aws

Guide du développeur

AWS Apprentissage profond (deep learning) AMIs

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que 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. Ils DLAMIs sont disponibles dans la plupart des Régions AWS pour différents types d'instances Amazon Elastic Compute Cloud (AmazonEC2), qu'il s'agisse d'une petite instance CPU uniquement ou GPU des instances multiples très puissantes les plus récentes. Ils DLAMIs sont préconfigurés avec NVIDIACUDAand NVIDIAcu DNN et 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 leDLAMIs. 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 la solution 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 AWS Inferentia et AWS Trainium et 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 leDLAMIs, nous vous recommandons de vous familiariser avec les outils de ligne de commande et les bases de Python.

Exemples de cas d'utilisation DLAMI

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

L'apprentissage en profondeur 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, DLAMI c'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 essayer un nouveau cadre, tester un nouveau modèle ou former de nouveaux modèles, alors DLAMI et AWS les capacités d'évolutivité peuvent atténuer les difficultés liées aux installations fastidieuses et à la gestion de plusieurs nœuds de formation.

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

Fonctionnalités d’DLAMI

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

Frameworks préinstallés

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

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

Le Deep Learning AMI 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. Le Deep Learning AMI avec Conda prend en charge les frameworks suivants :

- PyTorch
- TensorFlow 2

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

Logiciels préinstallés GPU

Même si vous utilisez une instance CPU uniquement, le DLAMIs will have NVIDIACUDAet le NVIDIAcu DNN. Le logiciel installé est le même, quel que soit le type d'instance. N'oubliez pas que les outils GPU spécifiques ne fonctionnent que sur une instance qui en possède au moins uneGPU. Pour plus d'informations sur les types d'instances, consultezChoix d'un type d'DLAMIinstance.

Pour plus d'informations surCUDA, voirInstallations de CUDA et liaisons d'infrastructures.

Service et visualisation de modèles

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

Commencer avec DLAMI

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

Si vous utilisez AWS ou utilisez Amazon pour la première foisEC2, commencez par leApprentissage profond AMI avec Conda. Si vous connaissez Amazon EC2 et d'autres AWS services tels qu'AmazonEMR, Amazon ou Amazon S3EFS, et que vous souhaitez intégrer ces services à des projets nécessitant une formation ou des inférences distribuées, 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 toute une gamme d'DLAMIoptions. Pour vous aider à sélectionner celle qui DLAMI convient 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 :

- DLAMIType : CUDA contre base contre framework unique contre framework multiple (Conda) DLAMI
- Architecture de calcul : Graviton basé sur x86 contre Graviton basé sur ARM64 AWS
- Type de processeur : GPUhttps://docs.aws.amazon.com/dlami/latest/devguide/gpu contre Inferentia CPUcontre Trainium
- SDK: CUDAcontre AWS Neuron
- Système d'exploitation : Amazon Linux contre Ubuntu

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

Rubriques

Suivant

Apprentissage profond AMI 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 des Deep Learning séries de versions CUDA sont disponibles

Toutes les Apprentissage profond AMI 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 mise à jour pour DLAMIs

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

Choisissez l'une des versions de CUDA et consultez la liste complète des DLAMI dotés de cette version dans l'annexe, ou apprenez-en plus sur les différents DLAMI avec l'option Next Up.

Suivant

AMI des Deep Learning

Rubriques connexes

- Pour plus d'informations sur le basculement entre les versions CUDA, consultez le didacticiel Utilisation de la base d'apprentissage profond AMI.

AMI des 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'à l'installation d'un framework particulier, et vous pouvez choisir entre les versions CUDA.

Pourquoi choisir la base DLAMI

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 apprenez-en plus sur les différents DLAMI à l'aide de l'option Next Up.

Suivant

DLAMI avec Conda

Rubriques connexes

- Utilisation de l'AMI Deep Learning Base

Apprentissage profond AMI avec Conda

Le Conda DLAMI utilise des environnements `conda` 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 idéal pour apprendre et expérimenter avec tous les cadres qu'DLAMIil a à offrir. La plupart des utilisateurs trouvent que le nouveau Deep Learning AMI avec Conda est parfait pour eux.

Ils sont souvent mis à jour avec les dernières versions des frameworks et disposent des derniers GPU pilotes et logiciels. Ils sont généralement désignés comme tels AWS Apprentissage profond (deep learning) AMIs dans la plupart des documents. Ils sont DLAMIs compatibles avec les systèmes d'exploitation Ubuntu 20.04 et Amazon Linux 2. 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

Les Conda AMIs utilisent 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 celles d'Intel MKLDNN, qui accélèrent l'entraînement et l'inférence sur les types d'instances C5 et C4. CPU Les fichiers binaires sont également compilés pour prendre en charge les ensembles d'instructions Intel avancés, y compris, mais sans s'y limiterAVX, AVX -2, SSE4 .1 et SSE4 .2. Ils accélèrent les opérations vectorielles et en virgule flottante sur CPU les architectures Intel. De plus, pour les types d'GPUinstance, the CUDA et cu, DNN sont mis à jour avec la version prise en charge par la dernière version officielle.

Le Deep Learning AMI with 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 Utiliser le Deep Learning AMI avec Conda.

Si vous souhaitez effectuer une installation à partir des sources, en utilisant des options de construction personnalisées ou optimisées, le AMI des 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 DLAMI (v26, v25, etc.) contenant les environnements Python 2 Conda sont toujours disponibles. Cependant, nous fournissons des mises à jour des environnements Python 2 Conda sur les DLAMI versions publiées précédemment uniquement si des correctifs de sécurité ont été publiés par la communauté open source pour ces versions. DLAMIles versions contenant les dernières versions des PyTorch frameworks TensorFlow et ne contiennent pas les environnements Python 2 Conda.

CUDASupport

Les numéros de CUDA version spécifiques se trouvent dans les notes GPU DLAMI de publication.

Suivant

Options d'architecture DLAMI

Rubriques connexes

- Pour un didacticiel sur l'utilisation d'un Deep Learning AMI avec Conda, consultez le Utiliser le Deep Learning AMI 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 DLAMI du GPU ARM64, consultez. Le ARM64 DLAMI Pour plus de détails sur les types d'instances disponibles, consultezChoix d'un type d'DLAMIinstance.

Suivant

Options du système d'exploitation DLAMI

Options du système d'exploitation DLAMI

Les DLAMI sont proposés dans les systèmes d'exploitation suivants.

- Amazon Linux 2

- Ubuntu 20.04
- Ubuntu 22.04

Les anciennes versions des systèmes d'exploitation sont disponibles sur les DLAMI obsolètes. 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'DLAMIinstance

Choix d'un type d'DLAMIinstance

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

- Si vous débutez dans le domaine de l'apprentissage profond, une instance dotée d'une instance unique GPU peut répondre à vos besoins.
- Si vous êtes soucieux de votre budget, vous pouvez utiliser CPU uniquement des instances.
- 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 GPU instance hautes performances avec une CPU architecture 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 avec une fraction de. GPU
- Pour les services d'inférence à volume élevé, une CPU instance unique avec beaucoup de mémoire, ou un cluster d'instances de ce type, 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 deGPUs. 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 de machine learning à l'aide de NVIDIA Collective Communications Library (NCCL) 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 NVIDIA pilotes, logiciels ou boîtes à outils uniquement sur les EC2 instances Amazon qui incluent NVIDIA du matériel.

Rubriques

Tarification des 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 DLAMI soit gratuit, vous devez tout de même payer pour le matériel d'EC2instance Amazon sous-jacent.

Certains types d'EC2instances Amazon sont étiquetés comme gratuits. Il est possible de l'exécuter DLAMI sur l'une de ces instances gratuites. Cela signifie que l'utilisation de DLAMI est entièrement gratuite lorsque vous utilisez uniquement la capacité de cette instance. Si vous avez besoin d'une instance plus puissante avec plus de CPU cœurs, plus d'espace disque, plusRAM, ou un ou

plusieursGPUs, 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 EC2les tarifs Amazon.

DLAMI : disponibilité dans les régions

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. DLAMIsne sont pas disponibles dans toutes les régions, mais il est possible de les DLAMIs copier dans la région de votre choix. Voir Copier un 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'en 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

GPUInstances recommandées

GPUInstances recommandées

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

Les types d'instances suivants prennent en charge leDLAMI. Pour plus d'informations sur les options de type d'GPUinstance 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 le nombre d'instances disponiblesRAM, choisissez un autre type d'instance avec suffisamment de mémoire pour votre application.

- 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 possèdent jusqu'à 4 NVIDIA GPUs T4.
- Les instances Amazon EC2 G5 comportent jusqu'à 8 NVIDIA GPUs A10G.
- Les instances Amazon EC2 G6 ont jusqu'à 8 NVIDIA GPUs L4.
- Les instances Amazon EC2 G6e possèdent jusqu'à 8 cœurs NVIDIA tensoriels L40S. GPUs
- Les instances Amazon EC2 G5g sont équipées de processeurs Graviton2 basés sur ARM64 AWS .

DLAMIles instances fournissent des outils pour surveiller et optimiser vos GPU processus. Pour plus d'informations sur la surveillance de vos GPU processus, consultezGPUSurveillance et optimisation.

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

Suivant

CPUInstances recommandées

CPUInstances recommandées

Que vous ayez un budget limité, que vous vous initiiez au deep learning ou que vous souhaitiez simplement gérer un service de prédiction, vous avez de nombreuses options abordables dans CPU cette catégorie. Certains frameworks tirent parti de ceux d'Intel MKLDNN, qui accélèrent la formation et l'inférence sur les types d'CPUinstances C5 (non disponibles dans toutes les régions). Pour plus d'informations sur les types d'CPUinstances, consultez d'EC2instances 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 le nombre d'instances disponiblesRAM, choisissez un autre type d'instance avec suffisamment de mémoire pour votre application.

- Les instances Amazon EC2 C5 possèdent jusqu'à 72 processeurs IntelvCPUs. 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 AWS Neuron SDK, 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 le nombre d'instances disponiblesRAM, 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 InferentiaDLAMIs, 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 AWS Neuron SDK, 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 le nombre d'instances disponiblesRAM, 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 DLAMI instance

Après avoir sélectionné DLAMI et choisi le type d'instance Amazon Elastic Compute Cloud (AmazonEC2) que vous souhaitez utiliser, vous êtes prêt à configurer votre nouvelle DLAMI instance.

Si vous n'avez pas encore choisi DLAMI de type d'EC2instance, consultezCommencer avec DLAMI.

Rubriques

- Trouver l'identifiant d'un DLAMI
- Lancement d'une DLAMI instance
- Connexion à une DLAMI instance
- Configuration d'un serveur Jupyter Notebook sur une instance DLAMI
- Nettoyage d'une DLAMI instance

Trouver l'identifiant d'un DLAMI

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

Vous pouvez trouver l'identifiant DLAMI de votre choix en utilisant une AWS CLI commande pour Amazon EC2 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 DLAMI identifiant en utilisant 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, *architecture* peut être l'un **x86_64** ou l'autre**arm64**. Spécifiez le *ami_type* en prenant le DLAMI nom et en supprimant les mots clés « deep », « learning » et « ami ». AMILe nom se trouve dansNotes de mise à jour pour DLAMIs.

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

- ```
  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 DLAMI frameworks actuellement pris en charge, vous trouverez des exemples de ssm get-parameter commandes plus spécifiques dansNotes de mise à jour pour DLAMIs. Choisissez le lien vers les notes de publication de votre choixDLAMI, puis recherchez leur identifiant dans les notes de publication.

Using Amazon EC2 CLI

Pour trouver un DLAMI identifiant en utilisant ec2 describe-images

Dans la ec2 describe-imagescommande suivante, entrez le DLAMI nom pour la valeur du filtre`Name=name`. 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 à la commande DLAMI de votre choix, consultezNotes de mise à jour pour DLAMIs. Choisissez le lien vers les notes de publication de votre choixDLAMI, puis recherchez leur identifiant dans les notes de publication.

Étape suivante

Lancement d'une DLAMI instance

Lancement d'une DLAMI instance

Une fois que vous avez trouvé l'ID DLAMI que vous souhaitez utiliser pour lancer une DLAMI instance, 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 procédure pas à pas, nous pouvons faire des références spécifiques au pilote OSS Nvidia Deep Learning Base GPU AMI (Ubuntu 22.04). Même si vous en sélectionnez un autreDLAMI, 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 DLAMI instance avec un adaptateur Elastic Fabric (EFA). Pour des instructions spécifiques, voirLancement d'une AWS Apprentissage profond (deep learning) AMIs instance avec EFA.

1. Ouvrez la EC2console.
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 le nom de votre instance et sélectionnez DLAMI celui qui vous convient.

   a. Trouvez-en un existant DLAMI dans Mon AMIs ou choisissez Démarrage rapide.

   b. Recherche par DLAMI identifiant. Parcourez les options, puis sélectionnez votre choix.
5. Choisissez un type d'instance. Vous trouverez les familles d'instances recommandées pour votre DLAMI inNotes de mise à jour pour DLAMIs. Pour des recommandations générales sur les types d'DLAMIinstances, consultezChoix d'un type d'DLAMIinstance.
6. Choisissez Launch Instances (Lancer les instances).

AWS CLI

- Pour utiliser le AWS CLI, vous devez disposer de l'ID de l'instance DLAMI que vous souhaitez utiliser, du type d'EC2instance Région AWS et 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, notamment 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 EC2console 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 DLAMI instance

Connexion à une DLAMI instance

Une fois que vous avez lancé une DLAMI instance et que celle-ci est en cours d'exécution, vous pouvez vous y connecter depuis un client (Windows, macOS ou Linux) à l'aide deSSH. 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 SSH connexion 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 DLAMI instance.
- 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

Pour garantir la sécurité de votre serveur Jupyter Notebook, nous vous recommandons de définir un mot de passe et de créer un SSL certificat pour le serveur. Pour configurer un mot de passeSSL, connectez-vous d'abord à votre DLAMI instance, 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. Créez un SSL 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 SSL certificat standard signé par un tiers et n'entraînant aucun avertissement de sécurité du 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 SSL, vous pouvez démarrer le serveur. Connectez-vous à votre DLAMI instance et exécutez la commande suivante qui utilise le SSL certificat que vous avez créé précédemment.

