableau des versions prises en charge ?.

Comment puis-je trouver la dernière image corrigée pour une version de framework prise en charge ?

Pour utiliser un DLAMI avec la dernière version du framework, vous pouvez utiliser les paramètres AWS CLI ou SSM pour récupérer l'ID DLAMI et l'utiliser pour lancer le DLAMI à l'aide de la console. EC2 Pour des exemples de commandes de paramètres AWS CLI ou SSM permettant de récupérer l' AWS Apprentissage profond (deep learning) AMIs ID, reportez-vous à la page des notes de version du DLAMI (notes de version du DLAMI à structure unique). La version du framework que vous choisissez doit être répertoriée sous Versions du framework prises en charge dans le tableau des politiques de AWS Apprentissage profond (deep learning) AMIs support.

À quelle fréquence les nouvelles images sont-elles publiées ?

Fournir des versions de correctifs mises à jour est notre priorité absolue. Nous créons régulièrement des images corrigées dès que possible. Nous surveillons les nouvelles versions du framework corrigées (ex. TensorFlow 2.9 à TensorFlow 2.9.1) et les nouvelles versions mineures (ex. TensorFlow 2.9 à TensorFlow 2.10) et les rendre disponibles dès que possible. Lorsqu'une version existante de TensorFlow est publiée avec une nouvelle version de CUDA, nous publions un nouveau DLAMI pour cette version de avec prise en charge de la nouvelle version TensorFlow de CUDA.

Mon instance sera-t-elle mise en place pendant que ma charge de travail est en cours d'exécution ?

Non. Les mises à jour des correctifs pour DLAMI ne sont pas des mises à jour « sur place ».

Vous devez activer une nouvelle EC2 instance, migrer vos charges de travail et vos scripts, puis désactiver votre instance précédente.

Que se passe-t-il lorsqu'une nouvelle version du framework corrigée ou mise à jour est disponible ?

Pour être informé des modifications apportées au DLAMI, veuillez vous abonner aux notifications relatives au DLAMI concerné, voir Recevoir des notifications lors de nouvelles mises à jour.

Les dépendances sont-elles mises à jour sans modifier la version du framework ?

Nous mettons à jour les dépendances sans modifier la version du framework. Cependant, si une mise à jour de dépendance entraîne une incompatibilité, nous créons une image avec une version différente. Assurez-vous de consulter les notes de version relatives au DLAMI pour obtenir des informations de dépendance mises à jour.

Quand le support actif pour ma version de framework prend-il fin ?

Les images DLAMI sont immuables. Une fois créés, ils ne changent pas. Quatre raisons principales peuvent expliquer la fin du support actif pour une version du framework :

- Mises à niveau de la version du framework (correctif)
- AWS correctifs de sécurité
- Date de fin de mise à jour (Aging out)
- Dépendance end-of-support

> **Note**
>
> En raison de la fréquence des mises à niveau des versions et des correctifs de sécurité, nous vous recommandons de consulter régulièrement la page des notes de publication de votre DLAMI et de procéder à une mise à niveau lorsque des modifications sont apportées.

Mises à niveau de la version du framework (correctif)

Si vous disposez d'une charge de travail DLAMI basée TensorFlow sur la version 2.7.0 TensorFlow et que vous publiez la version 2.7.1, publiez AWS une nouvelle DLAMI avec GitHub la version 2.7.1. TensorFlow Les images précédentes avec 2.7.0 ne sont plus activement maintenues une fois que la nouvelle image avec TensorFlow 2.7.1 est publiée. Le DLAMI doté de la version 2.7.0 ne TensorFlow reçoit pas d'autres correctifs. La page des notes de mise à jour du DLAMI TensorFlow pour la version 2.7 est ensuite mise à jour avec les dernières informations. Il n'existe pas de page de note de publication individuelle pour chaque correctif mineur.

Les nouveaux DLAMIs produits créés à la suite de mises à niveau de correctifs sont désignés par un nouvel ID d'AMI.

