aws

Entwicklerhandbuch

AWS Deep Learning AMIs

AWS Deep Learning AMIs: Entwicklerhandbuch

Table of Contents

Was ist das DLAMI? 1
Informationen zu diesem Handbuch 1
Voraussetzungen 1
Beispielanwendungsfälle 1
Features 2
Vorinstallierte Frameworks 2
Vorinstallierte GPU-Software 3
Bereitstellung und Visualisierung von Modellen 3
Versionshinweise zu DLAMI 4
P6 Unterstützt DLAMI 4
P6 Unterstützt DLAMI 4
GPU-Funktionstest 5
Base DLAMIs 8
X86 8
ARM64 63
Einzelnes Framework DLAMIs 77
PyTorch DLAMIs 77
TensorFlow DLAMIs 122
DLAMI mit mehreren Frameworks 131
X86 132
Erste Schritte 146
Einen DLAMI wählen 146
CUDA-Installationen und Framework-Bindungen 147
Base 148
Conda 149
Architektur 150
BS 150
Eine Instanz auswählen 151
Preisgestaltung 152
Verfügbarkeit in Regionen 153
GPU 153
CPU 154
Inferentia 155
Trainium 155

Einrichtung ... 157
Eine DLAMI-ID finden ... 157
Starten einer -Instance ... 159
Herstellen einer Verbindung mit einer Instance ... 161
Jupyter einrichten ... 161
Server sichern ... 162
Server wird gestartet ... 163
Client wird verbunden ... 164
Einloggen ... 165
Bereinigen ... 167
Verwendung eines DLAMI ... 169
Conda DLAMI ... 169
Einführung in das Deep Learning AMI mit Conda ... 169
Loggen Sie sich in Ihr DLAMI ein ... 170
Starten Sie die Umgebung TensorFlow ... 170
Wechseln Sie zur PyTorch Python-3-Umgebung ... 171
Entfernen von Umgebungen ... 172
Basis DLAMI ... 172
Verwenden des Deep Learning Base AMI ... 172
CUDA-Versionen konfigurieren ... 173
Jupyter Notebooks ... 173
Navigation der installierten Tutorials ... 174
Wechseln von Umgebungen mit Jupyter ... 175
Tutorials ... 175
Aktivieren von Frameworks ... 176
Elastic Fabric Adapter ... 179
GPU-Überwachung und -Optimierung ... 193
AWS Inferentia ... 203
ARM64 DLAMI ... 225
Inferenz ... 229
Modellbereitstellung ... 229
Ihr DLAMI aufrüsten ... 234
DLAMI-Upgrade ... 234
Software-Updates ... 235
Benachrichtigungen veröffentlichen ... 236
Sicherheit ... 238

Datenschutz ... 239
Identity and Access Management ... 240
Authentifizierung mit Identitäten ... 240
Verwalten des Zugriffs mit Richtlinien ... 244
IAM mit Amazon EMR ... 247
Compliance-Validierung ... 247
Ausfallsicherheit ... 248
Sicherheit der Infrastruktur ... 248
Überwachen ... 248
Nachverfolgung der Nutzung ... 249
DLAMI-Supportrichtlinie ... 250
DLAMI-Unterstützung FAQs ... 250
Welche Framework-Versionen erhalten Sicherheitspatches? ... 251
Welches Betriebssystem erhält Sicherheitspatches? ... 251
Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden? ... 251
Welche Bilder erhalten neue SageMaker AWS KI/Funktionen? ... 251
Wie ist die aktuelle Version in der Tabelle Unterstützte Frameworks definiert? ... 251
Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Version aufgeführt ist? ... 252
Werden frühere Patch-Versionen einer Framework-Version DLAMIs unterstützt? ... 252
Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version? ... 252
Wie oft werden neue Images veröffentlicht? ... 252
Wird meine Instance an Ort und Stelle gepatcht, während mein Workload läuft? ... 253
Was passiert, wenn eine neue gepatchte oder aktualisierte Framework-Version verfügbar ist? ... 253
Werden Abhängigkeiten aktualisiert, ohne die Framework-Version zu ändern? ... 253
Wann endet der aktive Support für meine Framework-Version? ... 253
Werden Images mit Framework-Versionen, die nicht mehr aktiv verwaltet werden, gepatcht? ... 255
Wie verwende ich eine ältere Framework-Version? ... 255
Wie bleibe ich up-to-date bei Support-Änderungen an Frameworks und deren Versionen? ... 255
Benötige ich eine kommerzielle Lizenz, um das Anaconda Repository nutzen zu können? ... 256
Tabelle mit den Richtlinien Support den DLAMI-Support ... 257
Unterstützte Framework-Versionen ... 257
Unterstützte Betriebssystemversionen ... 257

Nicht unterstützte Framework-Versionen 257
Nicht unterstützte Betriebssystemversionen 258
Archiv der DLAMI-Versionshinweise nicht unterstützt 259
Base 259
Ein einziges Framework 259
Multifunktionales Framework 261
Wichtige Änderungen 262
Änderung des DLAMI NVIDIA-Treibers FAQs 262
Was hat sich geändert? 262
Warum war diese Änderung erforderlich? 263
DLAMIs Worauf hat sich diese Änderung ausgewirkt? 264
Was bedeutet das für dich? 264
Gibt es bei der neueren Version einen Verlust an Funktionalität? DLAMIs 264
Hatte diese Änderung Auswirkungen auf Deep Learning Containers? 265
Ähnliche Informationen 266
Veraltete Funktionen 267
Dokumentverlauf 270
..... cclxxiii

Was ist AWS Deep Learning AMIs?

AWS Deep Learning AMIs (DLAMI) bietet maßgeschneiderte Maschinenbilder, die Sie für Deep Learning in der Cloud verwenden können. Sie DLAMIs sind in den meisten Versionen AWS-Regionen für eine Vielzahl von Amazon Elastic Compute Cloud (Amazon EC2) -Instance-Typen verfügbar, von einer kleinen reinen CPU-Instance bis hin zu den neuesten leistungsstarken Multi-GPU-Instances. DLAMIs Sie sind mit NVIDIA CUDA und NVIDIA cuDNN sowie den neuesten Versionen der beliebtesten Deep-Learning-Frameworks vorkonfiguriert.

Informationen zu diesem Handbuch

Der Inhalt von kann Ihnen helfen, das zu starten und zu verwenden. DLAMIs Der Leitfaden behandelt mehrere gängige Anwendungsfälle für Deep Learning, sowohl für Schulungen als auch für Inferenz. Außerdem erfahren Sie, wie Sie das richtige AMI für Ihren Zweck auswählen und welche Art von Instances Sie bevorzugen könnten.

Darüber hinaus DLAMIs enthalten sie mehrere Tutorials, die die von ihnen unterstützten Frameworks bereitstellen. In diesem Handbuch erfahren Sie, wie Sie die einzelnen Frameworks aktivieren, und finden die entsprechenden Tutorials für den Einstieg. Es enthält auch Tutorials zu verteiltem Training, Debugging, Verwendung von AWS Inferentia und AWS Trainium und anderen Schlüsselkonzepten. Anweisungen zum Einrichten eines Jupyter-Notebookservers für die Ausführung der Tutorials in Ihrem Browser finden Sie unter. Einrichtung eines Jupyter Notebook-Servers auf einer DLAMI-Instanz

Voraussetzungen

Um das erfolgreich auszuführen, empfehlen wir DLAMIs, dass Sie mit Befehlszeilentools und grundlegendem Python vertraut sind.

Beispiele für DLAMI-Anwendungsfälle

Im Folgenden finden Sie Beispiele für einige gängige Anwendungsfälle für AWS Deep Learning AMIs (DLAMI).

Lernen Sie mehr über Deep Learning — DLAMI ist eine hervorragende Wahl für das Lernen oder Unterrichten von Frameworks für maschinelles Lernen und Deep Learning. DLAMIs Sie ersparen Ihnen die Kopfschmerzen bei der Fehlerbehebung bei den Installationen der einzelnen

Frameworks und bringen sie dazu, auf demselben Computer mitzuspielen. Sie DLAMIs enthalten ein Jupyter-Notizbuch und erleichtern die Ausführung der Tutorials, die die Frameworks für Personen bereitstellen, die mit maschinellem Lernen und Deep Learning noch nicht vertraut sind.

App-Entwicklung — Wenn Sie ein App-Entwickler sind, der daran interessiert ist, Deep Learning zu nutzen, damit Ihre Apps die neuesten Fortschritte in der KI nutzen, dann ist DLAMI die perfekte Testumgebung für Sie. Jedes Framework enthält Tutorials zum Einstieg in Deep-Learning. Viele bieten Modellszenarien, mit denen Deep-Learning einfach ausprobiert werden kann, ohne selbst neuronale Netze erstellen zu müssen oder eines der Modelltrainings durchzuführen. Einige Beispiele zeigen Ihnen, wie Sie eine Bilderkennungsanwendung in wenigen Minuten erstellen können oder wie Sie eine Spracherkennungsanwendung für Ihren eigenen Chatbot erstellen können.

Maschinelles Lernen und Datenanalyse — Wenn Sie Datenwissenschaftler sind oder daran interessiert sind, Ihre Daten mit Deep Learning zu verarbeiten, werden Sie feststellen, dass viele der Frameworks R und Spark unterstützen. Sie umfassen Tutorials von einfachen Regressionen bis hin zum Aufbau skalierbarer Datenverarbeitungssysteme für Personalisierungs- und Prognosesysteme.

Forschung — Wenn Sie ein Forscher sind, der ein neues Framework ausprobieren, ein neues Modell testen oder neue Modelle trainieren möchte, dann können DLAMI und AWS Capabilities for Scale die mühsamen Installationen und die Verwaltung mehrerer Trainingsknoten lindern.