```
$ jupyter notebook --certfile=~/ssl/mycert.pem --keyfile ~/ssl/mykey.key
```

Une fois le serveur démarré, vous pouvez désormais vous y connecter via un SSH tunnel depuis votre ordinateur client. Lorsque le serveur fonctionne, un message de Jupyter vous le confirme. À ce stade, ignorez l'avertissement selon lequel vous pouvez accéder au serveur via un hôte localURL, 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 DLAMI instance, 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 SSH tunnel :

- Le DNS nom 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 DLAMI instance depuis un client Windows, macOS ou Linux, suivez les instructions du système d'exploitation de votre client.

Windows

Pour vous connecter à votre DLAMI instance depuis un client Windows à l'aide de SSH

1. Utilisez un SSH client pour Windows, tel que PuTTY. Pour obtenir des instructions, consultez Connect to your Linux instance using Pu TTY dans le guide de EC2 l'utilisateur Amazon. Pour d'autres options de SSH connexion, consultez Se connecter à votre instance Linux à l'aide deSSH.
2. (Facultatif) Créez un SSH tunnel 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 DLAMI instance depuis 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'EC2instance Amazon et le DNS nom public de votre EC2 instance Amazon. Notez que pour un Amazon LinuxAMI, 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'EC2instance Amazon distante qui exécute le serveur Jupyter Notebook.

Étape suivante

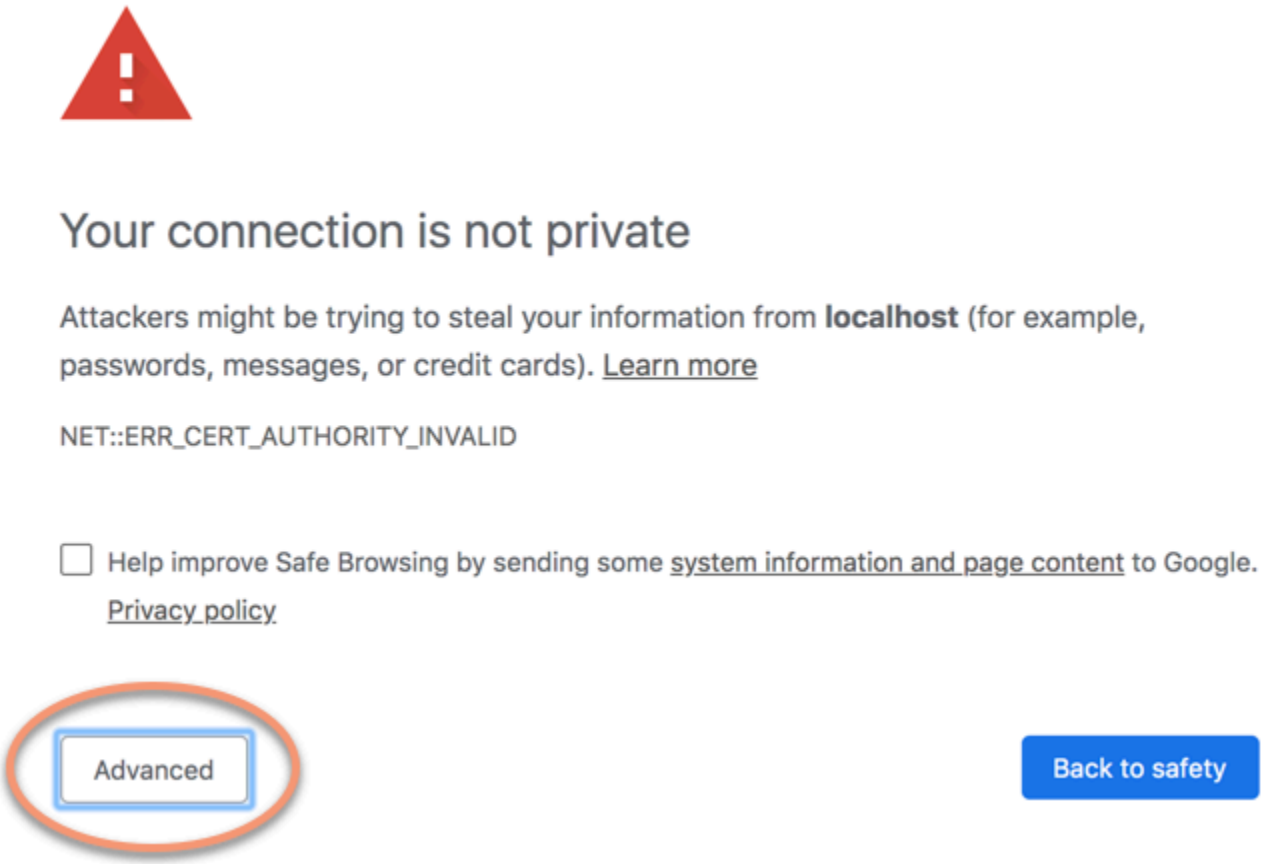
Connexion au serveur Jupyter Notebook sur une instance DLAMI

Connexion au serveur Jupyter Notebook sur une instance DLAMI

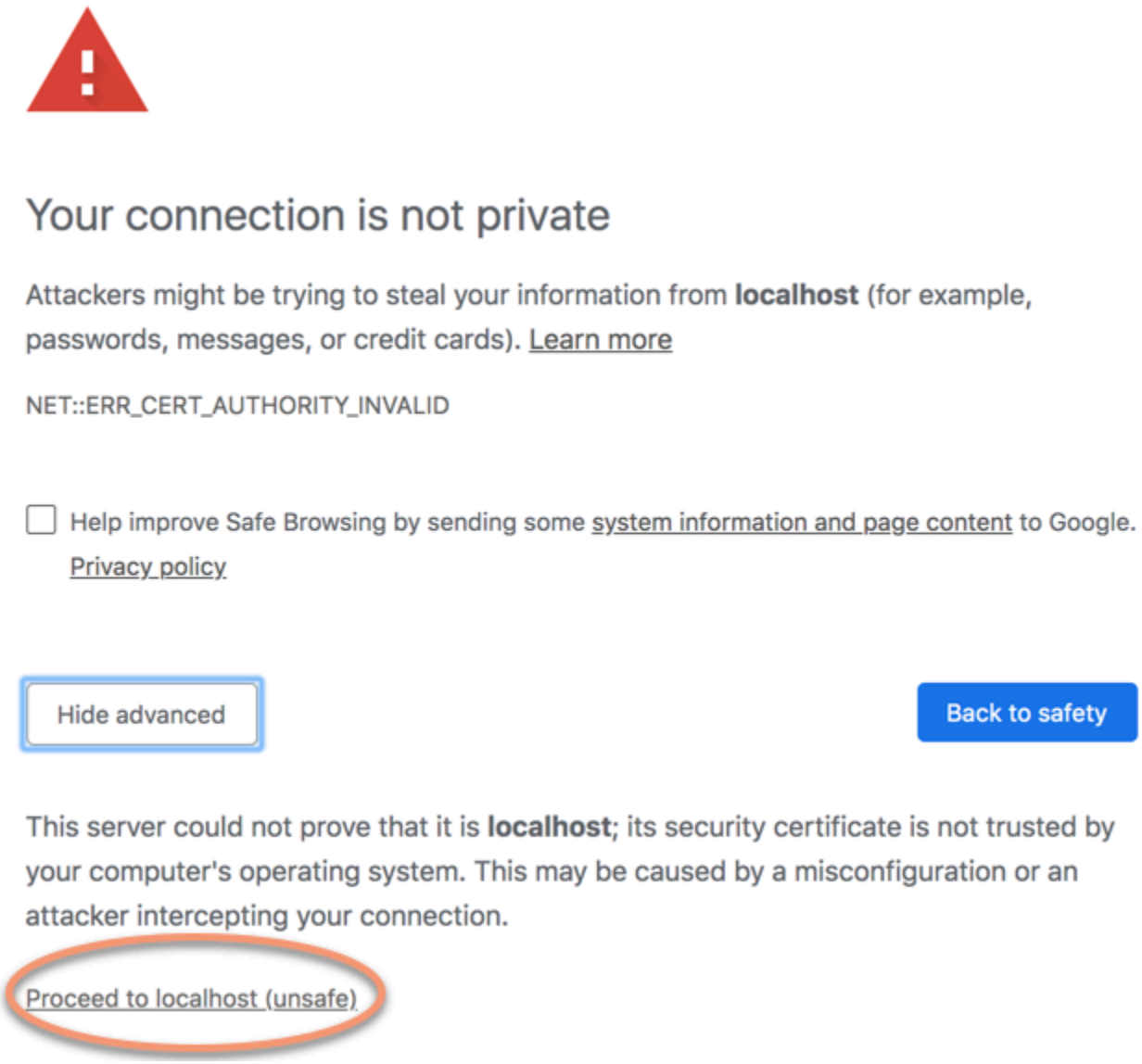
Après avoir connecté votre client au serveur Jupyter Notebook sur votre DLAMI instance, vous pouvez vous connecter au serveur.

Pour vous connecter au serveur dans votre navigateur

1. Dans la barre d'adresse de votre navigateur, saisissez ce qui suit URL ou cliquez sur ce lien : https://localhost:8888
2. Avec un SSL certificat auto-signé, 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'DLAMIinstance. Vous pouvez créer de nouveaux blocs-notes ou exécuter les Didacticiels.

Nettoyage d'une DLAMI instance

Lorsque vous n'avez plus besoin de votre DLAMI instance, vous pouvez l'arrêter ou la résilier sur Amazon EC2 afin d'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'EC2instances 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.

À l'aide d'un DLAMI

Rubriques

Les sections suivantes décrivent comment le Deep Learning AMI avec Conda peut être utilisé 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.

Utiliser le Deep Learning AMI avec Conda

Rubriques

Présentation du Deep Learning AMI 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.

Le Deep Learning AMI with Conda a été configuré 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 les DLAMI exemples de frameworks trouvés sur le site de projet de chaque framework.

Connectez-vous à votre DLAMI

Une fois connecté à votre serveur, vous verrez un « message du jour » du serveur (MOTD) décrivant les différentes commandes Conda que vous pouvez utiliser pour passer d'un framework d'apprentissage profond à un autre. Vous trouverez ci-dessous un exempleMOTD. Vos spécificités MOTD peuvent varier à mesure que de nouvelles versions 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. Le Deep Learning AMI with 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 iPython terminal.

```
(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 iPython console, utilisez-la`quit()`, 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 iPython terminal.

```
(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 leDLAMI, vous pouvez choisir de désinstaller les packages Conda que vous n'utilisez pas :

```
conda env list
conda env remove --name <env_name>
```

Utilisation de la base d'apprentissage profond AMI

Utilisation de la base d'apprentissage profond AMI

The Base AMI est fourni avec une plate-forme de base de GPU pilotes et de bibliothèques d'accélération pour déployer votre propre environnement d'apprentissage profond personnalisé. Par défaut, AMI il est configuré avec n'importe quel environnement de NVIDIA CUDA version. Vous pouvez également passer d'une version à l'autre deCUDA. Consultez les instructions suivantes pour savoir comment procéder.

Configuration des CUDA versions

Vous pouvez vérifier la CUDA version en exécutant NVIDIA le `nvcc` programme.