AWS correctifs de sécurité

Si vous avez une charge de travail basée sur une image avec la version TensorFlow 2.7.0 et que AWS vous créez un correctif de sécurité, une nouvelle version du DLAMI est publiée pour la version 2.7.0. TensorFlow La version précédente des images avec TensorFlow 2.7.0 n'est plus activement maintenue. Pour plus d'informations, voir Mon instance sera-t-elle mise en place pendant que ma charge de travail est en cours d'exécution ? Pour savoir comment trouver le DLAMI le plus récent, voir Comment puis-je trouver la dernière image corrigée pour une version de framework prise en charge ?

Les nouveaux DLAMIs produits créés à la suite de mises à niveau de correctifs sont désignés par un nouvel ID d'AMI.

Date de fin de mise à jour (Aging out)

DLAMIs ont atteint leur date de fin de mise à jour 365 jours après la date GitHub de sortie.

Pour le multi-framework DLAMIs, lorsque l'une des versions du framework est mise à jour, un nouveau DLAMI avec la version mise à jour est requis. Le DLAMI avec l'ancienne version du framework n'est plus activement maintenu.

> **Important**
>
> Nous faisons une exception en cas de mise à jour majeure du framework. Par exemple, si la version TensorFlow 1.15 est mise à jour vers la version TensorFlow 2.0, nous continuons à prendre en charge la version la plus récente de la version TensorFlow 1.15 pendant une période de deux ans à compter de la date de GitHub sortie ou six mois après l'arrêt du support par l'équipe de maintenance du framework d'origine, la date la plus proche étant retenue.

Dépendance end-of-support

Si vous exécutez une charge de travail sur une image TensorFlow DLAMI 2.7.0 avec Python 3.6 et que cette version de Python est end-of-support sélectionnée pour, toutes les images DLAMI basées sur Python 3.6 ne seront plus activement maintenues. De même, si une version du système d'exploitation telle qu'Ubuntu 16.04 est marquée pour end-of-support, toutes les images DLAMI qui dépendent d'Ubuntu 16.04 ne seront plus activement maintenues.

Les images dont les versions du framework ne sont plus activement maintenues seront-elles corrigées ?

Non. Les images qui ne sont plus activement maintenues ne seront pas publiées dans de nouvelles versions.

Comment utiliser une ancienne version du framework ?

Pour utiliser un DLAMI avec une ancienne version du framework, récupérez l'ID DLAMI et utilisez-le pour lancer le DLAMI à l'aide de la console. EC2 Pour les commandes AWS CLI permettant de récupérer l'ID de l'AMI, reportez-vous à la page des notes de version des notes de version du DLAMI à structure unique.

Comment puis-je suivre les modifications apportées up-to-date aux frameworks et à leurs versions ?

Restez fidèle up-to-date aux frameworks et aux versions du DLAMI à l'aide du tableau AWS Apprentissage profond (deep learning) AMIs Framework Support Policy et des notes de version du DLAMI.

Ai-je besoin d'une licence commerciale pour utiliser le référentiel Anaconda ?

Anaconda est passée à un modèle de licence commerciale pour certains utilisateurs. Maintenues activement, DLAMIs elles ont été migrées vers la version open source accessible au public de Conda (conda-forge) depuis le canal Anaconda.

Tableau des politiques de support DLAMI

Pour plus de détails, consultez la Politique de Support.

Versions du framework prises en charge

Framework	Version actuelle	Version CUDA	GitHub GA	Fin du patch
PyTorch	2.7.0	12,8	23/04/2025	23/04/2026/
PyTorch	2.6.0	12.6	29-01-2025	29-01-2026/
PyTorch	2.5.1	12.4	24/11/2024	24/11/2025
PyTorch	2.4.1	12.4	24-07-2024	24/07/2025
TensorFlow	2.18.0	12,5	24/10/2024	24/10/2025
TensorFlow	2.17.0	12.3	07/11/2024	07/11/2025

Versions de système d'exploitation prises en charge

Système d'exploitation	Fin du patch
Amazon Linux 2023	30/06/2029-06
Amazon Linux 2	30/06/2026/
Ubuntu 24.04	30/04/2029-04
Ubuntu 22.04	30/04/2027-04

Versions du framework non prises en charge

Les versions répertoriées dans ce tableau apparaîtront pendant 2 ans après leur date de support.