> **Note**
>
> Ihre erste Wahl könnte darin bestehen, Ihren Instanztyp auf eine größere Instanz mit mehr GPUs (bis zu 8) aufzurüsten, aber Sie können auch horizontal skalieren, indem Sie einen Cluster von DLAMI-Instanzen erstellen. Weitere Informationen zu Cluster-Builds finden Sie in den entsprechenden Weitere Informationen über DLAMI.

Eigenschaften von DLAMI

Zu den Funktionen von AWS Deep Learning AMIs (DLAMI) gehören vorinstallierte Deep-Learning-Frameworks, GPU-Software, Modellserver und Tools zur Modellvisualisierung.

Vorinstallierte Frameworks

Derzeit gibt es zwei Hauptvarianten von DLAMI mit anderen Varianten, die sich auf das Betriebssystem (OS) und die Softwareversionen beziehen:

- Deep-Learning-AMI mit Conda— Frameworks werden separat mithilfe von `conda` Paketen und separaten Python-Umgebungen installiert.
- Deep-Learning-Basis-AMI— Keine Frameworks installiert; nur NVIDIA CUDA und andere Abhängigkeiten.

Das Deep Learning-AMI mit Conda verwendet `conda` Umgebungen, um jedes Framework zu isolieren, sodass Sie nach Belieben zwischen ihnen wechseln können, ohne sich Sorgen machen zu müssen, dass ihre Abhängigkeiten miteinander in Konflikt geraten. Das Deep Learning AMI mit Conda unterstützt die folgenden Frameworks:

- PyTorch
- TensorFlow 2

> **Note**
>
> DLAMI unterstützt die folgenden Deep-Learning-Frameworks nicht mehr: Apache MXNet, Microsoft Cognitive Toolkit (CNTK), Caffe, Caffe2, Theano, Chainer und Keras.

Vorinstallierte GPU-Software

Selbst wenn Sie eine reine CPU-Instance verwenden, verfügt DLAMIs diese über NVIDIA CUDA und NVIDIA cuDNN. Die installierte Software ist unabhängig vom Instance-Typ identisch. Beachten Sie, dass GPU-spezifische Tools nur auf einer Instanz funktionieren, die über mindestens eine GPU verfügt. Weitere Informationen zu Instanztypen finden Sie unter. Auswahl eines DLAMI-Instanztyps

Weitere Informationen zu CUDA finden Sie unterCUDA-Installationen und Framework-Bindungen.

Bereitstellung und Visualisierung von Modellen

Deep Learning AMI mit Conda ist mit Modellservern sowohl für Modellvisualisierungen als auch TensorBoard für TensorFlow Modellvisualisierungen vorinstalliert. Weitere Informationen finden Sie unter TensorFlow Servieren.

AMIs Versionshinweise zu Deep Learning

Hier finden Sie detaillierte Versionshinweise für alle derzeit unterstützten AWS Deep Learning AMIs (DLAMI) Optionen.

Versionshinweise für DLAMI-Frameworks, die wir nicht mehr unterstützen, finden Sie im Abschnitt Unsupported Framework Release Notes Archive auf der Seite DLAMI Framework Support Policy.

> **Note**
>
> Sie AWS Deep Learning AMIs haben einen nächtlichen Veröffentlichungsrhythmus für Sicherheitspatches. Wir nehmen diese inkrementellen Sicherheitspatches nicht in die Versionshinweise auf.

Versionshinweise

- P6 Unterstützt DLAMI
- Versionshinweise für Base DLAMIs
- Versionshinweise für Single Framework DLAMIs
- Versionshinweise für Multi-Framework DLAMIs

P6 Unterstützt DLAMI

Im Folgenden finden Sie die detaillierten Anforderungen für die Ausführung von DLAMI auf Amazon EC2 P6-Instances

P6 wird unterstützt DLAMIs

Die folgenden DLAMI unterstützen P6-Instanzen:

- AWS Deep-Learning-Basis-AMI (Amazon Linux 2023)
- AWS Deep-Learning-Basis-AMI (Ubuntu 24.04)
- AWS Deep-Learning-Basis-AMI (Ubuntu 22.04)

Diese DLAMI enthalten die folgende Software, die für den Betrieb von P6-B200-Instanzen erforderlich ist:

Software	Mindestanforderung an die Version
Nvidia CUDA-Toolkit	12.8
Nvidia-Treiber	R570
NVLINK 5	R570
Linux-Kernel	6.1
Elastic Fabric Adapter (EFA)	1.41.0
AWS OFI NCCL-Plugin	1.15.0

Bestätigen Sie die GPU-Funktionalität

Um die Funktionsfähigkeit zu bestätigen GPUs:

1. Führen Sie den folgenden Nvidia-GPU-Geräteabfragetest aus

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

2. Bestätigen Sie die folgende Ausgabe des Device Query Run:

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

 CUDA Device Query (Runtime API)

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

Um zu überprüfen, ob der NVIDIA-Treiber funktionsfähig ist:

1. Führen Sie die Nvidia-Systemverwaltungsschnittstelle aus

```
$ nvidia-smi
```

2. Bestätigen Sie die folgende Ausgabe von der Systemverwaltungsschnittstelle

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

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

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

Wenn Sie Probleme mit P6-B200-Instances haben, wenden Sie sich bitte an den Support. AWS

Versionshinweise für Base DLAMIs

X86 Base DLAMI Versionshinweise

- X86 Base DLAMI Versionshinweise
- ARM64 Basis-DLAMI-Versionshinweise

X86 Base DLAMI Versionshinweise

Im Folgenden finden Sie die Versionshinweise für X86 Base DLAMI:

GPU

- AWS Deep Learning Base AMI (Amazon Linux 2023) (unterstützt P6-B200)
- AWS Deep Learning Base AMI (Ubuntu 24.04) (unterstützt P6-B200)
- AWS Deep Learning Base AMI (Ubuntu 22.04) (unterstützt P6-B200)
- AWS Deep-Learning-Basis-AMI (Amazon Linux 2)

Qualcomm

- AWS Deep-Learning-Basis Qualcomm AMI (Amazon Linux 2)

AWS Neuron

- Weitere Informationen finden Sie im Neuron DLAMI-Benutzerhandbuch.

AWS Deep-Learning-Base-GPU-AMI (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning Base OSS Nvidia-Treiber-GPU-AMI (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en, P6-B200

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: x86
- Die neueste verfügbare Version ist für die folgenden Pakete installiert:
  - Linux-Kernel: 6.1
  - FSx Glanz
  - NVIDIA GDS
  - Docker
  - AWS CLI v2 bei/usr/local/bin/aws2 und AWS CLI v1 bei/usr/bin/aws
  - NVIDIA DCGM
  - Nvidia-Container-Toolkit:
    - Versionsbefehl: -V nvidia-container-cli
  - NVIDIA-Docker2:
    - Versionsbefehl: nvidia-docker version
- NVIDIA-Treiber: 570.133.20

- NVIDIA CUDA 12.4-12.6- und 12.8-Stapel:
  - Installationsverzeichnisse für CUDA, NCCL und cuDDN:/-xx.x/ usr/local/cuda
    - Beispiel:/-12.8/ usr/local/cuda-12.8/ , /usr/local/cuda
  - Kompilierte NCCL-Version: 2.26.5
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.4/targets/x86_64-linux/lib
      - PATH soll//haben usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- EFA-Installationsprogramm: 1.40.0
- Nvidia GDRCopy: 2.5
- AWS OFI NCCL: 1.14.2-aws
  - AWS OFI NCCL unterstützt jetzt mehrere NCCL-Versionen mit einem einzigen Build
  - Der Installationspfad:/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/libwurde zu LD_LIBRARY_PATH hinzugefügt.
- AWS CLI v2 bei//2 und v1 bei/usr/local/bin/aws AWS CLI usr/bin/aws
- EBS-Volumetyp: gp3
- Python:/usr/bin/python3.9
- NVMe Speicherort des Instanzspeichers (bei unterstützten EC2 Instanzen):/opt/dlami/nvme
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

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

Hinweise

NVIDIA-Container-Toolkit 1.17.4

In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Unterstützungspolitik

Diese AMIs Komponenten dieses AMI, wie CUDA-Versionen, können auf der Grundlage von Framework-Supportrichtlinien oder zur Optimierung der Leistung für Deep-Learning-Container oder zur Reduzierung der AMI-Größe in einer future Version ohne vorherige Ankündigung entfernt und geändert werden. Wir entfernen CUDA-Versionen, AMIs wenn sie nicht von einer unterstützten Framework-Version verwendet werden.