```
nvcc --version
```

Vous pouvez sélectionner et vérifier une CUDA version particulière 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 DLAMI mise à jour 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 DLAMI laquelle. Si vous avez choisi l'Apprentissage profond AMI avec Conda, vous bénéficiez de quelques didacticiels déjà configurés et prêts à tester.

> **Important**
>
> Pour exécuter les didacticiels Jupyter Notebook installés sur leDLAMI, 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 utilisez le Deep Learning AMI 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 du Deep Learning AMI avec Conda.

> **Note**
>
> De nombreux didacticiels nécessitent des modules Python supplémentaires qui ne sont peut-être pas configurés sur votre ordinateurDLAMI. Si un message d'erreur s'affiche"`xyz`

`module not found`", par exemple, connectez-vous auDLAMI, 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 euxGPUs. Si votre type d'instance n'en possède pasGPU, vous devrez peut-être modifier une partie du code de l'exemple pour le faire fonctionner.

Navigation dans les didacticiels installés

Une fois que vous êtes connecté au serveur Jupyter et que vous pouvez voir le répertoire des tutoriels (uniquement pour le Deep Learning AMI avec Conda), des dossiers de didacticiels portant le nom de chaque framework vous seront présentés. Si aucun framework n'est répertorié, cela signifie que les didacticiels ne sont pas disponibles pour ce framework dans votre version actuelleDLAMI. 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 lancerez un bloc-notes sur le Deep Learning AMI 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 MOTD lorsque vous vous connectez pour la première fois auDLAMI.

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 du Deep Learning AMI avec le logiciel Conda.

Rubriques

- Activation des infrastructures
- Entraînement distribué à l'aide d'Elastic Fabric Adapter
- GPUSurveillance et optimisation
- La puce AWS Inferentia avec DLAMI
- Le ARM64 DLAMI
- Inférence
- Service de modèle

Activation des infrastructures

Voici les frameworks d'apprentissage profond installés sur le Deep Learning AMI avec Conda. 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 Deep Learning AMI with Conda.

Pour PyTorch Python 3 avec CUDA et MKL -DNN, exécutez cette commande :

```
$ source activate pytorch_p310
```

Démarrez le iPython terminal.

```
(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 Deep Learning AMI with 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 GPU instances) - Installez la dernière version nocturne de PyTorch with CUDA .0 :

```
(pytorch_p310)$ pip install torch_nightly -f https://download.pytorch.org/whl/
nightly/cu100/torch_nightly.html
```

   • (Option pour les CPU instances) - 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'MNISTexemple, 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 montre comment activer TensorFlow 2 sur une instance exécutant le Deep Learning AMI avec Conda (DLAMIsur Conda) et exécuter un programme TensorFlow 2.

Lorsqu'un package Conda stable d'un framework est publié, il est testé et préinstallé sur le. DLAMI

Activation de TensorFlow 2

Pour courir TensorFlow DLAMI avec Conda

1. Pour en activer TensorFlow 2, ouvrez une instance Amazon Elastic Compute Cloud (AmazonEC2) du DLAMI with 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 iPython terminal :

   ```
   (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 du TensorFlow Python API 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 DLAMI instance pour accélérer les applications de calcul haute performance (HPC). EFAvous permet d'atteindre les performances applicatives d'un HPC cluster sur site, grâce à l'évolutivité, à la flexibilité et à l'élasticité offertes par le AWS cloud.

Les rubriques suivantes vous montrent comment commencer EFA à utiliser leDLAMI.

> **Note**
>
> Choisissez votre DLAMI dans cette GPUDLAMIliste de base

Rubriques

- Lancement d'une AWS Apprentissage profond (deep learning) AMIs instance avec EFA
- En utilisant EFA sur le DLAMI

Lancement d'une AWS Apprentissage profond (deep learning) AMIs instance avec EFA

La dernière version de Base DLAMI est prête à l'emploi EFA et est livrée avec les pilotes, les modules de noyau, libfabric, openmpi et le NCCLOFIplugin requis pour les instances. GPU

Vous trouverez les CUDA versions prises en charge d'une base DLAMI dans les notes de publication.

Remarque :

- Lorsque vous exécutez une NCCL application à l'aide de `mpirun` onEFA, vous devez spécifier le chemin complet vers l'installation EFA prise en charge comme suit :

```
/opt/amazon/openmpi/bin/mpirun <command>
```

- Pour activer l'utilisation de votre applicationEFA, ajoutez `FI_PROVIDER="efa"` à la `mpirun` commande comme indiqué dansEn utilisant EFA sur le DLAMI.

Rubriques

- Préparer un groupe de sécurité activé pour EFA
- Lancer votre instance
- Vérifier la pièce jointe EFA

Préparer un groupe de sécurité activé pour EFA

EFAnécessite 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 EFAdocumentation.

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 l'instance VPC dans laquelle vous souhaitez lancer vos instances EFA activées.
   - Sélectionnez Create (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 Save (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 DLAMI instance.

Lancer votre instance

EFAsur le AWS Apprentissage profond (deep learning) AMIs est actuellement pris en charge avec les types d'instances et systèmes d'exploitation suivants :

- P3DN.24xlarge : Amazon Linux 2, Ubuntu 20.04
- P4D.24xLarge : Amazon Linux 2, Ubuntu 20.04
- P5.48xlarge : Amazon Linux 2, Ubuntu 20.04

La section suivante explique comment lancer une DLAMI instance EFA activée. Pour plus d'informations sur le lancement d'une instance EFA activée, voir Lancer des instances EFA activées 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 AMI page Choisir un produit, sélectionnez un support DLAMI trouvé sur la page des notes DLAMI de version
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. Reportez-vous à ce lien pour obtenir la liste des instances prises en charge : Get started with EFA and MPI
5. Sur la page Configurer les détails de l’instance, procédez de la façon suivante :

   - Dans le champ Nombre d'instances, entrez le nombre d'instances EFA activées que vous souhaitez lancer.
   - Pour Réseau et sous-réseau, sélectionnez le sous-réseau VPC et dans lequel vous souhaitez 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 associé à un IPv6 CIDR bloc, vous pouvez éventuellement spécifier une IPv6 adresse principale et une ou plusieurs IPv6 adresses secondaires.
- Choisissez Next: Add Storage (Suivant : Ajouter le stockage).

6. Sur la page Ajouter du stockage, spécifiez les volumes à associer aux instances en plus des volumes spécifiés par le AMI (comme le volume du périphérique racine), puis choisissez Next : Add Tags.
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.

Vérifier 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 ce n'est pas activé, vous pouvez résoudre ce problème soit en :

- Mettre fin à l'EC2instance et en lancer une nouvelle en suivant les mêmes étapes. Vérifiez que EFA est attaché.
- Attacher 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 de l'attacherEFA.
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 leDLAMI. 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 leDLAMI. 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 règles entrantes/sortantes correctes.

En utilisant EFA sur le DLAMI

La section suivante décrit comment exécuter des applications EFA à nœuds multiples sur le AWS Apprentissage profond (deep learning) AMIs.

Exécuter des applications multi-nœuds avec EFA

Pour exécuter une application sur un cluster de nœuds, la configuration suivante est requise

Rubriques

- Activer le mode sans mot de passe SSH
- Créer un fichier hosts.
- NCCLTests

Activer le mode sans mot de passe SSH

Sélectionnez un nœud de votre cluster comme nœud principal. Les autres nœuds sont appelés nœuds de membre.

1. Sur le nœud leader, générez la paire de RSA clés.

```
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
```

NCCLTests

> **Note**
>
> Ces tests ont été exécutés avec la EFA version 1.30.0 et le OFI NCCL plugin 1.7.4.

Vous trouverez ci-dessous un sous-ensemble de NCCL tests 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

Tests de fonctionnalité

NCCLTest de transfert de messages sur plusieurs nœuds

Le nccl_message_transfer est un test simple pour s'assurer que le NCCL OFI plugin fonctionne comme prévu. Le test valide le fonctionnement de l'établissement NCCL de la connexion et du transfert APIs de données. Assurez-vous d'utiliser le chemin complet vers mpirun comme indiqué dans l'exemple lorsque vous exécutez NCCL des applications avec. EFA Modifiez les paramètres np en N fonction du nombre d'instances présentes GPUs dans votre cluster. Pour plus d'informations, consultez la AWS OFINCCLdocumentation.

Le test nccl_message_transfer suivant concerne une version xx.x générique. CUDA Vous pouvez exécuter les commandes pour n'importe quelle CUDA version disponible dans votre EC2 instance Amazon en remplaçant la CUDA version dans le script.

```
$/opt/amazon/openmpi/bin/mpirun -n 2 -N 1 --hostfile hosts \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:$LD_LIBRARY_PATH \
--mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to none \
opt/aws-ofi-nccl/tests/nccl_message_transfer
```

Le résultat doit être similaire à ce qui suit. Vous pouvez vérifier la sortie pour voir qu'elle EFA est utilisée en tant que OFI fournisseur.

```
INFO: Function: nccl_net_ofi_init Line: 1069: NET/OFI Selected Provider is efa (found 4
 nics)
```

```
INFO: Function: nccl_net_ofi_init Line: 1160: NET/OFI Using transport protocol SENDRECV
INFO: Function: configure_ep_inorder Line: 261: NET/OFI Setting
 FI_OPT_EFA_SENDRECV_IN_ORDER_ALIGNED_128_BYTES not supported.
INFO: Function: configure_nccl_proto Line: 227: NET/OFI Setting NCCL_PROTO to "simple"
INFO: Function: main Line: 86: NET/OFI Process rank 1 started. NCCLNet device used on
 ip-172-31-13-179 is AWS Libfabric.
INFO: Function: main Line: 91: NET/OFI Received 4 network devices
INFO: Function: main Line: 111: NET/OFI Network supports communication using CUDA
 buffers. Dev: 3
INFO: Function: main Line: 118: NET/OFI Server: Listening on dev 3
INFO: Function: main Line: 131: NET/OFI Send connection request to rank 1
INFO: Function: main Line: 173: NET/OFI Send connection request to rank 0
INFO: Function: main Line: 137: NET/OFI Server: Start accepting requests
INFO: Function: main Line: 141: NET/OFI Successfully accepted connection from rank 1
INFO: Function: main Line: 145: NET/OFI Send 8 requests to rank 1
INFO: Function: main Line: 179: NET/OFI Server: Start accepting requests
INFO: Function: main Line: 183: NET/OFI Successfully accepted connection from rank 0
INFO: Function: main Line: 187: NET/OFI Rank 1 posting 8 receive buffers
INFO: Function: main Line: 161: NET/OFI Successfully sent 8 requests to rank 1
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 0
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 1
```

Tests de performance

Test de NCCL performance multi-nœuds sur P4D.24xlarge

Pour vérifier les NCCL performancesEFA, exécutez le test de NCCL performance standard disponible sur le référentiel officiel NCCL -Tests. Il DLAMI est livré 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'NCCLapplications avec. EFA
- Modifiez les paramètres np et N en fonction du nombre d'instances et GPUs de votre cluster.
- Ajoutez l'INFOindicateur NCCL _ DEBUG = et assurez-vous que les journaux indiquent l'EFAutilisation comme « 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 l'un des nœuds membres pour surveiller GPU l'utilisation. Les `watch nvidia-smi` commandes suivantes concernent une version CUDA xx.x générique et dépendent du système d'exploitation de votre instance. Vous pouvez exécuter les commandes pour n'importe quelle CUDA version disponible dans votre EC2 instance Amazon en remplaçant la CUDA version dans le script.

- Amazon Linux 2 :