Framework	Version actuelle	Version CUDA	GitHub GA	Fin du patch
PyTorch	2.3.0	12.1	24-04-2024	24/04/2025
PyTorch	2.2.0	12.1	30-01-2024	30-01-2025
PyTorch	1.13.1	11.7	28/10	28/10/2024
PyTorch	2.1.0	12.1	04/10/2023	04/10/2024
PyTorch	2.0.0	12.1	15 avril	15/03/2024
PyTorch	1.12.1	11.6	05.07-01	01/07/2023
PyTorch	1.11.0	11.5	10/03/2018	10 %
TensorFlow	2.16.0	12.3	07-03-2024	07/03/2025
TensorFlow	2.15.0	12.2	14/11/2023	14-11-2024
TensorFlow	2.13.0	11.8	19/07/2023	19/07/2024
TensorFlow	2.12.0	11.8	23 avril	23/03/2024
TensorFlow	2.11.0	11.2	11/18	18/11/2023
TensorFlow	2.10.1	11.2	05.09-06	06/09/2023
TensorFlow	2.9.3	11.2	17/05/2018	17/05/2023

Versions de système d'exploitation non prises en charge

Système d'exploitation	Fin du patch
Ubuntu 20.04	31/05/2025
Ubuntu 18.04	31/05/2023

Archive des notes de mise à jour du DLAMI non prise en charge

Base

GPU

- AWS AMI de base d'apprentissage profond (Ubuntu 20.04)
- AWS AMI de base d'apprentissage profond (Ubuntu 18.04)

Cadre unique

PyTorch AMI spécifique

GPU

- AWS GPU AMI PyTorch 2.3 pour apprentissage profond (Ubuntu 20.04)
- AWS GPU AMI PyTorch 2.3 pour apprentissage profond (Amazon Linux 2)
- AWS GPU ARM64 AMI PyTorch 2.3 pour apprentissage profond (Ubuntu 22.04)
- AWS GPU AMI PyTorch 2.2 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI PyTorch 2.2 pour apprentissage profond (Ubuntu 20.04)
- AWS GPU AMI PyTorch 2.1 pour apprentissage profond (Ubuntu 20.04)
- AWS GPU AMI PyTorch 2.0 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI PyTorch 2.0 pour apprentissage profond (Ubuntu 20.04)
- AWS GPU AMI PyTorch 1.13 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond PyTorch 1.13 (Ubuntu 20.04)
- AWS GPU AMI PyTorch 1.12 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond PyTorch 1.12 (Ubuntu 20.04)
- AWS GPU AMI PyTorch 1.11 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond PyTorch 1.11 (Ubuntu 20.04)
- AWS Processeur graphique AMI pour apprentissage profond PyTorch 1.10 (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond PyTorch 1.10 (Ubuntu 20.04)
- AWS GPU Graviton avec AMI d'apprentissage profond PyTorch 1.10 (Ubuntu 20.04)
- AWS GPU AMI d'apprentissage profond PyTorch 1.10 (Ubuntu 18.04)

- AWS GPU AMI PyTorch 1.9 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond PyTorch 1.9 (Ubuntu 20.04)
- AWS GPU AMI d'apprentissage profond PyTorch 1.9 (Ubuntu 18.04)