P6-B200-Instanzen

P6-B200-Instances enthalten 8 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

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

P5en-Instanzen

P5en-Instances enthalten 16 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

P5/P5e-Instanzen

P5- und P5e-Instances enthalten 32 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

```
sudo dnf versionlock kernel*
```

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-05-15

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250515

Hinzugefügt

- Unterstützung für P6-B200-Instances hinzugefügt EC2

Aktualisiert

- Der EFA Installer wurde von Version 1.38.1 auf 1.40.0 aktualisiert

- GDRCopy Von Version 2.4 auf 2.5 aktualisiert
- Das AWS OFI NCCL Plugin wurde von Version 1.13.0-aws auf 1.14.2-aws aktualisiert
- Die kompilierte NCCL-Version wurde von Version 2.25.1 auf 2.26.5 aktualisiert
- Die Standard-CUDA-Version wurde von Version 12.6 auf 12.8 aktualisiert
- Die Nvidia DCGM-Version wurde von 3.3.9 auf 4.4.3 aktualisiert

Datum der Veröffentlichung: 2025-04-22

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250421

Aktualisiert

- Der Nvidia-Treiber wurde von Version 570.124.06 auf 570.133.20 aktualisiert und entspricht nun der Adresse, die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom April 2025 CVEs enthalten ist

Datum der Veröffentlichung: 31.03.2025

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250328

Hinzugefügt

- Unterstützung für NVIDIA GPU Direct Storage (GDS) wurde hinzugefügt

Veröffentlichungsdatum: 2025-02-17

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250215

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Veröffentlichungsdatum: 05.02.2025

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250205

Hinzugefügt

- CUDA-Toolkit Version 12.6 wurde im Verzeichnis/-12.6 hinzugefügt usr/local/cuda
- Unterstützung für G5-Instances hinzugefügt EC2

Entfernt

- Die CUDA-Versionen 12.1 und 12.2 wurden aus diesem DLAMI entfernt. Kunden, die diese CUDA-Toolkit-Versionen benötigen, können sie über den folgenden Link direkt von NVIDIA installieren
  - https://developer.nvidia.com/cuda-toolkit-archive

Veröffentlichungsdatum: 2025-02-03

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250131

Aktualisiert

- Die EFA-Version wurde von 1.37.0 auf 1.38.0 aktualisiert
  - EFA bündelt jetzt das AWS OFI-NCCL-Plugin, das sich jetzt in/-ofi-nccl/ befindet. opt/amazon/ofi-nccl rather than the original /opt/aws Wenn Sie Ihre Variable LD_LIBRARY_PATH aktualisieren, stellen Sie bitte sicher, dass Sie Ihren OFI-NCCL-Speicherort korrekt ändern.
- Das Nvidia Container Toolkit wurde von 1.17.3 auf 1.17.4 aktualisiert

Datum der Veröffentlichung: 2025-01-08

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250107

Aktualisiert

- Unterstützung für G4dn-Instances hinzugefügt

Veröffentlichungsdatum: 2024-12-09

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20241206

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.17.0 auf 1.17.3 aktualisiert

Veröffentlichungsdatum: 2024-11-21

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-GPU-AMI (Amazon Linux 2023) 20241121

Hinzugefügt

- Unterstützung für P5en-Instances hinzugefügt. EC2

Aktualisiert

- Der EFA Installer wurde von Version 1.35.0 auf 1.37.0 aktualisiert
- Aktualisieren Sie das AWS OFI NCCL Plugin von Version 1.121-aws auf 1.13.0-aws

Datum der Veröffentlichung: 2024-10-30

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20241030

Hinzugefügt

- Erste Version des Deep Learning Base OSS DLAMI für Amazon Linux 2023

Bekannte Probleme

- Dieses DLAMI unterstützt derzeit keine G4dn- und EC2 G5-Instances. AWS ist sich einer Inkompatibilität bewusst, die zu CUDA-Initialisierungsfehlern führen kann, die sich sowohl auf die G4dn- als auch auf die G5-Instance-Familien auswirken, wenn die Open-Source-NVIDIA-Treiber zusammen mit einer Linux-Kernel-Version 6.1 oder neuer verwendet werden. Dieses Problem

betrifft unter anderem Linux-Distributionen wie Amazon Linux 2023, Ubuntu 22.04 oder neuer oder SUSE Linux Enterprise Server 15 SP6 oder neuer.

AWS Deep-Learning-Base-GPU-AMI (Ubuntu 24.04)

Hilfe zu den ersten Schritten finden Sie unter. Erste Schritte mit DLAMI

AMI-Namensformat

- Deep Learning Base OSS Nvidia-Treiber-GPU-AMI (Ubuntu 24.04) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en, P6-B200.

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 24.04
- Rechenarchitektur: x86
- Die neueste verfügbare Version ist für die folgenden Pakete installiert:
  - Linux-Kernel: 6. 8
  - FSx Glanz
  - Docker
  - AWS CLI v2 bei/usr/bin/aws
  - NVIDIA DCGM
  - Nvidia-Container-Toolkit:
    - Versionsbefehl: -V nvidia-container-cli
  - NVIDIA-Docker2:
    - Versionsbefehl: nvidia-docker version
- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.6- und 12.8-Stapel:

  - Installationsverzeichnisse für CUDA, NCCL und cuDDN:/-xx.x/ usr/local/cuda
    - Beispiel:/-12.8/ usr/local/cuda-12.8/ , /usr/local/cuda
  - Kompilierte NCCL-Version: 2.25.1
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/sbsa-linux/lib:/usr/local/cuda-12.8/nvvm/lib64:/usr/local/cuda-12.8/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- EFA-Installationsprogramm: 1.40.0
- Nvidia GDRCopy: 2.5.1
- AWS OFI NCCL: 1.14.2-aws
  - Installationspfad:/wird zu LD_LIBRARY_PATH hinzugefügt. opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/lib
- AWS CLI v2 bei/usr/bin/aws
- EBS-Volumetyp: gp3
- Python:/usr/bin/python3.12
- NVMe Speicherort des Instanzspeichers (bei unterstützten EC2 Instanzen):/opt/dlami/nvme
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

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

Hinweise

Unterstützungspolitik

Diese AMIs Komponenten dieses AMI, wie CUDA-Versionen, können auf der Grundlage von Framework-Supportrichtlinien oder zur Optimierung der Leistung für Deep-Learning-Container oder zur Reduzierung der AMI-Größe in einer future Version ohne vorherige Ankündigung entfernt und geändert werden. Wir entfernen CUDA-Versionen, AMIs wenn sie nicht von einer unterstützten Framework-Version verwendet werden.

EC2 Instanz mit mehreren Netzwerkkarten

- Viele Instance-Typen, die EFA unterstützen, verfügen auch über mehrere Netzwerkkarten.
- DeviceIndex ist für jede Netzwerkkarte eindeutig und muss eine nicht negative Ganzzahl sein, die unter dem Grenzwert von ENIs per NetworkCard liegt. Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind.
  - Erstellen Sie für die primäre Netzwerkschnittstelle (Netzwerkkartenindex 0, Geräteindex 0) eine EFA-Schnittstelle (EFA mit ENA). Sie können eine Nur-EFA-Netzwerkschnittstelle nicht als primäre Netzwerkschnittstelle verwenden.
  - Verwenden Sie für jede weitere Netzwerkschnittstelle den nächsten ungenutzten Netzwerkkartenindex, Geräteindex 1, und entweder eine EFA (EFA mit ENA) oder eine reine EFA-Netzwerkschnittstelle, je nach Ihrem Anwendungsfall, z. B. den ENA-Bandbreitenanforderungen oder dem IP-Adressraum. Anwendungsfälle finden Sie beispielsweise unter EFA-Konfiguration für P5-Instances.
  - Weitere Informationen finden Sie im EFA-Leitfaden hier.

P6-B200-Instanzen

P6-B200-Instances enthalten 8 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

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

P5en-Instanzen

P5en enthalten 16 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

P5/P5e-Instanzen

P5- und P5e-Instances enthalten 32 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

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

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es handelt sich um einen Sicherheitspatch), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-05-22

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 24.04) 20250522

Hinzugefügt

- Unterstützung für P6-B200-Instanzen hinzugefügt EC2

Aktualisiert

- Der EFA Installer wurde von Version 1.40.0 auf 1.41.0 aktualisiert
- Die kompilierte NCCL-Version wurde von Version 2.25.1 auf 2.26.5 aktualisiert
- Die Nvidia DCGM-Version wurde von 3.3.9 auf 4.4.3 aktualisiert

Datum der Veröffentlichung: 2025-05-13

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 24.04) 20250513

Hinzugefügt

- Erste Veröffentlichung des Deep Learning Base OSS DLAMI für Ubuntu 24.04

AWS Deep-Learning-Base-GPU-AMI (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning Base OSS Nvidia-Treiber-GPU-AMI (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P6-B200.

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: x86

- Die neueste verfügbare Version ist für die folgenden Pakete installiert:
  - Linux-Kernel: 6. 8
  - FSx Glanz
  - Docker
  - AWS CLI v2 bei/usr/local/bin/aws2 und AWS CLI v1 bei/usr/bin/aws
  - NVIDIA DCGM
  - Nvidia-Container-Toolkit:
    - Versionsbefehl: -V nvidia-container-cli
  - NVIDIA-Docker2:
    - Versionsbefehl: nvidia-docker version
- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.4-12.6- und 12.8-Stapel:
  - Installationsverzeichnisse für CUDA, NCCL und cuDDN:/-xx.x/ usr/local/cuda
    - Beispiel:/-12.8/ usr/local/cuda-12.8/ , /usr/local/cuda
  - Kompilierte NCCL-Version: 2.26.5
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/x86_64-linux/lib:/usr/local/cuda-12.8/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- EFA-Installationsprogramm: 1.40.0
- Nvidia GDRCopy: 2.5
- AWS OFI NCCL: 1.14.2-aws
  - Installationspfad:/wird zu LD_LIBRARY_PATH hinzugefügt. opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/lib
- AWS CLI v2 bei//2 und v1 bei/usr/local/bin/aws AWS CLI usr/bin/aws
- EBS-Volumetyp: gp3
- Python:/usr/bin/python3.10

- NVMe Speicherort des Instanzspeichers (bei unterstützten EC2 Instanzen):/opt/dlami/nvme
- Fragen Sie die AMI-ID mit dem SSM-Parameter ab (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Hinweise

NVIDIA-Container-Toolkit 1.17.4

In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

EFA-Updates von 1.37 auf 1.38 (Veröffentlichung am 31.01.2025)

EFA bündelt jetzt das AWS OFI-NCCL-Plugin, das jetzt in/-ofi-nccl/ zu finden ist. opt/amazon/ofi-nccl rather than the original /opt/aws Wenn Sie Ihre Variable LD_LIBRARY_PATH aktualisieren, stellen Sie bitte sicher, dass Sie Ihren OFI-NCCL-Speicherort korrekt ändern.