```
 $ /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 :

```
$ /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   9591 on ip-172-31-4-37 device  0 [0x10] NVIDIA A100-SXM4-40GB
#  Rank  1 Group  0 Pid   9592 on ip-172-31-4-37 device  1 [0x10] NVIDIA A100-SXM4-40GB
#  Rank  2 Group  0 Pid   9593 on ip-172-31-4-37 device  2 [0x20] NVIDIA A100-SXM4-40GB
#  Rank  3 Group  0 Pid   9594 on ip-172-31-4-37 device  3 [0x20] NVIDIA A100-SXM4-40GB
#  Rank  4 Group  0 Pid   9595 on ip-172-31-4-37 device  4 [0x90] NVIDIA A100-SXM4-40GB
```

```
#  Rank  5 Group  0 Pid   9596 on ip-172-31-4-37 device  5 [0x90] NVIDIA A100-SXM4-40GB
#  Rank  6 Group  0 Pid   9597 on ip-172-31-4-37 device  6 [0xa0] NVIDIA A100-SXM4-40GB
#  Rank  7 Group  0 Pid   9598 on ip-172-31-4-37 device  7 [0xa0] NVIDIA A100-SXM4-40GB
#  Rank  8 Group  0 Pid  10216 on ip-172-31-13-179 device  0 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank  9 Group  0 Pid  10217 on ip-172-31-13-179 device  1 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank 10 Group  0 Pid  10218 on ip-172-31-13-179 device  2 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank 11 Group  0 Pid  10219 on ip-172-31-13-179 device  3 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank 12 Group  0 Pid  10220 on ip-172-31-13-179 device  4 [0x90] NVIDIA A100-
SXM4-40GB
#  Rank 13 Group  0 Pid  10221 on ip-172-31-13-179 device  5 [0x90] NVIDIA A100-
SXM4-40GB
#  Rank 14 Group  0 Pid  10222 on ip-172-31-13-179 device  6 [0xa0] NVIDIA A100-
SXM4-40GB
#  Rank 15 Group  0 Pid  10223 on ip-172-31-13-179 device  7 [0xa0] NVIDIA A100-
SXM4-40GB
ip-172-31-4-37:9591:9591 [0] NCCL INFO Bootstrap : Using ens32:172.31.4.37
ip-172-31-4-37:9591:9591 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin_v6
 symbol.
ip-172-31-4-37:9591:9591 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin
 symbol (v4 or v5).
ip-172-31-4-37:9591:9591 [0] NCCL INFO cudaDriverVersion 12020
NCCL version 2.18.5+cuda12.2
...
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.7.4-aws
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Using CUDA runtime version 11070
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Using CUDA runtime version 11070
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
 variable to 1
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Disabling NVLS support due to NCCL
 version 21602
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
 variable to 1
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Disabling NVLS support due to NCCL
 version 21602
```

```
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Running on p4d.24xlarge platform,
 Setting NCCL_TOPO_FILE environment variable to /opt/aws-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)
           0             0     float     sum      -1    11.02    0.00    0.00      0
 11.04    0.00    0.00      0
           0             0     float     sum      -1    11.01    0.00    0.00      0
 11.00    0.00    0.00      0
           0             0     float     sum      -1    11.02    0.00    0.00      0
 11.02    0.00    0.00      0
           0             0     float     sum      -1    11.01    0.00    0.00      0
 11.00    0.00    0.00      0
           0             0     float     sum      -1    11.02    0.00    0.00      0
 11.02    0.00    0.00      0
         256             4     float     sum      -1    632.7    0.00    0.00      0
 628.2    0.00    0.00      0
         512             8     float     sum      -1    627.4    0.00    0.00      0
 629.6    0.00    0.00      0
        1024            16     float     sum      -1    632.2    0.00    0.00      0
 631.7    0.00    0.00      0
        2048            32     float     sum      -1    631.0    0.00    0.00      0
 634.2    0.00    0.00      0
        4096            64     float     sum      -1    623.3    0.01    0.01      0
 633.6    0.01    0.01      0
        8192           128     float     sum      -1    635.1    0.01    0.01      0
 633.5    0.01    0.01      0
       16384           256     float     sum      -1    634.8    0.03    0.02      0
 637.0    0.03    0.02      0
       32768           512     float     sum      -1    647.9    0.05    0.05      0
 636.8    0.05    0.05      0
       65536          1024     float     sum      -1    658.9    0.10    0.09      0
 667.0    0.10    0.09      0
      131072          2048     float     sum      -1    671.9    0.20    0.18      0
 662.9    0.20    0.19      0
      262144          4096     float     sum      -1    692.1    0.38    0.36      0
 685.1    0.38    0.36      0
```

```
      524288          8192     float     sum      -1    715.3    0.73    0.69      0
 696.6    0.75    0.71      0
     1048576         16384     float     sum      -1    734.6    1.43    1.34      0
 729.2    1.44    1.35      0
     2097152         32768     float     sum      -1    785.9    2.67    2.50      0
 794.5    2.64    2.47      0
     4194304         65536     float     sum      -1    837.2    5.01    4.70      0
 837.6    5.01    4.69      0
     8388608        131072     float     sum      -1    929.2    9.03    8.46      0
 931.4    9.01    8.44      0
    16777216        262144     float     sum      -1   1773.6    9.46    8.87      0
 1772.8    9.46    8.87      0
    33554432        524288     float     sum      -1   2110.2   15.90   14.91      0
 2116.1   15.86   14.87      0
    67108864       1048576     float     sum      -1   2650.9   25.32   23.73      0
 2658.1   25.25   23.67      0
   134217728       2097152     float     sum      -1   3943.1   34.04   31.91      0
 3945.9   34.01   31.89      0
   268435456       4194304     float     sum      -1   7216.5   37.20   34.87      0
 7178.6   37.39   35.06      0
   536870912       8388608     float     sum      -1    13680   39.24   36.79      0
 13676   39.26   36.80      0
[  1073741824      16777216     float     sum      -1    25645   41.87   39.25      0
 25497   42.11   39.48      0 ] <- Used For Benchmark
...
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 7.46044
```

Tests de validation

Pour vérifier que les EFA tests ont renvoyé un résultat valide, veuillez utiliser les tests suivants pour confirmer :

- Obtenez le type d'instance à l'aide des métadonnées d'EC2instance :

```
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

    # Information on how the version come from logs
    #
    # ip-172-31-27-205:6427:6427 [0] NCCL INFO cudaDriverVersion 12020
    # NCCL version 2.16.2+cuda11.8
    # ip-172-31-27-205:6427:6820 [0] NCCL INFO NET/OFI Initializing aws-ofi-nccl
 1.7.1-aws
    # ip-172-31-27-205:6427:6820 [0] NCCL INFO NET/OFI Using CUDA runtime version
 11060

    # cudaDriverVersion 12020  --> This is max supported cuda version by nvidia
 driver
    # NCCL version 2.16.2+cuda11.8 --> This is NCCL version compiled with cuda
 version
    # Using CUDA runtime version 11060 --> This is selected cuda version

    # Validation of logs
    grep "NET/OFI Using CUDA runtime version ${CUDA_RUNTIME_VERSION}" ${TRAINING_LOG}
 || { echo "Runtime cuda text not found"; exit 1; }
    grep "NET/OFI Initializing aws-ofi-nccl" ${TRAINING_LOG} || { echo "aws-ofi-nccl
 is not working, please check if it is installed correctly"; exit 1; }
    grep "NET/OFI Configuring AWS-specific options" ${TRAINING_LOG} || { echo "AWS-
specific options text not found"; exit 1; }
    grep "Using network AWS Libfabric" ${TRAINING_LOG} || { echo "AWS Libfabric 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/AWS Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Text not found:
 NET/AWS 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; }
        grep "aws-ofi-nccl/xml/p4d-24xl-topo.xml" ${TRAINING_LOG} || { echo "Topology
 file not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p4de.24xlarge" ]]; then
        grep "NET/AWS 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; }
        grep "aws-ofi-nccl/xml/p4de-24xl-topo.xml" ${TRAINING_LOG} || { echo
 "Topology file not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p5.48xlarge" ]]; then
        grep "NET/AWS 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; }
        grep "aws-ofi-nccl/xml/p5.48xl-topo.xml" ${TRAINING_LOG} || { echo "Topology
 file 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}"
```

GPUSurveillance et optimisation

La section suivante vous guidera à travers les options GPU d'optimisation et de surveillance. 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

Vous êtes DLAMI livré avec plusieurs outils de GPU surveillance préinstallés. Ce manuel indique également les outils disponibles afin d'être téléchargés et installés.

- Moniteur GPUs avec CloudWatch- un utilitaire préinstallé qui transmet les statistiques GPU d'utilisation à Amazon CloudWatch.
- nvidia-smi CLI - un utilitaire permettant de surveiller l'utilisation globale GPU du calcul et de la mémoire. Il est préinstallé sur votre AWS Apprentissage profond (deep learning) AMIs (DLAMI).
- NVMLbibliothèque C : une bibliothèque basée sur le C permettant d'accéder directement API aux fonctions GPU de surveillance et de gestion. Il est utilisé par le nvidia-smi CLI sous le capot et est préinstallé sur votre. DLAMI Elle comporte également les liaisons Perl et Python pour faciliter le développement dans ces langages. L'utilitaire gpumon.py préinstallé sur votre ordinateur DLAMI utilise le package pynvml de. nvidia-ml-py
- NVIDIADCGM- Un outil de gestion de clusters. Visitez la page destinée aux développeurs pour apprendre à installer et à configurer cet outil.

> **Tip**
>
> Consultez le blog NVIDIA des développeurs pour obtenir les dernières informations sur l'utilisation des CUDA outils installés sur votre DLAMI :
>
> - Surveillance de TensorCore l'utilisation à l'aide de Nsight IDE et nvprof.

Moniteur GPUs avec CloudWatch

Lorsque vous utilisez votre DLAMI with a, GPU vous trouverez peut-être que vous recherchez des moyens de suivre son utilisation pendant l'entraînement ou 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 GPU les métriques 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 le vôtre DLAMI à l' CloudWatch agent unifié pour configurer GPU les métriques et surveiller l'utilisation des GPU coprocessus dans les instances EC2 accélérées Amazon.

Il existe quatre manières de configurer GPUles métriques avec votre DLAMI :

- Configurer des GPU métriques minimales
- Configurer des GPU métriques partielles
- Configurer toutes les GPU métriques disponibles
- Configurer des GPU métriques 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 IAM autorisations d'EC2instance Amazon qui permettent à votre instance d'envoyer des métriques à CloudWatch. Pour connaître les étapes détaillées, voir Création de IAM rôles et d'utilisateurs à utiliser avec l' CloudWatch agent.

Configurer des GPU métriques minimales

Configurez GPU des métriques 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 GPU métriques 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 GPU métriques partielles

Configurez GPU des métriques partielles 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 GPU métriques 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 GPU métriques disponibles

Configurez toutes les GPU métriques 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 GPU mesures 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 GPU métriques 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 version actuelle DLAMI avec les derniers correctifs de sécurité en fonction du système d'exploitation de votre choix.

Amazon Linux 2

`yum`À utiliser pour obtenir les derniers correctifs de sécurité des AWS CloudWatch agents pour Amazon Linux 2DLAMI.

```
sudo yum update
```

Ubuntu

Pour obtenir les derniers correctifs AWS CloudWatch de sécurité pour Ubuntu, il est nécessaire de réinstaller l' AWS CloudWatch agent à l'aide d'un lien de téléchargement Amazon S3. DLAMI

```
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 appelé gpumon.py est préinstallé sur votreDLAMI. Il s'intègre CloudWatch et prend en charge la surveillance par GPU utilisation : GPU mémoire, GPU température et GPU alimentation. 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 GPU surveillance avec CloudWatch

1. Créez un IAM utilisateur 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 IAM politique 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 IAM d'un utilisateur et l'ajout de politiques pour CloudWatch, consultez la CloudWatch documentation.

2. Sur votreDLAMI, exécutez AWS configure et spécifiez les informations IAM d'identification de l'utilisateur.

```
$ aws configure
```

3. Vous devrez peut-être modifier l'utilitaire gpumon avant de l'exécuter. Vous pouvez trouver l'utilitaire gpumon et 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 :

   - Modifiez la région dans le fichier gpumon.py si votre instance se trouve NOT dans us-east-1.
   - 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 DLAMI général.

```
$ source activate python3
```

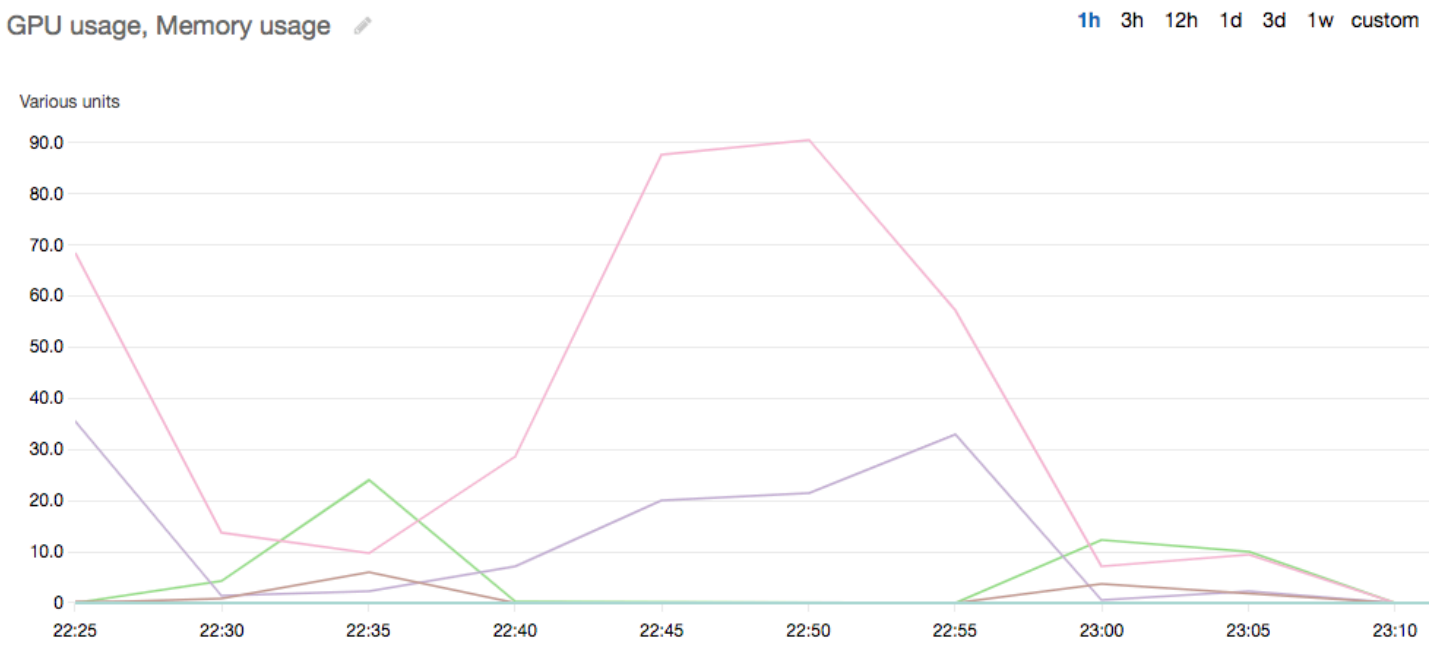
5. Exécutez l'utilitaire gpumon en arrière-plan.