TensorFlow AMI spécifique

GPU

- AWS Processeur graphique AMI TensorFlow 2.16 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.16 (Ubuntu 20.04)
- AWS Processeur graphique AMI TensorFlow 2.15 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.15 (Ubuntu 20.04)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.13 (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.13 (Ubuntu 20.04)
- AWS GPU AMI TensorFlow 2.12 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.12 (Ubuntu 20.04)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.11 (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.11 (Ubuntu 20.04)
- AWS Processeur graphique AMI TensorFlow 2.10 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.10 (Ubuntu 20.04)
- AWS GPU AMI TensorFlow 2.9 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.9 (Ubuntu 20.04)
- AWS GPU AMI TensorFlow 2.8 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.8 (Ubuntu 20.04)
- AWS GPU AMI TensorFlow 2.7 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.7 (Ubuntu 20.04)
- AWS GPU AMI TensorFlow 2.6 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.6 (Ubuntu 20.04)
- AWS GPU Graviton avec AMI d'apprentissage profond TensorFlow 2.6 (Ubuntu 20.04)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.6 (Ubuntu 18.04)
- AWS GPU AMI TensorFlow 2.5 pour apprentissage profond (Amazon Linux 2)
- AWS GPU AMI d'apprentissage profond TensorFlow 2.5 (Ubuntu 20.04)

Cadre multiple

GPU

- AWS AMI d'apprentissage profond (Ubuntu 18.04)

Modifications importantes apportées au pilote NVIDIA DLAMIs

Le 15 novembre 2023, des modifications importantes AWS ont été apportées au AWS Apprentissage profond (deep learning) AMIs (DLAMI) concernant le pilote NIVIDA que j'utilise. DLAMIs Pour plus d'informations sur ce qui a changé et si cela a une incidence sur votre utilisation de DLAMIs, consultezModification du pilote DLAMI NVIDIA FAQs.

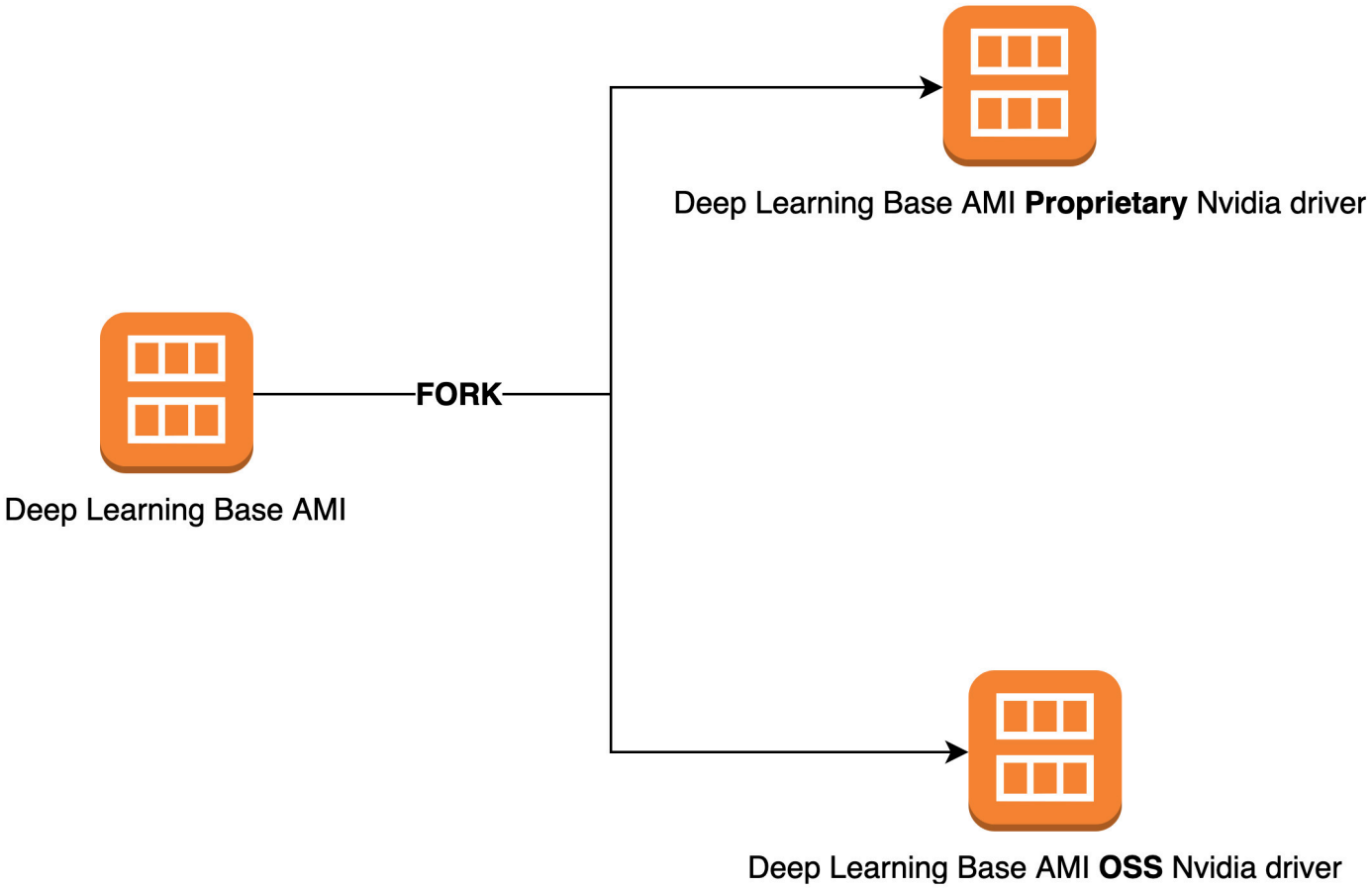
Modification du pilote DLAMI NVIDIA FAQs