Unterstützung mehrerer ENI

- Ubuntu 22.04 richtet das Quell-Routing automatisch auf mehreren ein und konfiguriert es NICss mithilfe von Cloud-Init beim ersten Start. Wenn Ihr Arbeitsablauf den Vorgang beinhaltet attaching/

detaching , ENIs während eine Instanz gestoppt ist, muss den Cloud-Init-Benutzerdaten eine zusätzliche Konfiguration hinzugefügt werden, um sicherzustellen, dass die NICs während dieser Ereignisse ordnungsgemäß konfiguriert werden. Ein Beispiel für die Cloud-Konfiguration finden Sie unten.

- Weitere Informationen zur Konfiguration der Cloud-Konfiguration für Ihre Instanzen finden Sie in dieser Canonical-Dokumentation - -/https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automaticallysetup-multiple-nics

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

Unterstützungspolitik

Diese AMIs Komponenten dieses AMI, wie CUDA-Versionen, können auf der Grundlage von Framework-Supportrichtlinien oder zur Optimierung der Leistung für Deep-Learning-Container oder zur Reduzierung der AMI-Größe in einer future Version ohne vorherige Ankündigung entfernt und geändert werden. Wir entfernen CUDA-Versionen, AMIs wenn sie nicht von einer unterstützten Framework-Version verwendet werden.

EC2 Instanzen mit mehreren Netzwerkkarten

- Viele Instance-Typen, die EFA unterstützen, verfügen auch über mehrere Netzwerkkarten.
- DeviceIndex ist für jede Netzwerkkarte eindeutig und muss eine nicht negative Ganzzahl sein, die unter dem Grenzwert von ENIs per NetworkCard liegt. Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind.
  - Erstellen Sie für die primäre Netzwerkschnittstelle (Netzwerkkartenindex 0, Geräteindex 0) eine EFA-Schnittstelle (EFA mit ENA). Sie können eine Nur-EFA-Netzwerkschnittstelle nicht als primäre Netzwerkschnittstelle verwenden.
  - Verwenden Sie für jede weitere Netzwerkschnittstelle den nächsten ungenutzten Netzwerkkartenindex, Geräteindex 1, und entweder eine EFA (EFA mit ENA) oder eine reine EFA-Netzwerkschnittstelle, je nach Ihrem Anwendungsfall, z. B. den ENA-Bandbreitenanforderungen oder dem IP-Adressraum. Anwendungsfälle finden Sie beispielsweise unter EFA-Konfiguration für P5-Instances.
  - Weitere Informationen finden Sie im EFA-Leitfaden hier.

P6-B200-Instanzen

P6-B200 enthalten 8 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

P5en-Instanzen

P5en enthalten 16 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

P5/P5e-Instanzen

P5- und P5e-Instances enthalten 32 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

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

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-05-16

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250516

Hinzugefügt

- Unterstützung für P6-B200-Instanzen hinzugefügt EC2

Aktualisiert

- Der EFA Installer wurde von Version 1.39.0 auf 1.40.0 aktualisiert
- Aktualisieren Sie das AWS OFI NCCL Plugin von Version 1.13.0-aws auf 1.14.2-aws
- Die kompilierte NCCL-Version wurde von Version 2.22.3 auf 2.26.5 aktualisiert
- Die Standard-CUDA-Version wurde von Version 12.6 auf 12.8 aktualisiert
- Die Nvidia DCGM-Version wurde von 3.3.9 auf 4.4.3 aktualisiert

Datum der Veröffentlichung: 2025-05-05

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250503

Aktualisiert

- GDRCopy Von 2.4.1 auf 2.5.1 aktualisiert

Datum der Veröffentlichung: 24.04.2025

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250424

Aktualisiert

- Der Nvidia-Treiber wurde von Version 570.124.06 auf 570.133.20 aktualisiert und entspricht nun der Adresse CVEs , die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom April 2025 enthalten ist

Veröffentlichungsdatum: 17.02.2025

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250214

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren,

dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Veröffentlichungsdatum: 2025-02-07

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250205

Hinzugefügt

- CUDA-Toolkit Version 12.6 wurde im Verzeichnis/-12.6 hinzugefügt usr/local/cuda

Entfernt

- Die CUDA-Versionen 12.1 und 12.2 wurden aus diesem DLAMI entfernt. Kunden können diese Versionen von NVIDIA über den folgenden Link installieren
  - https://developer.nvidia.com/cuda-toolkit-archive

Veröffentlichungsdatum: 2025-01-31

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250131

Aktualisiert

- Die EFA-Version wurde von 1.37.0 auf 1.38.0 aktualisiert
  - EFA bündelt jetzt das AWS OFI-NCCL-Plugin, das sich jetzt in/-ofi-nccl/ befindet. opt/amazon/ofi-nccl rather than the original /opt/aws Wenn Sie Ihre Variable LD_LIBRARY_PATH aktualisieren, stellen Sie bitte sicher, dass Sie Ihren OFI-NCCL-Speicherort korrekt ändern.
- Das Nvidia Container Toolkit wurde von 1.17.3 auf 1.17.4 aktualisiert

Datum der Veröffentlichung: 2025-01-17

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250117

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert und entspricht nun der Adresse CVEs , die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom Januar 2025 enthalten ist

Veröffentlichungsdatum: 18.11.2024

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20241115

Hinzugefügt

- FSx Amazon-Paket für Lustre-Unterstützung hinzugefügt.

Fixed

- Aufgrund einer Änderung im Ubuntu-Kernel zur Behebung eines Fehlers in der KASLR-Funktionalität (Kernel Address Space Layout Randomization) können G4Dn/G5-Instances CUDA auf dem OSS-Nvidia-Treiber nicht ordnungsgemäß initialisieren. Um dieses Problem zu beheben, enthält dieses DLAMI Funktionen, die den proprietären Treiber für G4Dn- und G5-Instances dynamisch laden. Bitte rechnen Sie mit einer kurzen Initialisierungszeit für diesen Ladevorgang, um sicherzustellen, dass Ihre Instanzen ordnungsgemäß funktionieren.

  Um den Status und den Zustand dieses Dienstes zu überprüfen, können Sie den folgenden Befehl verwenden:

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

Datum der Veröffentlichung: 2024-10-23

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20241023

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.90.07 auf 550.127.05 aktualisiert und entspricht nun der Adresse, die im NVIDIA GPU Display Security Bulletin für Oktober 2024 enthalten ist CVEs

Veröffentlichungsdatum: 2024-10-01

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 20.04) 20240930

Aktualisiert

- Der Nvidia-Treiber und der Fabric Manager wurden von Version 535.183.01 auf 550.90.07 aktualisiert
- Das Nvidia Container Toolkit wurde von Version 1.16.1 auf 1.16.2 aktualisiert und die Sicherheitslücke CVE-2024-0133 behoben.
- Die EFA-Version wurde von 1.32.0 auf 1.34.0 aktualisiert
- NCCL wurde für alle CUDA-Versionen auf die neueste Version 2.22.3 aktualisiert
  - CUDA 12.1, 12.2 wurde von 2.18.5+ 2 aktualisiert CUDA12
  - CUDA 12.3 wurde von Version 2.21.5+ aktualisiert. CUDA12

Hinzugefügt

- CUDA-Toolkit-Version 12.4 wurde im Verzeichnis/-12.4 hinzugefügt usr/local/cuda
- Unterstützung für P5e-Instanzen hinzugefügt. EC2

Veröffentlichungsdatum: 2024-08-19

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20240816

Hinzugefügt

- Unterstützung für die G6e-Instanz hinzugefügt. EC2

Veröffentlichungsdatum: 2024-06-06

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20240606

Aktualisiert

- Die Nvidia-Treiberversion wurde von 535.161.08 auf 535.183.01 aktualisiert

Datum der Veröffentlichung: 2024-05-15

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20240513

Entfernt

- Die Unterstützung von Amazon FSx for Lustre wurde in dieser Version aufgrund von Inkompatibilität mit den neuesten Ubuntu 22.04-Kernelversionen entfernt. Die Support FSx für Lustre wird wieder aktiviert, sobald die neueste Kernelversion unterstützt wird. Kunden, die Lustre benötigen FSx , sollten weiterhin das Deep Learning Base GPU AMI (Ubuntu 20.04) verwenden.