```
(python3)$ python gpumon.py &
```

6. Ouvrez votre navigateur pour https://console.aws.amazon.com/cloudwatch/sélectionner 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.



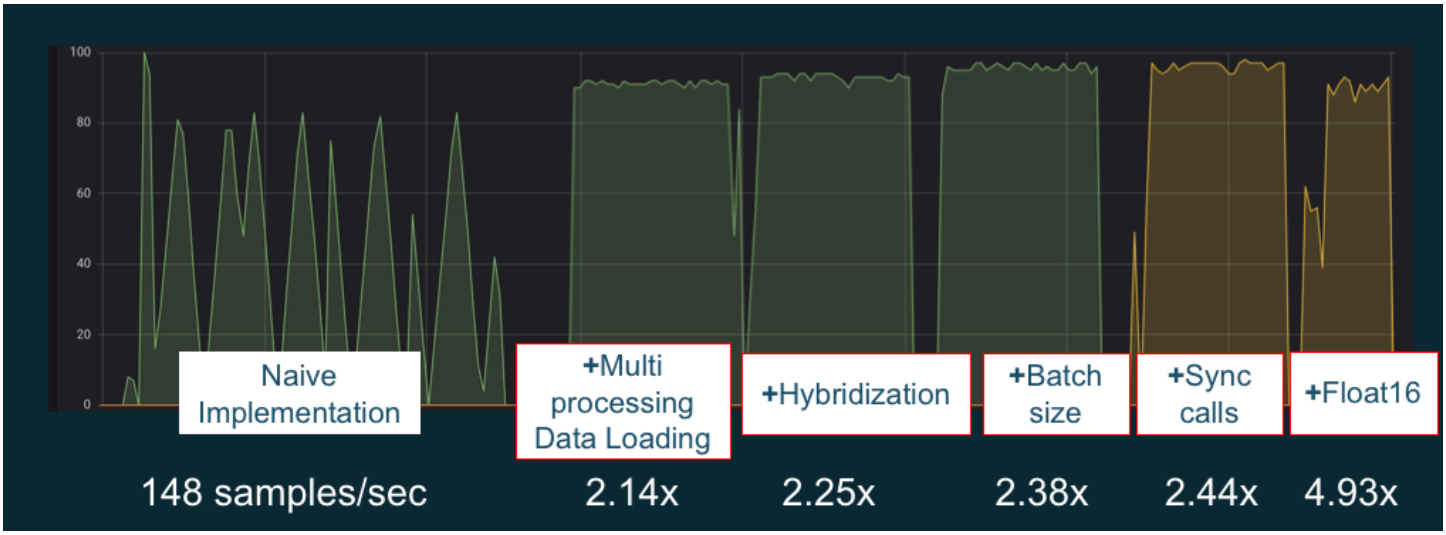
Ces autres sujets relatifs à la GPU surveillance et à l'optimisation pourraient vous intéresser :

- Surveillance
  - Moniteur GPUs avec CloudWatch
- Optimisation
  - Prétraitement
  - Entraînement

Optimisation

Pour en tirer le meilleur partiGPUs, vous pouvez optimiser votre pipeline de données et ajuster votre réseau de deep learning. Comme le décrit le tableau suivant, une implémentation naïve ou basique d'un réseau neuronal peut utiliser le réseau de GPU manière incohérente et ne pas exploiter tout son potentiel. En optimisant le prétraitement et le chargement des données, vous pouvez réduire le goulot d'étranglement entre votre et votreCPU. 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 GPU des performances. Source du graphique : Performance Tricks with MXNet Gluon

Les guides suivants présentent des options qui fonctionneront avec vous DLAMI et vous aideront à améliorer vos GPU performances.

Rubriques

- Prétraitement
- Entraînement

Prétraitement

Le prétraitement des données par le biais de transformations ou d'augmentations peut souvent être un processus CPU limité, ce qui peut constituer le goulot d'étranglement de votre pipeline global. Les frameworks ont des opérateurs intégrés pour le traitement des images, mais DALI (Data Augmentation Library) affiche des performances améliorées par rapport aux options intégrées des frameworks.

- NVIDIABibliothèque d'augmentation des données (DALI) : DALI télécharge l'augmentation des données vers leGPU. Il n'est pas préinstallé sur leDLAMI, mais vous pouvez y accéder en l'installant ou en chargeant un conteneur de framework compatible sur votre instance Amazon Elastic Compute Cloud DLAMI ou sur une autre instance. Consultez la page du DALI projet sur le NVIDIA site Web 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.

- nv JPEG : une bibliothèque de JPEG décodeurs GPU accélérés pour les programmeurs C. Il prend en charge le décodage d'images individuelles ou de lots ainsi que les opérations de transformation ultérieures courantes dans le deep learning. nv JPEG est intégré ou vous pouvez le télécharger depuis la NVIDIApage nvjpeg du site Web et l'utiliser séparément. DALI

Ces autres sujets relatifs à la GPU surveillance et à l'optimisation pourraient 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.

- NVIDIADeep Learning SDK : documentation sur le NVIDIA site Web décrivant la mise en œuvre à précision mixte pour MXNet PyTorch, et. 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 NVIDIA site du blog.
> - Entraînement à précision mixte avec float16 avec MXNet - un FAQ article sur le site Web. MXNet

- NVIDIAApex : un outil pour un entraînement facile à précision mixte avec PyTorch un article de blog sur le NVIDIA site Web.

Ces autres sujets relatifs à la GPU surveillance et à l'optimisation pourraient 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 pour AWS que vous puissiez l'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 AWS Neuron (SDK) 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 leDLAMI.

Table des matières

Lancer une DLAMI instance avec AWS Neuron

La dernière version DLAMI est prête à être utilisée avec AWS Inferentia et est fournie avec le package AWS NeuronAPI. Pour lancer une DLAMI instance, consultez la section Lancement et configuration d'une instanceDLAMI. Après avoir obtenu unDLAMI, suivez les étapes ci-dessous pour vous assurer que votre puce AWS Inferentia et vos ressources AWS Neuron sont actives.

Table des matières

- Affichage de l'utilisation des ressources
- Utilisation de Neuron Monitor (neuron-monitor)
- Mise à niveau du logiciel Neuron

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 :

- NEURONDEVICE: ID logique attribué au NeuronDevice. Cet ID est utilisé lors de la configuration de plusieurs environnements d'exécution pour en utiliser différents. NeuronDevices
- NEURONCORES: Le nombre de NeuronCores présents dans le NeuronDevice.
- NEURONMEMORY: La quantité de DRAM mémoire contenue dans le NeuronDevice.
- CONNECTEDDEVICES: Autre NeuronDevices connecté au NeuronDevice.
- PCIBDF: L'identifiant de fonction du périphérique de PCI bus (BDF) du NeuronDevice.

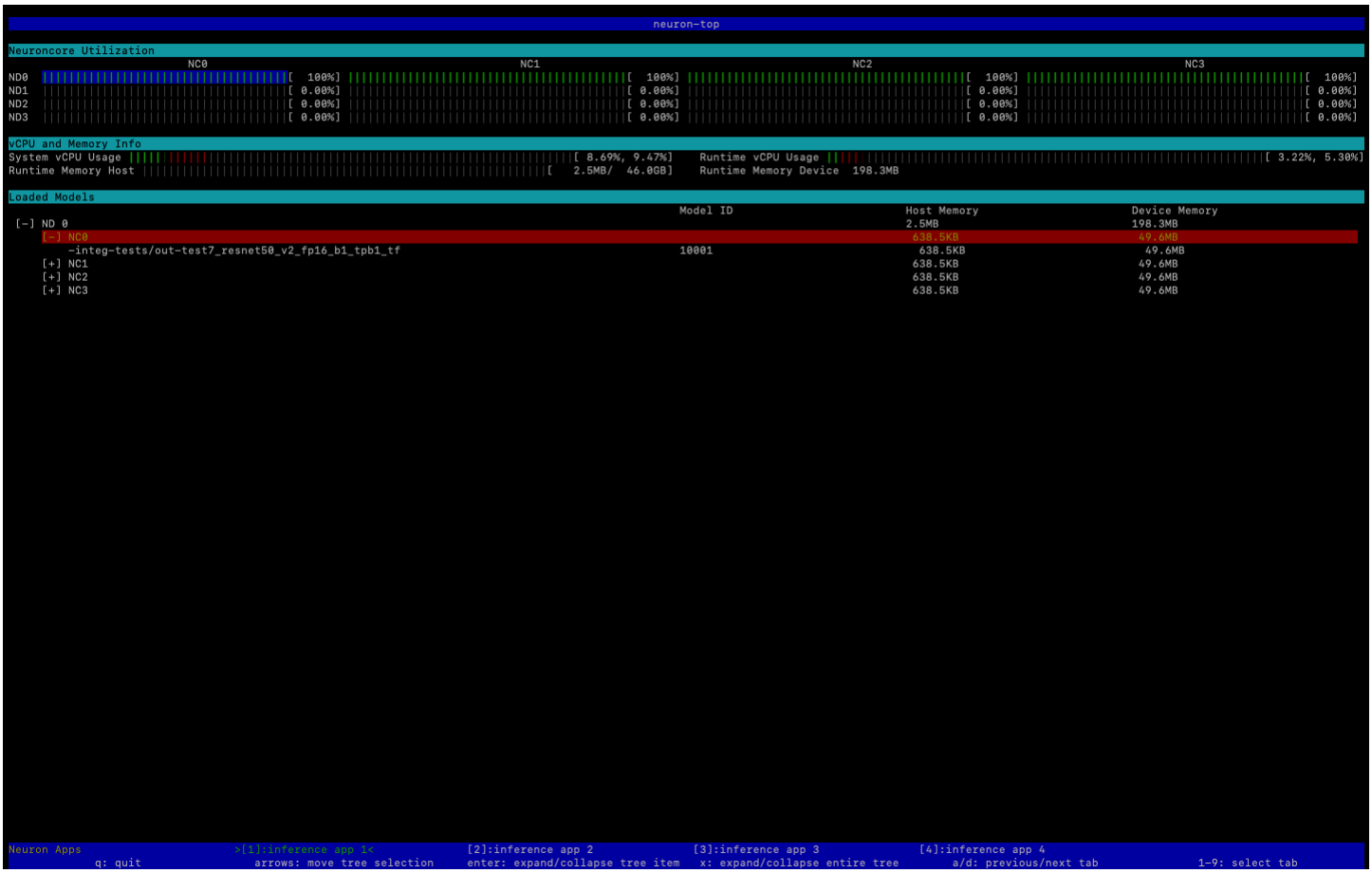
Affichage de l'utilisation des ressources

Affichez des informations utiles sur l'utilisation de NeuronCore et v, CPU l'utilisation de la mémoire, les modèles chargés et les applications Neuron à l'aide de la `neuron-top` commande. 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 au format stdout. 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 SDK logiciel Neuron intégréDLAMI, consultez le guide de configuration de AWS Neuron.

Étape suivante

Utilisation du DLAMI avec AWS Neuron

Utilisation du DLAMI avec AWS Neuron

Un flux de travail typique avec le AWS Neuron SDK 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 NeuronSDK, consultez la documentation du AWS Neuron SDK.

Table des matières

- ResNet50 Inférence

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancer une DLAMI instance avec AWS Neuron. Vous devez également être familiarisé avec le deep learning et l'utilisation duDLAMI.

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 méthode 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 binaire se trouve dans `/usr/local/bin/tensorflow_model_server_neuron` et est préinstallé dans leDLAMI.

Pour plus d'informations sur le NeuronSDK, consultez la documentation du AWS Neuron SDK.

Table des matières

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancer une DLAMI instance avec AWS Neuron. Vous devez également être familiarisé avec le deep learning et l'utilisation duDLAMI.

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 le dispositif Inferentia DRAM 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 un framework 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 en utilisant g RPC 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

La compilation MXNet -Neuron API fournit une méthode pour compiler un modèle de graphe que vous pouvez exécuter sur un appareil AWS Inferentia.

Dans cet exemple, vous utilisez le API pour compiler un modèle ResNet -50 et l'utiliser pour exécuter une inférence.

Pour plus d'informations sur le NeuronSDK, consultez la documentation du AWS Neuron SDK.

Table des matières

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancer une DLAMI instance avec AWS Neuron. Vous devez également être familiarisé avec le deep learning et l'utilisation duDLAMI.

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 la compilation Python MXNet -Neuron API 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). MMSest un easy-to-use outil flexible destiné à servir des modèles de deep learning formés à l'aide de n'importe quel framework d'apprentissage automatique ou d'apprentissage profond. Ce didacticiel inclut une étape de compilation à l'aide de AWS Neuron et une implémentation de l'MMSutilisation MXNet de.

Pour plus d'informations sur le NeuronSDK, consultez la documentation du AWS Neuron SDK.

Table des matières

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancer une DLAMI instance avec AWS Neuron. Vous devez également être familiarisé avec le deep learning et l'utilisation duDLAMI.

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 le en RESTful API utilisant les 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 le serveur modèle RESTful API et arrêtez le modèle 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

La compilation PyTorch -Neuron API 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 PyTorch SDKdocumentation AWS Neuron.

Table des matières

- Compilation Resnet50
- ResNet50 Inférence

Prérequis

Avant d'utiliser ce didacticiel, vous devez avoir terminé les étapes de configuration figurant dans Lancer une DLAMI instance avec AWS Neuron. Vous devez également être familiarisé avec le deep learning et l'utilisation duDLAMI.

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 la compilation Python PyTorch -Neuron API 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 ARM64GPUDLAMIssont 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 ARM64GPUDLAMIssont préconfigurés avec Docker, NVIDIA Docker, NVIDIA Driver, Cu CUDA DNNNCCL, ainsi qu'avec 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 GPU accélérés à un coût nettement inférieur à celui des instances x86 avec accélération. GPU

Sélectionnez un ARM64 DLAMI

Lancez une instance G5g avec l'instance ARM64 DLAMI de votre choix.

Pour step-by-step obtenir des instructions sur le lancement d'unDLAMI, reportez-vous à la section Lancement et configuration d'unDLAMI.

Pour obtenir la liste des plus récents ARM64DLAMIs, consultez les notes de publication de DLAMI.

Démarrer

Les rubriques suivantes vous montrent comment commencer à utiliser le ARM64DLAMI.

Table des matières

En utilisant le ARM64 GPU PyTorch DLAMI

AWS Apprentissage profond (deep learning) AMIs Il est prêt à être utilisé avec le processeur Arm64 et est GPUs optimisé pour. PyTorch ARM64GPU PyTorch DLAMIIl inclut un environnement Python

préconfiguré avec et TorchServepour PyTorchles TorchVisioncas d'utilisation de l'apprentissage profond et 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 MNIST de formation :