Qu'est-ce qui a changé ?

Nous nous sommes DLAMIs divisés en deux groupes distincts :

- DLAMIs qui utilisent un pilote propriétaire NVIDIA (compatible avec P3, P3dn, G3)
- DLAMIs qui utilisent le pilote NVIDIA OSS (compatible avec G4dn, G5, P4, P5)

Par conséquent, nous en avons créé de nouvelles DLAMIs pour chacune des deux catégories avec de nouveaux noms et de nouvelles AMI IDs. Ils ne DLAMIs sont pas interchangeables. En d'autres termes, DLAMIs les instances prises en charge par l'autre groupe ne sont pas prises en charge par un groupe. Par exemple, le DLAMI qui prend en charge le P5 ne prend pas en charge le G3, et le DLAMI qui prend en charge le G3 ne prend pas en charge le P5.



Pourquoi ce changement a-t-il été nécessaire ?

Auparavant, DLAMIs NVIDIA GPUs incluait un pilote de noyau propriétaire de NVIDIA. Cependant, la communauté du noyau Linux en amont a accepté une modification qui empêche les pilotes de noyau propriétaires, tels que le pilote GPU NVIDIA, de communiquer avec d'autres pilotes de noyau. Cette modification désactive le GPUDirect RDMA sur les instances des séries P4 et P5, qui est le mécanisme qui permet d'utiliser efficacement EFA GPUs pour une formation distribuée. Par conséquent, utilisez DLAMIs désormais le pilote OpenRM (pilote open source NVIDIA), associé aux pilotes open source EFA pour prendre en charge les modèles G4dn, G5, P4 et P5. Cependant, ce pilote OpenRM ne prend pas en charge les anciennes instances (telles que P3 et G3). Par conséquent, pour garantir que nous continuons à fournir des solutions actuelles, performantes et sécurisées DLAMIs qui prennent en charge les deux types d'instances, nous nous sommes DLAMIs divisés en deux groupes : le premier avec le pilote OpenRM (qui prend en charge les modèles G4dn, G5, P4 et P5) et l'autre avec l'ancien pilote propriétaire (compatible avec les versions P3, P3dn et G3).

Qu' DLAMIs est-ce que ce changement a affecté ?

Ce changement a touché tout le monde DLAMIs.

Qu'est-ce que cela signifie pour toi ?