Datum der Veröffentlichung: 2024-04-29

AMI-Name: Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20240429

Hinzugefügt

- Erste Veröffentlichung des Deep Learning Base OSS DLAMI für Ubuntu 22.04

AWS Deep-Learning-Basis-AMI (Amazon Linux 2)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version $ {XX.X}
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2), Version $ {XX.X}

Unterstützte Instanzen EC2

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en
- Deep Learning mit proprietärem Nvidia-Treiber unterstützt G3 (G3.16x nicht unterstützt), P3, P3dn

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2
- Rechenarchitektur: x86
- Die neueste verfügbare Version ist für die folgenden Pakete installiert:

  - Linux-Kernel: 5.10
  - Docker
  - AWS CLI v2 bei/usr/local/bin/aws2 und AWS CLI v1 bei/usr/bin/aws
  - Nvidia-Container-Toolkit:
    - Versionsbefehl: -V nvidia-container-cli
  - NVIDIA-Docker2:
    - Versionsbefehl: nvidia-docker version
- Python:/usr/bin/python3.7
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.163.01
  - Proprietärer Nvidia-Treiber: 550.163.01
- NVIDIA CUDA 12.1-12.4-Stapel:
  - Installationsverzeichnisse für CUDA, NCCL und cuDDN:/-xx.x/ usr/local/cuda
  - Standard-CUDA: 12.1
    - PATH//usr/local/cudazeigt auf CUDA 12.1
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1:/usr/local/cuda-12.1/targets/x86_64-linux/lib
      - PATH soll//haben usr/local/cuda-12.1/bin/:/usr/local/cuda-12.1/include
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
  - Kompilierte NCCL-Version: 2.22.3
  - Ort der NCCL-Tests:
    - all_reduce, all_gather und reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
    - Um NCCL-Tests ausführen zu können, muss LD_LIBRARY_PATH mit den folgenden Aktualisierungen bestanden werden.
      - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs
        - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- EFA-Installationsprogramm: 1.38.0
- Nvidia: 2,4 GDRCopy
- AWS OFI NCCL: 1.13.2

  - AWS OFI NCCL unterstützt jetzt mehrere NCCL-Versionen mit einem einzigen Build
  - Installationspfad:/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/lib64 wurde zu LD_LIBRARY_PATH hinzugefügt.
- EBS-Volumetyp: gp3
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

  - Eigener Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

```
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
```

  - Eigener Nvidia-Treiber:

```
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
```

Hinweise

NVIDIA-Container-Toolkit 1.17.4

In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

EFA-Updates von 1.37 auf 1.38 (Veröffentlichung am 04.02.2025)

EFA bündelt jetzt das AWS OFI-NCCL-Plugin, das sich jetzt in/-ofi-nccl/ befindet. opt/amazon/ofi-nccl rather than the original /opt/aws Wenn Sie Ihre Variable LD_LIBRARY_PATH aktualisieren, stellen Sie bitte sicher, dass Sie Ihren OFI-NCCL-Speicherort korrekt ändern.

Unterstützungspolitik

Diese AMIs Komponenten dieses AMI, wie CUDA-Versionen, können auf der Grundlage von Framework-Supportrichtlinien oder zur Optimierung der Leistung für Deep-Learning-Container oder zur Reduzierung der AMI-Größe in einer future Version ohne vorherige Ankündigung entfernt und geändert werden. Wir entfernen CUDA-Versionen, AMIs wenn sie nicht von einer unterstützten Framework-Version verwendet werden.

EC2 Instanzen mit mehreren Netzwerkkarten

- Viele Instance-Typen, die EFA unterstützen, verfügen auch über mehrere Netzwerkkarten.
- DeviceIndex ist für jede Netzwerkkarte eindeutig und muss eine nicht negative Ganzzahl sein, die unter dem Grenzwert von ENIs per NetworkCard liegt. Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind.
  - Erstellen Sie für die primäre Netzwerkschnittstelle (Netzwerkkartenindex 0, Geräteindex 0) eine EFA-Schnittstelle (EFA mit ENA). Sie können eine Nur-EFA-Netzwerkschnittstelle nicht als primäre Netzwerkschnittstelle verwenden.
  - Verwenden Sie für jede weitere Netzwerkschnittstelle den nächsten ungenutzten Netzwerkkartenindex, Geräteindex 1, und entweder eine EFA (EFA mit ENA) oder eine reine EFA-Netzwerkschnittstelle, je nach Ihrem Anwendungsfall, z. B. den ENA-Bandbreitenanforderungen oder dem IP-Adressraum. Anwendungsfälle finden Sie beispielsweise unter EFA-Konfiguration für P5-Instances.
  - Weitere Informationen finden Sie im EFA-Leitfaden hier.

P5/P5e-Instanzen

- P5- und P5e-Instances enthalten 32 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

P5en-Instanzen

- P5en enthalten 16 Netzwerkschnittstellenkarten und können mit dem folgenden Befehl gestartet werden: AWS CLI

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

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

```
sudo yum versionlock kernel*
```

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-04-22

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 69.3
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 67.0

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.144.03 auf 550.163.01 aktualisiert, um die im Sicherheitsbulletin für NVIDIA-GPU-Bildschirmtreiber vom CVEs April 2025 enthaltene Adresse zu ersetzen

Veröffentlichungsdatum: 2025-02-17

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 68.5

- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 66.3

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert. Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. CVEs Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt

Veröffentlichungsdatum: 2025-02-04

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 68.4
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 66.1

Aktualisiert

- Die EFA-Version wurde von 1.37.0 auf 1.38.0 aktualisiert

Veröffentlichungsdatum: 2025-01-17

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 68.3
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 66.0

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert, um die im Sicherheitsbulletin für NVIDIA-GPU-Bildschirmtreiber vom CVEs Januar 2025 enthaltene Adresse zu ersetzen

Veröffentlichungsdatum: 2025-01-06

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 68.2
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 65.9

Aktualisiert

- EFA wurde von Version 1.34.0 auf 1.37.0 aktualisiert
- AWS OFI NCCL wurde von Version 1.11.0 auf 1.13.0 aktualisiert

Veröffentlichungsdatum: 2024-12-09

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 68.1
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 65.8

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.17.0 auf 1.17.3 aktualisiert

Veröffentlichungsdatum: 2024-11-09

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 67.9
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 65.6

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.16.2 auf 1.17.0 aktualisiert und die Sicherheitslücke CVE-2024-0134 behoben.

Datum der Veröffentlichung: 22.10.2024

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 67.7
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 65.4

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.90.07 auf 550.127.05 aktualisiert und entspricht nun der Adresse, die im NVIDIA GPU-Display-Sicherheitsbulletin für CVEs Oktober 2024 enthalten ist

Veröffentlichungsdatum: 2024-10-03

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI-Version (Amazon Linux 2)
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 65.2

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.16.1 auf 1.16.2 aktualisiert und die Sicherheitslücke CVE-2024-0133 behoben.

Veröffentlichungsdatum: 2024-08-27

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 67.0

Aktualisiert

- Der Nvidia-Treiber und der Fabric Manager wurden von Version 535.183.01 auf 550.90.07 aktualisiert
  - Aufgrund der Empfehlungen von Nvidia wurde die Anforderung einer Mehrbenutzer-Shell aus Fabric Manager entfernt
  - Weitere Informationen finden Sie hier unter Bekannte Probleme für den Tesla-Treiber 550.90.07
- Die EFA-Version wurde von 1.32.0 auf 1.34.0 aktualisiert
- NCCL wurde für alle CUDA-Versionen auf die neueste Version 2.22.3 aktualisiert

- CUDA 12.1, 12.2 wurde von 2.18.5+ 2 aktualisiert CUDA12
- CUDA 12.3 wurde von 2.21.5+ aktualisiert. CUDA12

Hinzugefügt

- CUDA-Toolkit-Version 12.4 wurde im Verzeichnis/-12.4 hinzugefügt usr/local/cuda
- Unterstützung für P5e-Instanzen hinzugefügt. EC2

Entfernt

- Der CUDA Toolkit-Stack der Version 11.8 wurde entfernt, der im Verzeichnis/-11.8 vorhanden ist usr/local/cuda

Veröffentlichungsdatum: 2024-08-19

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 66.3

Hinzugefügt

- Unterstützung für EC2 G6e-Instances hinzugefügt.

Veröffentlichungsdatum: 2024-06-06

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 65.4
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 63.9

Aktualisiert

- Die Nvidia-Treiberversion wurde von 535.161.08 auf 535.183.01 aktualisiert

Datum der Veröffentlichung: 2024-05-02

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 64.7

- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 63.2

Aktualisiert

- Die EFA-Version wurde von Version 1.30 auf Version 1.32 aktualisiert
- Das AWS OFI NCCL-Plugin wurde von Version 1.7.4 auf Version 1.9.1 aktualisiert
- Das Nvidia-Container-Toolkit wurde von Version 1.13.5 auf Version 1.15.0 aktualisiert

Hinzugefügt

- CUDA123.3-Stack mit CUDA12 .3, NCCL 2.21.5, cuDNN 8.9.7 hinzugefügt

  Version 1.15.0 enthält NICHT die Pakete und nvidia-docker2. nvidia-container-runtime Es wird empfohlen, nvidia-container-toolkit Pakete direkt zu verwenden, indem Sie den Dokumenten zum Nvidia-Container-Toolkit folgen.

Entfernt

- CUDA11.7, CUDA12 .0-Stapel entfernt, die bei/-12.0 vorhanden waren usr/local/cuda-11.7 and / usr/local/cuda
- Das nvidia-docker2-Paket und sein Befehl nvidia-docker wurden als Teil des Nvidia-Container-Toolkit-Updates von 1.13.5 auf 1.15.0 entfernt, das NICHT die Pakete und nvidia-docker2 enthält. nvidia-container-runtime

Veröffentlichungsdatum: 2024-04-04

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 64.0

Hinzugefügt

- Für den OSS-Nvidia-Treiber wurde DLAMIs Unterstützung für G6- und EC2 Gr6-Instances hinzugefügt

Veröffentlichungsdatum: 2024-03-29

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 62.3

- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 63.2

Aktualisiert

- Der Nvidia-Treiber wurde sowohl im proprietären als auch im OSS-Nvidia-Treiber von 535.104.12 auf 535.161.08 aktualisiert. DLAMIs
- Die neuen unterstützten Instanzen für jedes DLAMI lauten wie folgt:
  - Deep Learning mit proprietärem Nvidia-Treiber unterstützt G3 (G3.16x nicht unterstützt), P3, P3dn
  - Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, P4d, P4de, P5.

Entfernt

- Die Unterstützung für G4dn-, G5- und EC2 G3.16x-Instanzen wurde aus dem proprietären Nvidia-Treiber DLAMI entfernt.