```
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 DLAMI l'aide des frameworks et des outils.

Outils d'inférence

- TensorFlow Servir

Service de modèle

Voici les options de service de modèles installées sur le Deep Learning AMI avec Conda. 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 Deep Learning AMI with Conda `tensorflow-serving-api` est préinstallé ! Vous trouverez un exemple de scripts pour entraîner, exporter et servir un MNIST modèle`~/examples/tensorflow-serving/`.

Pour exécuter l'un de ces exemples, connectez-vous d'abord à votre Deep Learning AMI avec Conda et activez l' TensorFlow environnement.

```
$ source activate tensorflow2_p310
```

Maintenant, accédez au dossier servant les exemples de scripts.

```
$ cd ~/examples/tensorflow-serving/
```

Service d'un modèle Inception préformé

Voici un exemple que vous pouvez tester pour servir différents modèles tels qu'Inception. En règle générale, vous avez besoin d'un modèle servable et de scripts clients déjà téléchargés sur votreDLAMI.

Service et test d'inférence avec un modèle Inception

1. Téléchargez le modèle.

   ```
   $ curl -O https://s3-us-west-2.amazonaws.com/tf-test-models/INCEPTION.zip
   ```

2. Décompactez le modèle.

```
$ unzip INCEPTION.zip
```

3. Tout d'abord, téléchargez une image de husky.

```
$ curl -O https://upload.wikimedia.org/wikipedia/commons/b/b5/Siberian_Husky_bi-
eyed_Flickr.jpg
```

4. Lancez le serveur. Notez que, pour Amazon Linux, vous devez modifier le répertoire utilisé pour `model_base_path` : remplacez `/home/ubuntu` par `/home/ec2-user`.

```
$ tensorflow_model_server --model_name=INCEPTION --model_base_path=/home/ubuntu/
examples/tensorflow-serving/INCEPTION/INCEPTION --port=9000
```

5. Le serveur étant en cours d'exécution au premier plan, vous devez lancer une autre session de terminal pour continuer. Ouvrez un nouveau terminal et activez-le TensorFlow avec`source activate tensorflow2_p310`. Utilisez ensuite votre éditeur de texte préféré pour créer un script avec le contenu suivant. Nommez-la `inception_client.py`. Ce script prendra comme paramètre un nom de fichier d'image et obtiendra un résultat de prédiction à partir du modèle préformé.

```
from __future__ import print_function

import grpc
import tensorflow as tf
import argparse

from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc

parser = argparse.ArgumentParser(
    description='TF Serving Test',
    formatter_class=argparse.ArgumentDefaultsHelpFormatter
)
parser.add_argument('--server_address', default='localhost:9000',
                    help='Tenforflow Model Server Address')
parser.add_argument('--image', default='Siberian_Husky_bi-eyed_Flickr.jpg',
                    help='Path to the image')
args = parser.parse_args()


def main():
```

```
  channel = grpc.insecure_channel(args.server_address)
  stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
  # Send request
  with open(args.image, 'rb') as f:
    # See prediction_service.proto for gRPC request/response details.
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'INCEPTION'
    request.model_spec.signature_name = 'predict_images'

    input_name = 'images'
    input_shape = [1]
    input_data = f.read()
    request.inputs[input_name].CopyFrom(
      tf.make_tensor_proto(input_data, shape=input_shape))

    result = stub.Predict(request, 10.0)  # 10 secs timeout
    print(result)

  print("Inception Client Passed")


if __name__ == '__main__':
  main()
```

6. 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.

```
$ python3 inception_client.py --server=localhost:9000 --image Siberian_Husky_bi-
eyed_Flickr.jpg
```

Former et servir un MNIST mannequin

Dans le cadre de ce didacticiel, nous allons exporter un modèle, puis le servir avec l'application `tensorflow_model_server`. Enfin, vous pouvez tester le modèle de serveur avec un exemple de script client.

Exécutez le script qui entraînera et exportera un MNIST modèle. En tant que seul argument du script, vous devez fournir un emplacement dossier pour qu'il y enregistre le modèle. Pour l'instant, nous le plaçons simplement dans `mnist_model`. Le script crée le dossier pour vous.

```
$ python mnist_saved_model.py /tmp/mnist_model
```

Soyez patient, car le script peut prendre un certain temps avant de fournir une sortie. Lorsque la formation est terminée et que le modèle est enfin exportée, vous devriez voir les éléments suivants :

```
Done training!
Exporting trained model to mnist_model/1
Done exporting!
```

Votre prochaine étape consiste à exécuter `tensorflow_model_server` pour servir le modèle exporté.

```
$ tensorflow_model_server --port=9000 --model_name=mnist --model_base_path=/tmp/
mnist_model
```

Un script client est fourni pour vous permettre de tester le serveur.

Pour le tester, vous devez ouvrir une nouvelle fenêtre de terminal.

```
$ python mnist_client.py --num_tests=1000 --server=localhost:9000
```

Autres exemples et fonctions

Si vous souhaitez en savoir plus sur TensorFlow Serving, consultez le TensorFlow site Web.

Vous pouvez également utiliser TensorFlow Serving avec Amazon Elastic Inference. Consultez le guide sur l'utilisation d'Elastic Inference with TensorFlow Serving pour plus d'informations.

TorchServe

TorchServe est un outil flexible destiné à servir des modèles de deep learning exportés depuis PyTorch. TorchServe est préinstallé avec le Deep Learning AMI with 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 (AmazonEC2) avec Deep Learning AMI 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 paramètre de prédiction renvoie une prédiction JSON similaire aux cinq meilleures 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 DLAMI instance. 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 ordinateur DLAMI et des conseils sur la mise à jour logicielle de votreDLAMI.

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 compteDLAMI, 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 d'une DLAMI

DLAMIles images système sont régulièrement mises à jour pour tirer parti des nouvelles versions du framework d'apprentissage profond, CUDA des autres mises à jour logicielles et de l'optimisation des performances. Si vous utilisez une instance 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 joindre à de nouvellesDLAMI. Ainsi, vous pouvez mettre à niveau souvent, tout en minimisant le temps nécessaire pour transférer vos données.

> **Note**
>
> Lorsque vous attachez et déplacez EBS des volumes Amazon entre euxDLAMIs, vous devez placer le volume DLAMIs et le nouveau volume dans la même zone de disponibilité.

1. Utilisez Amazon EC2console pour créer un nouveau EBS volume Amazon. Pour obtenir des instructions détaillées, consultez Création d'un EBS volume Amazon.
2. Joignez le EBS volume Amazon que vous venez de créer à votre volume existantDLAMI. Pour obtenir des instructions détaillées, consultez Joindre un EBS volume Amazon.
3. Transférez vos données, comme des ensembles de données, des points de contrôle et des fichiers de configuration.
4. Lancez unDLAMI. Pour obtenir des instructions complètes, consultez Configuration d'une DLAMI instance.
5. Détachez le EBS volume Amazon de votre ancien. DLAMI Pour obtenir des instructions détaillées, consultez Détacher un EBS volume Amazon.
6. Joignez le EBS volume Amazon à votre nouveauDLAMI. Suivez les instructions de l'étape 2 pour attacher le volume.
7. Après avoir vérifié que vos données sont disponibles sur votre nouveauDLAMI, arrêtez et résiliez l'ancienDLAMI. Pour obtenir des instructions de nettoyage, consultez Nettoyage d'une DLAMI instance.

Astuces pour les mises à jour de logiciels

De temps en temps, vous souhaiterez peut-être mettre à jour manuellement le logiciel de votreDLAMI. 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 le Deep Learning AMI with 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 d'une mise à jour du 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 ». DLAMIgarantit que tous les environnements DLAMI fournis 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'une nouvelle DLAMI est publiée. Les notifications sont publiées avec Amazon SNS à l'aide de la rubrique suivante.

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

Les messages sont publiés ici lorsqu'une nouvelle publication DLAMI est publiée. La version, les métadonnées et les AMI identifiants régionaux du 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 SNSconsole Amazon.
2. Dans la barre de navigation, remplacez la AWS région par US West (Oregon), si nécessaire. Vous devez sélectionner la région dans laquelle la SNS notification 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 le sujet ARN, copiez et collez le nom de ressource Amazon suivant (ARN) : **`arn:aws:sns:us-west-2:767397762724:dlami-updates`**

   b. Pour le protocole, choisissez-en un parmi [AmazonSQS, 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 son utilisationDLAMI. Les rubriques suivantes expliquent comment procéder à la configuration DLAMI pour atteindre 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 DLAMI ressources.

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 la section Confidentialité des données FAQ. Pour plus d'informations sur la protection des données en Europe, consultez le modèle de responsabilitéAWS partagée et le billet de GDPR blog sur le blog sur la AWS sécurité.

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) pour chaque compte.
- UtilisezSSL/TLSpour communiquer avec les AWS ressources. Nous avons besoin de la TLS version 1.2 et recommandons la TLS version 1.3.
- Configuration API et 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 FIPS 140 à 3 modules cryptographiques validés pour accéder AWS via une interface de ligne de commande ou unAPI, utilisez un point de terminaison. FIPS Pour plus d'informations sur les FIPS points de terminaison disponibles, voir Federal Information Processing Standard (FIPS) 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 d'autres Services AWS utilisateurs de la consoleAPI, 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 un URL à un serveur externe, nous vous recommandons vivement de ne pas inclure d'informations d'identification dans le URL afin de valider votre demande auprès de 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. IAMles administrateurs contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les DLAMI ressources. IAMest un Service AWS ventilateur 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
- IAMavec 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 que Utilisateur racine d'un compte AWS, en tant qu'IAMutilisateur ou en assumant un IAM rôle.

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 (IAMIdentity 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 en tant qu'identité fédérée, votre administrateur a préalablement configuré la fédération d'identité à l'aide de IAM rôles. 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 la section 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 vous-même les demandes, consultez la section Signature des AWS API demandes dans le guide de IAM l'utilisateur.

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 en savoir plus, consultez Authentification multifactorielle dans le guide de AWS IAM Identity Center l'utilisateur et Utilisation de l'authentification multifactorielle (MFA) AWS dans le guide de l'IAMutilisateur.

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 nécessitent que vous vous connectiez en tant qu'utilisateur root, consultez la section Tâches nécessitant des informations d'identification utilisateur root dans le Guide de IAM l'utilisateur.

Utilisateurs et groupes IAM

Un IAMutilisateur est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des IAM utilisateurs dotés d'informations d'identification à long terme, telles que des mots de passe et des clés d'accès. Toutefois, si vous avez des cas d'utilisation spécifiques qui nécessitent des informations d'identification à long terme auprès des IAM utilisateurs, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, voir Rotation régulière des clés d'accès

pour les cas d'utilisation nécessitant des informations d'identification à long terme dans le Guide de IAM l'utilisateur.

Un IAMgroupe est une identité qui définit un ensemble d'IAMutilisateurs. 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 IAM des ressources.

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 en savoir plus, voir Quand créer un IAM utilisateur (au lieu d'un rôle) dans le Guide de IAM l'utilisateur.