Tous DLAMIs continueront à fournir des fonctionnalités, des performances et une sécurité tant que vous les exécuterez sur un type d'instance Amazon Elastic Compute Cloud (Amazon EC2) compatible. Pour déterminer les types d' EC2 instances pris en charge par un DLAMI, consultez les notes de version de ce DLAMI, puis recherchez les instances prises en charge. EC2 Pour obtenir une liste des options DLAMI actuellement prises en charge et des liens vers leurs notes de version, voir. Notes de AMIs mise à jour du Deep Learning

De plus, vous devez utiliser les commandes correctes AWS Command Line Interface (AWS CLI) pour appeler le courant DLAMIs.

Pour les bases DLAMIs compatibles P3, P3dn et G3, utilisez cette commande :

```
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
```

Pour les bases DLAMIs compatibles avec G4dn, G5, P4 et P5, utilisez cette commande :

```
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
```

Y a-t-il une perte de fonctionnalité avec la version la plus récente DLAMIs ?

Non, il n'y a aucune perte de fonctionnalité. Les versions actuelles DLAMIs offrent toutes les fonctionnalités, les performances et la sécurité des versions précédentes DLAMIs, à condition que vous les exécutiez sur un type d' EC2 instance compatible.

Ce changement a-t-il affecté les Deep Learning Containers ?

Non, cette modification n'a pas affecté les AWS Deep Learning Containers, car ils n'incluent pas le pilote NVIDIA. Assurez-vous toutefois d'exécuter les Deep Learning Containers sur AMIs des instances compatibles avec les instances sous-jacentes.

Informations connexes sur DLAMI

Vous pouvez trouver d'autres ressources contenant des informations connexes sur le DLAMI en dehors du Guide du AWS Apprentissage profond (deep learning) AMIs développeur. Non AWS re:Post, consultez les questions d'autres clients sur le DLAMI ou posez vos propres questions. Sur le AWS Machine Learning Blog et sur d'autres AWS blogs, lisez les articles officiels sur le DLAMI.

AWS re:Post

Tag : AWS Apprentissage profond (deep learning) AMIs

AWS Blog

- AWS Blog sur le Machine Learning | Catégorie : AWS Apprentissage profond (deep learning) AMIs
- AWS Blog Machine Learning | Entraînement plus rapide grâce à la version TensorFlow 1.6 optimisée sur les instances Amazon EC2 C5 et P3
- AWS Blog sur le Machine Learning | Nouveau AWS Apprentissage profond (deep learning) AMIs pour les praticiens du Machine Learning
- AWS Partner Network Blog (APN) | Nouveaux cours de formation disponibles : introduction au Machine Learning et au Deep Learning sur AWS
- AWS Blog d'actualités | Plongez dans le Deep Learning avec AWS

Fonctionnalités obsolètes du DLAMI

Le tableau suivant répertorie les fonctionnalités obsolètes de (AWS Apprentissage profond (deep learning) AMIs DLAMI), la date à laquelle nous les avons rendues obsolètes et les raisons pour lesquelles nous les avons abandonnées.

Fonctionnalité	Date	Détails
Ubuntu 16.04	07/10/2021	Ubuntu Linux 16.04 LTS a atteint la fin de sa période LTS de cinq ans le 30 avril 2021 et n'est plus pris en charge par son fournisseur. Il n'y a plus de mises à jour de l'AMI Deep Learning Base (Ubuntu 16.04) dans les nouvelles versions depuis octobre 2021. Les versions précédentes continueront d'être disponibles.
Amazon Linux	07/10/2021	Amazon Linux date end-of-lifede décembre 2020. Il n'y a plus de mises à jour de l'AMI Deep Learning (Amazon Linux) dans les nouvelles versions depuis octobre 2021. Les versions précédentes de l'AMI Deep Learning (Amazon Linux) continueront d'être disponibles.
Chainer	01/07/2020	Chainer a annoncé la fin des principales versions à compter de décembre

Fonctionnalité	Date	Détails
		2019. Par conséquent, nous n'inclurons plus les environnements Chainer Conda dans le DLAMI à partir de juillet 2020. Les versions précédentes du DLAMI contenant ces environnements continuer ont d'être disponibles. Nous fournirons des mises à jour de ces environnements uniquement si des correctif s de sécurité sont publiés par la communauté open source pour ces infrastru ctures.
Python 3.6	15/06/2020	Suite aux demandes des clients, nous passons à Python 3.7 pour les nouvelles TF/MX/PT versions.