Datum der Veröffentlichung: 20.03.2024

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 63.1

Hinzugefügt

- awscliv2 wurde im AMI als//usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awsauf dem OSS Nvidia Driver AMI hinzugefügt

Datum der Veröffentlichung: 2024-03-13

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 63.0

Aktualisiert

- Aktualisierter OSS Nvidia-Treiber DLAMI mit G4dn- und G5-Unterstützung, basierend darauf sieht die aktuelle Unterstützung wie folgt aus:
  - Das proprietäre Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) unterstützt P3, P3dn, G3, G4dn, G5.
  - Das Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) unterstützt G4dn, G5, P4, P5.

- DLAMIs Es wird empfohlen, die OSS-Nvidia-Treiber für G4dn, G5, P4, P5 zu verwenden.

Veröffentlichungsdatum: 2024-02-13

AMI-Namen

- Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 62.1
- Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 62.1

Aktualisiert

- Der OSS Nvidia-Treiber wurde von 535.129.03 auf 535.154.05 aktualisiert
- EFA wurde von 1.29.0 auf 1.30.0 aktualisiert
- AWS OFI NCCL wurde von 1.7.3-aws auf 1.7.4-aws aktualisiert

Datum der Veröffentlichung: 2024-02-01

AMI-Name: Proprietäres Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) Version 62.0

Sicherheit

- Die Runc-Paketversion wurde aktualisiert, um den Patch für CVE-2024-21626 zu verwenden.

Version 6.1.4

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 61.4

Aktualisiert

- Der OSS-Nvidia-Treiber wurde von 535.104.12 auf 535.129.03 aktualisiert

Version 61.0

AMI-Name: Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 61.4

Aktualisiert

- EFA wurde von 1.26.1 auf 1.29.0 aktualisiert
- GDRCopy aktualisiert von 2.3 auf 2.4

Hinzugefügt

- AWS Deep Learning AMI (DLAMI) ist in zwei separate Gruppen aufgeteilt:
  - DLAMI, das den proprietären Treiber von Nvidia verwendet (zur Unterstützung von P3, P3dn, G3, G5, G4dn).
  - DLAMI, das den Nvidia OSS-Treiber verwendet, um EFA zu aktivieren (zur Unterstützung von P4, P5).
- Weitere Informationen zu DLAMI Split finden Sie in der öffentlichen Ankündigung.
- AWS CLI Abfragen finden Sie unter dem Aufzählungspunkt Abfrage AMI-ID mit AWSCLI (Beispiel Region ist us-east-1)

Version 60.6

AMI-Name: Deep Learning Base-AMI (Amazon Linux 2) Version 60.6

Aktualisiert

- AWS Das OFI NCCL Plugin wurde von Version 1.7.2 auf Version 1.7.3 aktualisiert
- Die CUDA 12.0-12.1-Verzeichnisse wurden mit der NCCL-Version 2.18.5 aktualisiert
- CUDA12.1 wurde als Standard-CUDA-Version aktualisiert
  - LD_LIBRARY_PATH wurde auf//aktualisiert usr/local/cuda-12.1/targets/x86_64-linux/lib/:/usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1 and PATH to have /usr/local/cuda-12.1/bin
  - Für Kunden, die zu einer anderen CUDA-Version wechseln möchten, definieren Sie bitte die Variablen LD_LIBRARY_PATH und PATH entsprechend.

Hinzugefügt

- Kernel Live Patching ist jetzt aktiviert. Live-Patching ermöglicht es Kunden, Sicherheitslücken und kritische Bug-Patches auf einen laufenden Linux-Kernel anzuwenden, ohne Neustarts oder Unterbrechungen laufender Anwendungen. Bitte beachten Sie, dass die Live-Patching-Unterstützung für Kernel 5.10.192 am 30.11.23 endet.

Version 6.0.5

AMI-Name: Deep Learning Base-AMI (Amazon Linux 2) Version 60.5

Aktualisiert

- Der NVIDIA-Treiber wurde von 535.54.03 auf 535.104.12 aktualisiert

  Dieser neueste Treiber behebt wichtige NVML-ABI-Änderungen im 535.54.03-Treiber sowie die Treiberregression im Treiber 535.86.10, die CUDA-Toolkits auf P5-Instances betraf. Einzelheiten zu den Problembehebungen finden Sie in den folgenden NVIDIA-Versionshinweisen:
  - 4235941 — Behebung einer wichtigen Änderung in NVML ABI
  - 4228552 — CUDA Toolkit-Fehler behoben
- CUDA 12.2-Verzeichnisse mit NCCL 2.18.5 aktualisiert
- EFA wurde von 1.24.1 auf die neueste Version 1.26.1 aktualisiert

Hinzugefügt

- 2.2 bei//-12.2 hinzugefügt CUDA12 usr/local/cuda

Entfernt

- Die Unterstützung für CUDA 11.5 und CUDA 11.6 wurde entfernt

Version 60.2

AMI-Name: Deep Learning Base-AMI (Amazon Linux 2) Version 60.2

Aktualisiert

- aws-ofi-ncclDas Plugin wurde von v1.7.1 auf v1.7.2 aktualisiert

Version 60.0

Veröffentlichungsdatum: 2023-08-11

Hinzugefügt

- Dieses AMI bietet jetzt Unterstützung für Trainingsfunktionen mit mehreren Knoten auf P5 und allen zuvor unterstützten Instances EC2
- Für EC2 P5-Instances wird die Verwendung von NCCL 2.18 empfohlen. Es wurde zu Version 2.0 und .1 hinzugefügt. CUDA12 CUDA12

Entfernt

- Die Unterstützung für .5 wurde entfernt. CUDA11

Version 5.9.2

Veröffentlichungsdatum: 2023-08-08

Entfernt

- CUDA-11.3 und CUDA-11.4 wurden entfernt

Version 59.1

Veröffentlichungsdatum: 2023-08-03

Aktualisiert

- Das AWS OFI NCCL-Plugin wurde auf v1.7.1 aktualisiert
- Made CUDA11 .8 als Standard, da PyTorch 2.0 11.8 unterstützt und für EC2 P5-Instances wird empfohlen, >= .8 zu verwenden. CUDA11
  - LD_LIBRARY_PATH wurde auf//aktualisiert usr/local/cuda-11.8/targets/x86_64-linux/lib/:/usr/local/cuda-11.8/lib:/usr/local/cuda-11.8/lib64:/usr/local/cuda-11.8 and PATH to have /usr/local/cuda-11.8/bin
  - Für jede andere Cuda-Version definieren Sie LD_LIBRARY_PATH bitte entsprechend.

Fixed

- Das in der früheren Version 2023-07-19 erwähnte Problem beim Laden von Nvidia Fabric Manager (FM) -Paketen wurde behoben.

Version 58.9

Veröffentlichungsdatum: 2023-07-19

Aktualisiert

- Der Nvidia-Treiber wurde von 525.85.12 auf 535.54.03 aktualisiert
- Das EFA-Installationsprogramm wurde von 1.22.1 auf 1.24.1 aktualisiert

Hinzugefügt

- Es wurden C-State-Änderungen hinzugefügt, um den Leerlaufstatus des Prozessors zu deaktivieren, indem der maximale C-Status auf C1 gesetzt wurde. Diese Änderung wird vorgenommen, indem `intel_idle.max_cstate=1 processor.max_cstate=1` in den Linux-Boot-Argumenten in der Datei/gesetzt wird etc/default/grub
- AWS EC2 Unterstützung für P5-Instanzen:
  - EC2 P5-Instanzunterstützung für Workflows hinzugefügt, die einen einzelnen Knoten/eine einzelne Instanz verwenden. Unterstützung mehrerer Knoten (z. B. für Schulungen mit mehreren Knoten) mithilfe von EFA (Elastic Fabric Adapter) und dem AWS OFI NCCL-Plugin wird in einer kommenden Version hinzugefügt.
  - Bitte verwenden Sie CUDA>=11.8 für eine optimale Leistung.
  - Bekanntes Problem: Das Laden des Nvidia Fabric Manager (FM) -Pakets auf P5 dauert einige Zeit. Kunden müssen nach dem Start der P5-Instance 2-3 Minuten warten, bis FM geladen wird. Um zu überprüfen, ob FM gestartet wurde, führen Sie bitte den Befehl sudo systemctl is-active nvidia-fabricmanager aus. Er sollte wieder aktiv sein, bevor Sie einen Workflow starten. Dies wird in der kommenden Version behoben.

Version 58.0

Veröffentlichungsdatum: 2023-05-19

Entfernt

- Der Stapel CUDA11 .0-11.2 wurde gemäß der im oberen Abschnitt dieses Dokuments genannten Support-Richtlinie entfernt.

Version 5.7.3

Veröffentlichungsdatum: 2023-04-06

Hinzugefügt

- Nvidia 2.3 hinzugefügt GDRCopy

Version 56.8

Veröffentlichungsdatum: 2023-03-09

Aktualisiert

- Der NVIDIA-Treiber wurde von 515.65.01 auf 525.85.12 aktualisiert

Hinzugefügt

- usr/local/cudacuda-11.8 bei/-11.8/ hinzugefügt

Ausführung 56.0

Veröffentlichungsdatum: 2022-06

Aktualisiert

- Die EFA-Version wurde von 1.17.2 auf 1.19.0 aktualisiert

Version 55.0

Veröffentlichungsdatum: 04.11.2022

Aktualisiert

- Der NVIDIA-Treiber wurde von 510.47.03 auf 515.65.01 aktualisiert

Hinzugefügt

- usr/local/cudacuda-11.7 unter/-11.7/ hinzugefügt

Ausführung 54.0

Veröffentlichungsdatum: 2015-09-15

Aktualisiert

- Die EFA-Version wurde von 1.16.0 auf 1.17.2 aktualisiert

Version 53.3

Veröffentlichungsdatum: 25.05.2022

Aktualisiert

- Auf Version aws-efa-installer 1.15.2 aktualisiert
- Auf Version 1.3.0-aws aktualisiert aws-ofi-nccl, die die Topologie für p4de.24xlarge enthält.