IAMrôles

Un IAMrôle est une identité au sein de Compte AWS vous dotée d'autorisations spécifiques. Il est similaire à un IAM utilisateur, mais n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un IAM rôle dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une AWS API opération AWS CLI or ou en utilisant une option personnaliséeURL. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez la section Méthodes pour assumer un rôle dans le Guide de IAM l'utilisateur.

IAMles rôles dotés d'informations d'identification temporaires sont utiles dans les situations suivantes :

- 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 plus d'informations sur les rôles pour la fédération, voir Création d'un rôle pour un fournisseur d'identité tiers dans le guide de IAM l'utilisateur. Si vous utilisez IAM Identity Center, vous configurez un ensemble d'autorisations. Pour contrôler les accès auxquels vos identités peuvent accéder après leur authentification, IAM Identity Center met en corrélation l'ensemble d'autorisations avec un rôle dans. IAM Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations IAM utilisateur temporaires : un IAM utilisateur ou un rôle peut assumer un IAM rôle afin d'obtenir temporairement différentes autorisations pour une tâche spécifique.

- Accès entre comptes : vous pouvez utiliser un IAM rôle pour autoriser une personne (un mandant fiable) d'un autre compte à 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 connaître la différence entre les rôles et les politiques basées sur les ressources pour l'accès entre comptes, voir Accès aux ressources entre comptes IAM dans le guide de l'IAMutilisateur.
- 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 transmises (FAS) — Lorsque vous utilisez un IAM utilisateur ou un rôle 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. FASutilise les autorisations du principal appelant an Service AWS, combinées à la demande Service AWS pour adresser des demandes aux services en aval. FASles demandes 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 les politiques relatives FAS aux demandes, consultez la section Transférer les sessions d'accès.
  - Rôle de service — Un rôle de service est un IAMrôle qu'un service assume pour effectuer des actions en votre nom. Un IAM administrateur peut créer, modifier et supprimer un rôle de service de l'intérieurIAM. Pour plus d'informations, consultez la section Création d'un rôle auquel déléguer des autorisations Service AWS dans le Guide de IAM l'utilisateur.
  - 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 IAM administrateur peut consulter, mais pas modifier les autorisations pour les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un IAM rôle pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui font AWS CLI des AWS API demandes. Cela est préférable au stockage des clés d'accès dans l'EC2instance. 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'EC2instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez la section Utilisation

d'un IAM rôle pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le Guide de IAM l'utilisateur.

Pour savoir s'il faut utiliser IAM des rôles ou des IAM utilisateurs, voir Quand créer un IAM rôle (au lieu d'un utilisateur) dans le guide de IAM l'utilisateur.

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 JSON documents. Pour plus d'informations sur la structure et le contenu des documents de JSON politique, voir Présentation des JSON politiques dans le guide de IAM l'utilisateur.

Les administrateurs peuvent utiliser AWS JSON des politiques 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 autoriser les utilisateurs à effectuer des actions sur les ressources dont ils ont besoin, un IAM administrateur peut créer des IAM politiques. L'administrateur peut ensuite ajouter les IAM politiques aux rôles, et les utilisateurs peuvent assumer les rôles.

IAMles politiques définissent les autorisations pour une action, quelle que soit la méthode que vous utilisez pour effectuer l'opération. Par exemple, supposons que vous disposiez d’une politique qui autorise l’action `iam:GetRole`. Un utilisateur doté de cette politique peut obtenir des informations sur le rôle auprès du AWS Management Console AWS CLI, ou du AWS API.

Politiques basées sur l’identité

Les politiques basées sur l'identité sont JSON des documents de politique d'autorisation que vous pouvez joindre à une identité, telle qu'un IAM utilisateur, un groupe d'utilisateurs ou un rôle. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour savoir comment créer une politique basée sur l'identité, consultez la section Création de IAM politiques dans le Guide de l'IAMutilisateur.

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 savoir comment choisir entre une politique gérée ou une politique intégrée, voir Choisir entre des politiques gérées et des politiques intégrées dans le Guide de l'IAMutilisateur.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents JSON de stratégie que vous attachez à une ressource. Les politiques de confiance dans les IAM rôles et les politiques relatives aux compartiments Amazon S3 sont des exemples de 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 de politiques AWS gérées depuis une IAM 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. ACLssont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format du document JSON de stratégie.

Amazon S3 et Amazon VPC sont des exemples de services compatiblesACLs. AWS WAF Pour en savoir plusACLs, 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.

- Limites d'autorisations — Une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une IAM entité (IAMutilisateur ou rôle). 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 la section Limites d'autorisations pour les IAM entités dans le Guide de IAM l'utilisateur.
- Politiques de contrôle des services (SCPs) : SCPs JSON politiques 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. Les SCP limites d'autorisations pour les entités présentes dans les comptes des membres, y compris chacune d'entre elles Utilisateur racine d'un compte AWS. Pour plus d'informations sur les OrganizationsSCPs, voir Politiques de contrôle des services 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 la section Politiques de session dans le guide de IAM l'utilisateur.

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 IAM l'utilisateur.

IAMavec Amazon EMR

Vous pouvez utiliser IAM Amazon EMR pour définir les utilisateurs, les AWS ressources, les groupes, les rôles et les politiques. Vous pouvez également contrôler les utilisateurs et les rôles auxquels Services AWS ces utilisateurs et rôles peuvent accéder.

Pour plus d'informations sur l'utilisation IAM avec AmazonEMR, consultez AWS Identity and Access Management 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 une 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

Votre responsabilité en matière de conformité lors de l'utilisation DLAMI est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les 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 AmazonEC2. 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 DLAMI instance est fournie avec plusieurs outils GPU de surveillance, notamment un utilitaire qui transmet les statistiques GPU d'utilisation à Amazon CloudWatch. Pour

plus d'informationsGPUSurveillance et optimisation, consultez et 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 DLAMI type et les informations du système d'exploitation.

> **Note**
>
> AWS ne collecte ni ne conserve aucune autre information sur leDLAMI, telle que les commandes que vous utilisez dans leDLAMI.

- Amazon Linux 2
- Ubuntu 20.04
- Ubuntu 18.04
- Ubuntu 16.04

Pour désactiver le suivi de l'utilisation

Si vous le souhaitez, vous pouvez désactiver le suivi de l'utilisation pour une nouvelle DLAMI instance. Pour vous désinscrire, vous devez ajouter une balise 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.

DLAMIpolitique de soutien au cadre

Vous trouverez ici les détails de la politique de support pour AWS Apprentissage profond (deep learning) AMIs (DLAMI) cadres.

Pour obtenir la liste des DLAMI frameworks qui AWS prend actuellement en charge, consultez la page Politique de support du DLAMI Framework. Dans les tableaux de cette page, gardez à l'esprit les points suivants :

- 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 AWS prend en charge cette version du framework.

Pour obtenir des informations détaillées sur des sujets spécifiquesDLAMIs, consultezNotes de mise à jour pour DLAMIs.

DLAMIsupport du cadre FAQs

- Quelles versions du framework reçoivent des correctifs de sécurité ?
- Ce que font les images AWS publier lorsque de nouvelles versions du framework sont publiées ?
- Quelles sont les nouvelles images SageMaker/AWS fonctionnalités ?
- Comment est définie la version actuelle dans le tableau Supported Frameworks ?
- Et si j'utilise une version qui ne figure pas dans le tableau Supported Frameworks ?
- Les versions précédentes de sont-elles prises en DLAMIs charge TensorFlow ?
- 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 ?
- Mon instance sera-t-elle mise en place pendant que ma charge de travail est en cours d'exécution ?
- Que se passe-t-il lorsqu'une nouvelle version du framework corrigée ou mise à jour est disponible ?
- Les dépendances sont-elles mises à jour sans modifier la version du framework ?
- Quand le support actif pour ma version de framework prend-il fin ?

- Les images dont les versions du framework ne sont plus activement maintenues seront-elles corrigées ?
- Comment utiliser une ancienne version du framework ?
- Comment puis-je suivre les modifications apportées up-to-date aux frameworks et à leurs versions ?
- Ai-je besoin d'une licence commerciale pour utiliser le référentiel Anaconda ?

Quelles versions du framework reçoivent des correctifs de sécurité ?

Si la version du framework est étiquetée Supportée dans le AWS Apprentissage profond (deep learning) AMIs Tableau des politiques de support du framework, il reçoit des correctifs de sécurité.

Ce que font les images AWS publier 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 sont les nouvelles images SageMaker/AWS fonctionnalités ?

Les nouvelles fonctionnalités sont généralement publiées dans la dernière version de DLAMIs for PyTorch et TensorFlow. Reportez-vous aux notes de publication d'une image spécifique pour plus de détails sur les nouveautés SageMaker ou AWS fonctionnalités. Pour obtenir la liste des produits disponiblesDLAMIs, consultez les notes de publication pour 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 dans le AWS Apprentissage profond (deep learning) AMIs Le tableau des politiques de support du framework fait référence à la dernière version du framework qui AWS met à disposition sur GitHub. Chaque dernière version inclut des mises à jour des pilotes, des bibliothèques

et des packages pertinents duDLAMI. 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 Supported Frameworks ?

Si vous utilisez une version qui ne figure pas dans AWS Apprentissage profond (deep learning) AMIs Tableau des politiques de support du framework, vous ne disposez peut-être pas des pilotes, bibliothèques et packages pertinents les plus récents. Pour une up-to-date version ultérieure, nous vous recommandons de passer à l'un des frameworks pris en charge disponibles en utilisant la dernière version DLAMI de votre choix. Pour obtenir la liste des produits disponiblesDLAMIs, consultez les notes de publication pour DLAMI.

Les versions précédentes de sont-elles prises en DLAMIs charge TensorFlow ?

Non Nous prenons en charge la dernière version de correctif de la dernière version majeure de chaque framework publiée 365 jours après sa GitHub sortie initiale, comme indiqué dans le AWS Apprentissage profond (deep learning) AMIs Tableau des politiques de support du cadre. Pour plus d'informations, consultez Et si j'utilise une version qui ne figure pas dans le tableau Supported Frameworks ?.

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, récupérez l'DLAMIID et utilisez-le pour le lancer à l'DLAMIaide de la EC2console. Pour échantillon AWS CLIcommandes pour récupérer le AWS Apprentissage profond (deep learning) AMIs ID, reportez-vous à la page des notes de DLAMI publication, des notes de DLAMImise à jour du framework unique. La version du framework que vous choisissez doit être étiquetée Supportée dans le AWS Apprentissage profond (deep learning) AMIs Tableau des politiques de support du cadre.

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 deCUDA, nous publions une nouvelle version DLAMI pour cette version de TensorFlow avec prise en charge de la nouvelle CUDA version.

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 ne DLAMI 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 ?

Consultez régulièrement la page des notes de publication correspondant à votre image. Nous vous encourageons à passer à de nouveaux frameworks corrigés ou mis à jour dès qu'ils seront disponibles. Pour obtenir la liste des produits disponiblesDLAMIs, consultez les notes de publication pour DLAMI.

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 DLAMI pour obtenir des informations de dépendance mises à jour.

Quand le support actif pour ma version de framework prend-il fin ?

DLAMIles images 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 DLAMI régulièrement la page des notes de publication et de procéder à une mise à niveau lorsque des modifications sont apportées.

Mises à niveau de la version du framework (correctif)

Si vous avez une DLAMI charge de travail basée sur la version TensorFlow 2.7.0 et que vous TensorFlow publiez la version 2.7.1 le GitHub AWS publie une nouvelle version DLAMI avec la TensorFlow version 2.7.1. 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 with TensorFlow 2.7.0 ne reçoit pas d'autres correctifs. La page des notes de DLAMI publication de la version TensorFlow 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 nouveautés DLAMIs créées à la suite de mises à niveau de correctifs sont désignées par un nouvel AMIidentifiant.

AWS correctifs de sécurité

Si vous avez une charge de travail basée sur une image avec TensorFlow 2.7.0 et AWS crée un correctif de sécurité, puis une nouvelle version de celui-ci DLAMI est publiée pour la version TensorFlow 2.7.0. 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 la dernière versionDLAMI, voir Comment puis-je trouver la dernière image corrigée pour une version de framework prise en charge ?

Les nouveautés DLAMIs créées à la suite de mises à niveau de correctifs sont désignées par un nouvel AMIidentifiant.

Date de fin de mise à jour (Aging out)

DLAMIsont 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, une nouvelle version DLAMI avec la version mise à jour est requise. La version DLAMI avec l'ancien framework n'est plus activement maintenue.

> **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 DLAMI image TensorFlow 2.7.0 avec Python 3.6 et que cette version de Python est marquée pour end-of-support, toutes les DLAMI images 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 DLAMI images dépendantes 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'DLAMIID et utilisez-le pour lancer le à l'DLAMIaide de la EC2console. Dans AWS CLIcommandes pour récupérer l'AMIID, reportez-vous à la page des notes de version dans les notes de DLAMI version du framework unique.

Comment puis-je suivre les modifications apportées up-to-date aux frameworks et à leurs versions ?

Restez up-to-date fidèle aux DLAMI frameworks et aux versions utilisant le AWS Apprentissage profond (deep learning) AMIs Tableau de la politique de support du framework, notes DLAMI de publication.