Fonctionnalité	Date	Détails
Python 2	01/01/2020	La communauté open source Python a officiell ement mis fin au support de Python 2. Les MXNet communautés TensorFlow PyTorch, et ont également annoncé que les versions TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4 et MXNet 1.6.0 seront les dernières à prendre en charge Python 2.

Historique du document pour DLAMI

Le tableau suivant fournit un historique des dernières versions du DLAMI et des modifications connexes apportées au AWS Apprentissage profond (deep learning) AMIs Guide du développeur.

Changements récents

Modification	Description	Date
Utiliser TensorFlow Serving pour entraîner un modèle MNIST	Exemple d'utilisation du service Tensorflow pour entraîner le modèle MNIST.	14 février 2025
ARM64 DLAMI	AWS Apprentissage profond (deep learning) AMIs Il prend désormais en charge les images basées sur le processeur GPUs Arm64.	29 novembre 2021
TensorFlow 2	L'AMI Deep Learning avec Conda est désormais disponibl e en TensorFlow 2 avec CUDA 10.	3 décembre 2019
AWS Inférentie	L'AMI Deep Learning prend désormais en charge le matériel AWS Inferentia et le SDK AWS Neuron.	3 décembre 2019
Installation PyTorch à partir d'une version nocturne	Un didacticiel a été ajouté qui explique comment désinstal ler PyTorch, puis installer une version nocturne de PyTorch votre AMI Deep Learning avec Conda.	25 septembre 2018

Tutoriel Conda	L'exemple MOTD a été mis à jour pour refléter une version plus récente.	23 juillet 2018

Changements antérieurs

Le tableau suivant fournit un historique des versions précédentes du DLAMI et des modifications associées avant juillet 2018.

Modification	Description	Date
TensorFlow avec Horovod	Ajout d'un tutoriel pour s'entraîner ImageNet avec Horovod TensorFlow et Horovod.	6 juin 2018
Guide de mise à niveau	Ajout du guide de mise à niveau.	15 mai 2018
Nouvelles régions et nouveau tutoriel de 10 minutes	Nouvelles régions ajoutées : USA Ouest (Californie du Nord), Amérique du Sud, Canada (Centre), UE (Londres) et UE (Paris). En outre, la première version d'un didacticiel de 10 minutes intitulé : « Mise en route avec l'AMI Deep Learning ».	26 avril 2018
Didacticiel Chainer	Un didacticiel a été ajouté pour l'utilisation de Chainer dans les modes à plusieurs GPU, à un seul GPU et à UC. L'intégration de CUDA a été mise à niveau de CUDA 8 à CUDA 9 pour plusieurs infrastructures.	28 février 2018

Modification	Description	Date
Linux AMIs v3.0, plus introduction de MXNet Model Server, TensorFlow Serving et TensorBoard	Ajout de didacticiels pour Conda AMIs avec de nouvelles fonctionnalités de service de modèles et de visualisation à l'aide de MXNet Model Server v0.1.5, TensorFlow Serving v1.4.0 et v0.4.0. TensorBoard Les capacités CUDA des AMI et de l'infrastructure sont décrites dans les présentations de Conda et de CUDA. Dernières notes de mise à jour déplacées vers https://aws.amazon.com/releasenotes/	25 janvier 2018
Linux AMIs v2.0	Base, Source et Conda ont été AMIs mis à jour avec NCCL 2.1. Source et Conda ont été AMIs mis à jour avec les MXNet versions 1.0, PyTorch 0.3.0 et Keras 2.0.9.	11 décembre 2017
Ajout de deux options d'AMI Windows	AMIs Sortie de Windows 2012 R2 et 2016 : ajout au guide de sélection des AMI et aux notes de version.	30 novembre 2017
Première édition de la documentation	Description détaillée de modification avec un lien vers les rubriques/sections ayant été modifiées.	15 novembre 2017

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.