Hinzugefügt

- Diese Version bietet Unterstützung für p4de.24xlarge-Instances. EC2

Version 53.0

Veröffentlichungsdatum: 28.04.2022

Hinzugefügt

- CloudWatch Amazon-Agent hinzugefügt
- Es wurden drei systemd-Dienste hinzugefügt, die vordefinierte JSON-Dateien verwenden, die unter Pfadopt/aws/amazon-cloudwatch-agent/etc//verfügbar sind, um GPU-Metriken mithilfe des Linux-Benutzers cwagent zu konfigurieren
  - dlami-cloudwatch-agent@minimal
    - Befehle zum Aktivieren von GPU-Metriken:

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

    - Es erstellt diese Metriken:`utilization_gpu`, `utilization_memory`
  - dlami-cloudwatch-agent@partial
    - Befehle zum Aktivieren von GPU-Metriken:

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

    - Es erstellt diese Metriken:`utilization_gpu`,`utilization_memory`,`memory_total`,`memory_used`, `memory_free`
  - dlami-cloudwatch-agent@all
    - Befehle zum Aktivieren von GPU-Metriken:

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

- Es erstellt alle verfügbaren GPU-Metriken

Version 52.0

Veröffentlichungsdatum: 08.03.2022

Aktualisiert

- Kernel-Version auf 5.10 aktualisiert

Version 51.0

Veröffentlichungsdatum: 04.03.2022

Aktualisiert

- Der Nvidia-Treiber wurde auf 510.47.03 aktualisiert

Version 50.0

Veröffentlichungsdatum: 17.02.2022

Aktualisiert

- Gesperrt aws-neuron-dkms und tensorflow-model-server-neuron sobald sie auf neuere Versionen aktualisiert werden, die von den in AMI vorhandenen Neuron-Paketen nicht unterstützt werden
  - Befehle, falls der Kunde das Paket entsperren möchte, um es auf die neueste Version zu aktualisieren: sudo yum versionlock delete sudo yum versionlock delete aws-neuron-dkms tensorflow-model-server-neuron

Version 49.0

Veröffentlichungsdatum: 13.01.2022

Hinzugefügt

- CUDA112.2 mit den folgenden Komponenten hinzugefügt:

- cuDNN v8.1.1.33
- NCCL 2.8.4
- CUDA 11.2.2

Aktualisiert

- Symlink Pip wurde auf Pip3 aktualisiert

Veraltungen

- Veraltete Unterstützung für den Instanztyp P2
- Python2.7 wurde verworfen und verwandte Python2.7-Pakete wie „python-dev“, „python-pip“ und „python-tk“ entfernt

Version 48.0

Veröffentlichungsdatum: 2021-12-27

Aktualisiert

- org.apache.ant_1.9.2.v201404171502\ lib\ ant-apache-log 4j.jar wurde aus den Cuda-Versionen entfernt, da es nicht verwendet wird und kein Risiko für Benutzer mit den Log4j-Dateien besteht. Weitere Informationen finden Sie unter https://nvidia.custhelp.com/app/answers/detail/a_id/5294.

Version 47.0

Veröffentlichungsdatum: 2021-11-24

Aktualisiert

- EFA wurde auf 1.14.1 aktualisiert

Version 46.0

Veröffentlichungsdatum: 2021-11-12

Aktualisiert

- Neuron-Pakete wurden von =1.5 aktualisiert. aws-neuron-dkms *, =1.5aws-neuron-runtime-base. *, aws-neuron-tools =1.6.* bis =2.2. aws-neuron-dkms *, =1,6. aws-neuron-runtime-base *, aws-neuron-tools =2,0*.
- Das Neuron-Paket aws-neuron-runtime =1.5.* wurde entfernt, da bei Neuron keine Runtime mehr als Daemon läuft und Runtime jetzt als Bibliothek in das Framework integriert ist.

Version 45.0

Veröffentlichungsdatum: 2021-10-21

Hinzugefügt

- Sicherheitsscan-Berichte im JSON-Format sind unter//verfügbar. opt/aws/dlami/info

Version 44.0

Veröffentlichungsdatum: 2021-10-08

der Änderung

- Für jeden Instance-Start mit DLAMI wird das Tag "aws-dlami-autogenerated-tag-do-not-delete" hinzugefügt, das es ermöglicht, Instance-Typ, Instance-ID, DLAMI-Typ und Betriebssysteminformationen AWS zu sammeln. Es werden keine Informationen zu den in der DLAMI verwendeten Befehlen gesammelt oder gespeichert. Es werden keine weiteren Informationen über das DLAMI gesammelt oder gespeichert. Um die Nutzungsverfolgung für Ihr DLAMI zu deaktivieren, fügen Sie Ihrer EC2 Amazon-Instance beim Start ein Tag hinzu. Das Tag sollte den Schlüssel OPT_OUT_TRACKING verwenden, wobei der zugehörige Wert auf true gesetzt ist. Weitere Informationen finden Sie unter Taggen Sie Ihre EC2 Amazon-Ressourcen.

Sicherheit

- Docker-Version auf Docker-20.10.7-3 aktualisiert

Version 43.0

Veröffentlichungsdatum: 2021-08-24

der Änderung

- „Notebook“ auf Version „6.4.1" aktualisiert.

Version 4.2.0

Veröffentlichungsdatum: 2021-07-23

der Änderung

- Die Version des Nvidia-Treibers und des Fabric Managers wurde auf 450.142.00 aktualisiert.

Version 41.0

Veröffentlichungsdatum: 2021-06-24

der Änderung

- Aktualisierte Neuron-Pakete gemäß Neuron Release v1.14.0

Version 40.0

Veröffentlichungsdatum: 2021-06-10

der Änderung

- Die awscli-Version wurde auf 1.19.89 aktualisiert

Version 39.0

Veröffentlichungsdatum: 2021-05-27

Sicherheit

- Die anfälligen CUDA-10.0-Komponenten (Visual Profiler, Nsight EE und JRE) wurden aus der CUDA-10.0-Installation (/-10.0) entfernt. usr/local/cuda

Version 38.0

Veröffentlichungsdatum: 2021-05-25

der Änderung

- Runc auf den neuesten Stand gebracht

Version 37.0

Veröffentlichungsdatum: 2021-04-23

der Änderung

- Die Version des Nvidia Tesla-Treibers und des Fabric Managers wurde auf 450.119.03 aktualisiert.

Version 36.1

Veröffentlichungsdatum: 2021-04-21

Fixed

- Es wurde ein Problem behoben, das die Startgeschwindigkeit der Instance verlangsamte.

Version 36.0

Veröffentlichungsdatum: 2021-03-24

Hinzugefügt

- tensorflow-model-server-neuronZur Unterstützung der Bereitstellung von Neuronenmodellen hinzugefügt.

der Änderung

- Jupyterlab wurde auf Version 3.0.8 für Python3 aktualisiert.

Fixed

- Die alte Installation von OpenMPI in/usr/local/mpi caused /opt/amazon/openmpi/bin/mpirun to be linked incorrectly. To fix the link issue, we removed /usr/local/mpi installation, OpenMPI installation in /opt/amazon/openmpiist verfügbar.

- Entfernt doppelte und nicht existierende Definitionen von Shell-Umgebungen, die die Shell-Umgebungsvariablen wie PATH und LD_LIBRARY_PATH verschmutzt haben. Als Ergebnis wurden ~/.dlami und/.sh hinzugefügt. etc/profile.d/var.sh has been removed, and /etc/profile.d/dlami

Sicherheit

- Das Paket Cryptography wurde auf die Adresse CVE-2020-36242 aktualisiert

Version 35.0

Veröffentlichungsdatum: 2021-03-08

Hinzugefügt

- TensorRT CUDA 11.0-Installation hinzugefügt

Version 34.3

Veröffentlichungsdatum: 25.02.2021

Fixed

- Es wurde ein Tippfehler in der MOTD (Message of the Day) behoben, durch den Version 34.1 fälschlicherweise angezeigt wurde.

Version 34.2

Veröffentlichungsdatum: 2021-02-24

Sicherheit

- Python2 und Python3 für CVE-2021-3177 gepatcht

Bekanntes Problem

- Es gibt einen Tippfehler in der MOTD (Nachricht des Tages), durch den Version 34.1 falsch angezeigt wurde. Wir werden Version 34.3 veröffentlichen, um dieses Problem zu beheben.

Version 34.0

Veröffentlichungsdatum: 2021-02-09

der Änderung

- Pip wurde für Python2 an Version 20.3.4 angeheftet. Dies ist die letzte Pip-Version, die Python2 und Python3.5 unterstützt.

Version 33.0

Veröffentlichungsdatum: 2021-01-19

der Änderung

- Die cuDNN-Version wurde auf Version 8.0.5.39 in Version 2.0 und 8.1 aktualisiert. CUDA11 CUDA11

Version 3.2.0

Veröffentlichungsdatum: 2020-12-01

Hinzugefügt

- CUDA11.1 mit NCCL 2.7.8, cuDNN 8.0.4.30 für Deep Learning AMI (Amazon Linux 2), Deep Learning AMI (Ubuntu 16.04), Deep Learning AMI (Ubuntu 18.04), Deep Learning Base AMI (Ubuntu 16.04), Deep Learning Base AMI (Ubuntu 18.04), Deep Learning Base AMI (Amazon Linux 2) hinzugefügt.