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.

Modifications importantes apportées au NVIDIA pilote DLAMIs

Le 15 novembre 2023, AWS a apporté des modifications importantes à AWS Apprentissage profond (deep learning) AMIs (DLAMI) lié au NIVIDA pilote qui DLAMIs utilise. Pour plus d'informations sur ce qui a changé et si cela a une incidence sur votre utilisation deDLAMIs, consultezDLAMINVIDIAchangement de pilote FAQs.

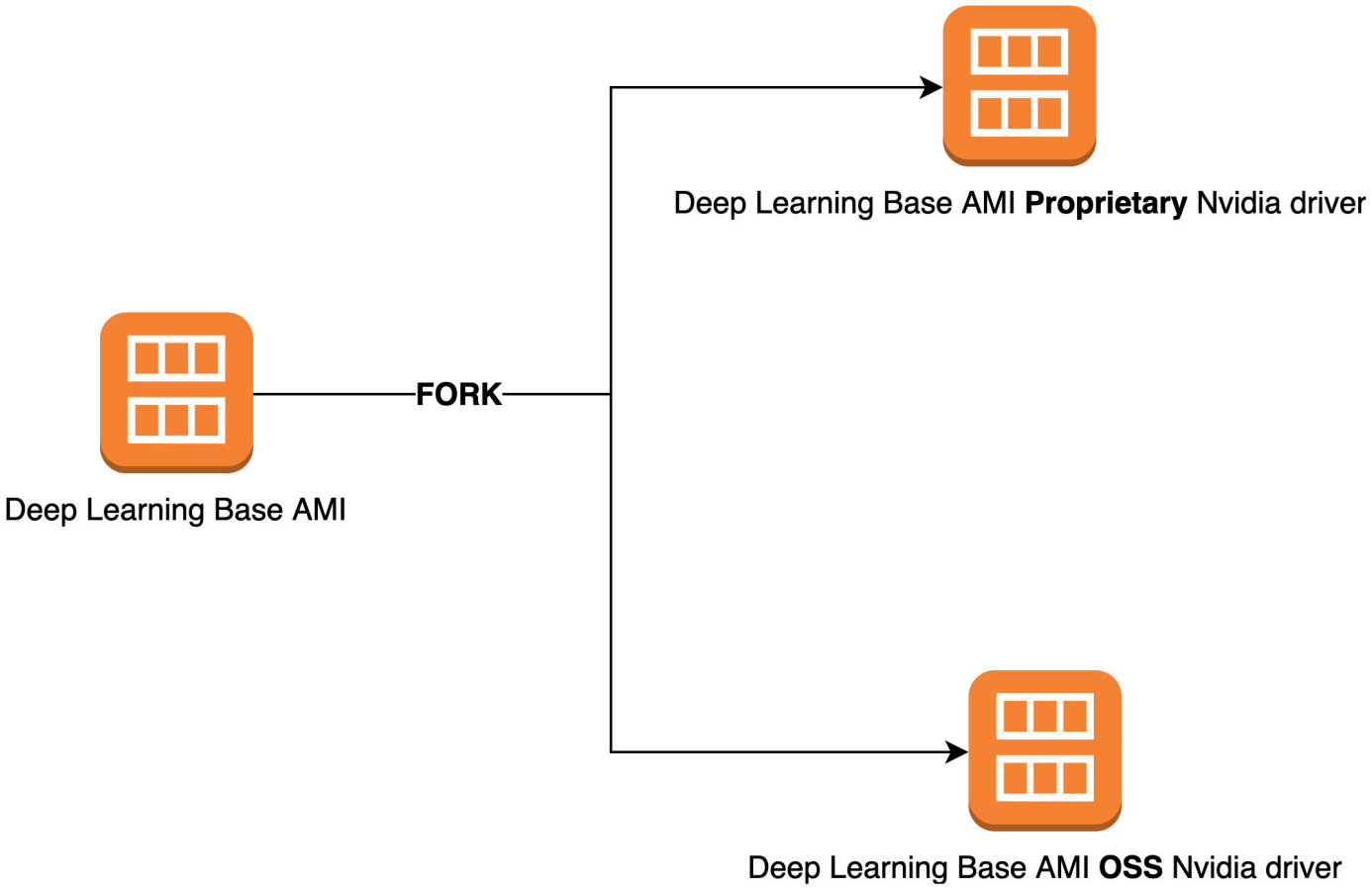
DLAMINVIDIAchangement de pilote FAQs

Qu'est-ce qui a changé ?

Nous nous sommes DLAMIs divisés en deux groupes distincts :

- DLAMIsqui utilisent un pilote NVIDIA propriétaire (pour prendre en charge P3, P3dn, G3)
- DLAMIsqui utilisent un NVIDIA OSS pilote (pour prendre en charge G4dn, G5, P4, P5)

En conséquence, nous en avons créé de nouvelles DLAMIs pour chacune des deux catégories avec de nouveaux noms et de nouveaux noms AMIIDs. 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, DLAMI celui qui prend en charge le P5 ne prend pas en charge le G3, et DLAMI celui qui prend en charge le G3 ne prend pas en charge le P5.



Pourquoi ce changement a-t-il été nécessaire ?

Auparavant, DLAMIs pour NVIDIA GPUs inclure un pilote de noyau propriétaire deNVIDIA. Cependant, la communauté du noyau Linux en amont a accepté une modification qui empêche les pilotes de noyau propriétaires, tels que le NVIDIA GPU pilote, de communiquer avec d'autres pilotes de noyau. Cette modification désactive les instances GPUDirect RDMA des séries P4 et P5, qui sont le mécanisme qui permet une utilisation efficace GPUs EFA pour la formation distribuée. Par conséquent, utilisez DLAMIs maintenant le pilote OpenRM (pilote NVIDIA open source), lié aux EFA pilotes open source pour prendre en charge 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'DLAMIsest-ce que ce changement a affecté ?

Ce changement a touché tout le mondeDLAMIs.

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 (AmazonEC2) compatible. Pour déterminer les types d'EC2instances pris DLAMI en charge par a, consultez les notes de publication correspondantesDLAMI, puis recherchez les EC2instances prises en charge. Pour obtenir la liste des DLAMI options actuellement prises en charge et des liens vers leurs notes de publication, consultezNotes de mise à jour pour DLAMIs.

De plus, vous devez utiliser le bon AWS Command Line Interface (AWS CLI) pour invoquer le courantDLAMIs.

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 précédentesDLAMIs, à condition que vous les exécutiez sur un type d'EC2instance compatible.

Ce changement a-t-il affecté les Deep Learning Containers ?

Non, cette modification n'a pas eu d'incidence AWS Deep Learning Containers, car ils n'incluent pas le NVIDIA pilote. 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 DLAMI en dehors du AWS Apprentissage profond (deep learning) AMIs . Activé AWS re:Post, consultez les questions DLAMI d'autres clients ou posez vos propres questions. Sur le AWS Blog Machine Learning et autres AWS blogs, lisez les articles officiels surDLAMI.

AWS re:Post

Tag : AWS Apprentissage profond (deep learning) AMIs

AWS Blogue

- 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 (APN) Blog | 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

Notes de mise à jour pour DLAMIs

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

Pour les notes de publication relatives aux DLAMI frameworks 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 des DLAMI frameworks.

> **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 progressifs dans les notes de publication officielles.

Base DLAMIs

GPU

- X86
  - AWS Base d'apprentissage approfondi AMI (Amazon Linux 2)
  - AWS Base d'apprentissage profond AMI (Ubuntu 22.04)
  - AWS Base d'apprentissage profond AMI (Ubuntu 20.04)
- ARM64
  - AWS Base d'apprentissage profond ARM64 AMI (Ubuntu 22.04)
  - AWS Base d'apprentissage approfondi ARM64 AMI (Amazon Linux 2)

AWS Neurone

- X86
  - AWS AMINeurone de base pour le Deep Learning (Amazon Linux 2)
  - AWS AMINeurone de base d'apprentissage profond (Ubuntu 20.04)

Qualcomm

- X86
  - AWS Base d'apprentissage profond Qualcomm AMI (Amazon Linux 2)

Cadre unique DLAMIs

PyTorch-spécifique AMIs

GPU

- X86
  - AWS Apprentissage profond AMI GPU PyTorch 2.4 (Ubuntu 22.04)
  - AWS Apprentissage profond AMI GPU PyTorch 2.3 (Ubuntu 20.04)
  - AWS Apprentissage profond AMI GPU PyTorch 2.3 (Amazon Linux 2)
  - AWS Apprentissage profond AMI GPU PyTorch 2.2 (Ubuntu 20.04)
  - AWS Apprentissage profond AMI GPU PyTorch 2.2 (Amazon Linux 2)
  - AWS Apprentissage profond AMI GPU PyTorch 1.13 (Amazon Linux 2)
  - AWS Apprentissage profond AMI GPU PyTorch 1.13 (Ubuntu 20.04)
- ARM64
  - AWS Apprentissage profond ARM64 AMI GPU PyTorch 2.4 (Ubuntu 22.04)
  - AWS Apprentissage profond ARM64 AMI GPU PyTorch 2.3 (Ubuntu 22.04)
  - AWS Apprentissage profond ARM64 AMI GPU PyTorch 2.2 (Ubuntu 20.04)

AWS Neurone

- X86
  - AWS AMINeuron d'apprentissage profond PyTorch 1.13 (Amazon Linux 2)
  - AWS AMINeuron d'apprentissage profond PyTorch 1.13 (Ubuntu 20.04)

TensorFlow-spécifique AMIs

GPU

- X86
  - AWS Apprentissage profond AMI GPU TensorFlow 2.17 (Ubuntu 22.04)

- AWS Apprentissage profond AMI GPU TensorFlow 2.16 (Amazon Linux 2)
- AWS Apprentissage profond AMI GPU TensorFlow 2.16 (Ubuntu 20.04)
- AWS Apprentissage profond AMI GPU TensorFlow 2.15 (Amazon Linux 2)
- AWS Apprentissage profond AMI GPU TensorFlow 2.15 (Ubuntu 20.04)

AWS Neurone

- X86
  - AWS AMINeuron d'apprentissage profond TensorFlow 2.10 (Amazon Linux 2)
  - AWS AMINeuron d'apprentissage profond TensorFlow 2.10 (Ubuntu 20.04)

Cadre multiple DLAMIs

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

GPU

- X86
  - AWS Apprentissage approfondi AMI (Amazon Linux 2)

AWS Neurone

- X86
  - AWS AMINeurone d'apprentissage profond (Amazon Linux 2023)
  - AWS AMINeurone d'apprentissage profond (Ubuntu 22.04)

Fonctionnalités obsolètes de DLAMI

Le tableau suivant répertorie les fonctionnalités obsolètes de AWS Apprentissage profond (deep learning) AMIs (DLAMI), la date à laquelle nous les avons déconseillés et des informations sur les raisons pour lesquelles nous les avons rendus obsolètes.

Fonctionnalité	Date	Détails
Ubuntu 16.04	07/10/2021	Ubuntu Linux 16.04 LTS a atteint la fin de sa période de LTS 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 la base d'apprentissage profond AMI (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 du Deep Learning AMI (Amazon Linux) dans les nouvelles versions depuis octobre 2021. Les versions précédentes du Deep Learning AMI (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 à DLAMI partir de juillet 2020. Les versions précédentes DLAMI contenant ces environnements continueront d'être disponibles. Nous fourniron s des mises à jour de ces environnements uniquemen t si des correctifs de sécurité sont publiés par la communauté open source pour ces infrastructures.
Python 3.6	15/06/2020	En raison de demandes de clients, nous passons à Python 3.7 pour de nouvelles versions TF/MX/ PT.
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 DLAMI versions et des modifications associées apportées au Guide du AWS Apprentissage profond (deep learning) AMIs développeur.

Changements récents

Modification	Description	Date
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	Le Deep Learning AMI avec Conda est désormais disponibl e en TensorFlow 2 avec CUDA 10.	3 décembre 2019
AWS Inférentie	Le Deep Learning prend AMI désormais en charge le matériel AWS Inferentia et le AWS SDK Neuron.	3 décembre 2019
Utiliser le TensorFlow service avec un modèle Inception	Un exemple d'utilisation de l'inférence avec un modèle Inception a été ajouté pour TensorFlow Serving, avec ou sans Elastic Inference.	28 novembre 2018
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 Deep Learning AMI with 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 DLAMI versions antérieures 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). Également, la première version d'un didacticiel de 10 minutes intitulé : « Getting Started with Deep Learning AMI ».	26 avril 2018
Didacticiel Chainer	Un didacticiel pour l'utilisation de Chainer en mode multiple GPUGPU, unique et en CPU mode unique a été ajouté. CUDAl'intégration a été mise à niveau de CUDA 8 à CUDA 9 pour plusieurs frameworks.	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 AMIet les CUDA fonctionnalités du framework décrites dans Conda et dans les CUDA aperçus. 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 la version 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
Deux AMI options Windows ont été ajoutées	AMIsPublication de Windows 2012 R2 et 2016 : ajout au guide de AMI sélection et aux notes de publication.	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.