Version 3.1.0

Veröffentlichungsdatum: 2020-11-02

der Änderung

- Das EFA-Installationsprogramm wurde auf Version 1.10.0 aktualisiert.
- Die cuDNN-Version wurde auf v8.0.4.30 für CUDA 11.0 aktualisiert.
- AWS Neuron wurde auf Version 1.1 aktualisiert

Version 30.0

Veröffentlichungsdatum: 2020-10-08

der Änderung

- Die NVIDIA-Treiber- und Fabric Manager-Versionen wurden auf 450.80.02 aktualisiert
- NCCL wurde für 2.0 auf 2.7.8 in aktualisiert CUDA11

Fixed

- Es wurde ein Problem behoben, bei dem Yum ein Python-Paket verwaltete, das von pipmanagten Installationen überschrieben wurde. Die ausführbaren Dateien pip, pip3 und pip3.7 wurden aus dem /-Teil dieses Fixes verschoben. usr/binto /usr/local/binas

Version 29.0

Veröffentlichungsdatum: 2020-09-11

der Änderung

- Der NVIDIA-Treiber wurde von Version 450.51.05 auf 450.51.06 aktualisiert
- NVIDIA Fabric Manager Version 450.51.06 wurde hinzugefügt
- EFA wurde auf 1.9.4 aktualisiert

Version 28.0

Veröffentlichungsdatum: 2020-08-19

der Änderung

- CUDA 11.0-Stack mit NCCL 2.7.6 und cuDNN 8.0.2.39 hinzugefügt

Version 27.0

Veröffentlichungsdatum: 2020-08-07

der Änderung

- EFA wurde von Version 1.7.1 auf 1.9.3 aktualisiert unter/opt/amazon/efa

- Das Upgrade von Open MPI von Version 4.0.3 auf 4.0.4 in '/' usr/local/mpi'. Open MPI at '/opt/amazon/openmpi/bin/mpirun ist immer noch auf Version 4.0.3
- Der NVIDIA-Treiber wurde von 440.33.01 auf 450.51.05 aktualisiert
- Die NCCL-Version wurde in 0.2 von 2.6.4 auf 2.7.6 aktualisiert CUDA1

Version 26.0

Veröffentlichungsdatum: 2020-08-03

der Änderung

- AWS OFI NCCL wurde auf den neuesten Stand gebracht. Weitere Informationen finden Sie hier.
- Cuda 8.0/9.0/9.2 wurden aus dem AMI entfernt

Fixed

- Es wurde ein Fehler behoben, bei dem die gemeinsam genutzte Objektdatei: libopencv_dnn.so.4.2 nicht geöffnet werden konnte.

Version 25.0

Veröffentlichungsdatum: 2020-07-19

der Änderung

- Die EFA-Version wurde auf 1.7.1 aktualisiert, um NCCL 2.6.4 zu unterstützen
- Die NCCL-Version wurde für CUDA 10.2 auf 2.6.4 aktualisiert
- Die awscli-Version wurde von 1.16.76 auf 1.18.80 aktualisiert
- Die boto3-Version wurde von 1.9.72 auf 1.14.3 aktualisiert

Version 24.1

Veröffentlichungsdatum: 2020-06-14

der Änderung

- Docker-Version auf 19.03.6 aktualisiert

Version 24.0

Veröffentlichungsdatum: 2020-05-20

der Änderung

- Docker-Version auf 19.03.6 aktualisiert

Version 23.0

Veröffentlichungsdatum: 2020-04-29

der Änderung

- Aktualisierte Python-Paketversionen

Version 22.0

Veröffentlichungsdatum: 2020-03-04

der Änderung

- CUDA 10.2-Stack hinzugefügt
- CUDA 10.0 und 10.1 für cuDNN- und NCCL-Version aktualisiert

AWS Deep-Learning-Basis Qualcomm AMI (Amazon Linux 2)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep-Learning-Basis Qualcomm AMI (Amazon Linux 2) $ {YYYY-MM-DD}

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2
- Rechenarchitektur: x86

- Linux-Kernel: 5.10.210-201.852.amzn2.x86_64
- SDK-Speicherort: /opt/qai-sdk
- Speicherort der QTI-Utils: /opt/qti-aic
- Plattform-SDK-Version: 1.12.0.88
- SDK-Version für Anwendungen: 1.12.0.87
- EBS-Volumetyp: gp3
- Python: Python3.8
- Unterstützte EC2 Instanzen: dl2q
- AMI-ID abfragen mit AWS CLI (Beispiel Region ist us-east-1):

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

Hinweise

[4/12/24] Entfernung des Audit-Pakets

- DLAMIs Die zwischen dem 26. März 2024 (2024-03-26) und dem 12. April 2024 (2024-04-12) veröffentlichten Artikel wurden ohne das Auditpaket ausgeliefert. Wenn Sie dieses spezielle Paket für Ihre Protokollierungs- und Überwachungsanforderungen benötigen, migrieren Sie Ihre Workflows bitte auf die neueste DLAMI-Version, um sie mit dem installierten Audit-Paket nutzen zu können.

QAIC-Umgebung:

- Standardmäßig werden die qaic-pytools nicht über das Qualcomm AI1 00 Apps SDK aktiviert. Um die QAIC-PIP-Umgebung qaic-env zu aktivieren und zu erstellen, führen Sie bitte die folgenden Befehle aus:

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

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

Version 20240314

Veröffentlichungsdatum: 2024-03-14

AMI-Name: Deep Learning Base Qualcomm AMI (Amazon Linux 2) 20240314

Hinzugefügt

- Das AI 100 Platform SDK wurde von Version 1.10.0.200 auf Version 1.12.0.88 aktualisiert
- AI 100 Apps SDK wurde von Version 1.10.0.193 auf Version 1.12.0.87 aktualisiert
- Die SDK-Version 1.12 bietet Unterstützung für Transformer-Decoder-Modelle wie Llama-2 und Starcoder

Version 20240.110

Datum der Veröffentlichung: 2024-01-10

AMI-Name: Deep Learning Base Qualcomm AMI (Amazon Linux 2) 20240103

Hinzugefügt

- Das Image der Qualcomm AI 100-Plattform wurde auf 1.10.0.200 aktualisiert

Version 20231115

Datum der Veröffentlichung: 2023-11-15

AMI-Name: Deep Learning Base Qualcomm AMI (Amazon Linux 2) 20231115

Hinzugefügt

- Erste Veröffentlichung der Deep Learning Base Qualcomm AMI-Serie (Amazon Linux 2).
  - Weitere Informationen zur Plattform und zum Apps-SDK finden Sie in der offiziellen Qualcomm-Dokumentation: https://quic.github.io//cloud-ai-sdk-pages
  - Weitere Informationen zu dl2q-Instances: Instances finden Sie in der offiziellen AWS Dokumentation. DL2q

ARM64 Basis-DLAMI-Versionshinweise

- ARM64 Basis-DLAMI-Versionshinweise

ARM64 Basis-DLAMI-Versionshinweise

Im Folgenden finden Sie die Versionshinweise für ARM64 Base DLAMI:

GPU

- AWS ARM64 Deep-Learning-Basis-AMI (Amazon Linux 2023)
- AWS ARM64 Deep-Learning-Basis-AMI (Ubuntu 22.04)
- AWS ARM64 Deep-Learning-Basis-AMI (Amazon Linux 2)

AWS Neuron

- Weitere Informationen finden Sie im Neuron DLAMI-Benutzerhandbuch.

AWS ARM64 Deep-Learning-Base-GPU-AMI (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning ARM64 Base OSS Nvidia-Treiber-GPU-AMI (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: ARM64
- Linux-Kernel: 6.12

- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.4, 12.5, 12.6, 12.8 Stapel:
  - CUDA-, NCCL - und cuDDN-Installationsverzeichnisse:/-xx.x/ usr/local/cuda
    - Beispiel:/-12.8/ usr/local/cuda-12.8/ , /usr/local/cuda
  - Kompilierte NCCL-Version:
    - Für das CUDA-Verzeichnis von 12.4, kompilierte NCCL-Version 2.22.3+ .4 CUDA12
    - Für das CUDA-Verzeichnis 12.5, kompilierte NCCL-Version 2.22.3+ .5 CUDA12
    - Für das CUDA-Verzeichnis von 12.6, kompilierte NCCL-Version 2.24.3+ .6 CUDA12
    - Für das CUDA-Verzeichnis von 12.8, kompilierte NCCL-Version 2.26.2+ .8 CUDA12
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/sbsa-linux/lib:/usr/local/cuda-12.8/nvvm/lib64:/usr/local/cuda-12.8/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- AWS CLI v2 bei/usr/local/bin/aws
- EBS-Volumetyp: gp3
- Nvidia-Container-Toolkit: 1.17.4
  - Versionsbefehl: -V nvidia-container-cli
- Docker: 25.0.5
- Python:/usr/bin/python3.9
- AMI-ID mit SSM-Parameter abfragen (Beispielregion ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispielregion ist us-east-1):

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters
'Name=name,Values=Deep Learning ARM64 Base OSS Nvidia Driver GPU AMI (Amazon Linux
```

```
2023) ????????' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
&CreationDate))[:1].ImageId' --output text
```

Hinweise

NVIDIA-Container-Toolkit 1.17.4

In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Unterstützungspolitik

Diese AMIs Komponenten dieses AMI, wie CUDA-Versionen, können auf der Grundlage von Framework-Supportrichtlinien oder zur Optimierung der Leistung für Deep-Learning-Container oder zur Reduzierung der AMI-Größe in einer future Version ohne vorherige Ankündigung entfernt und geändert werden. Wir entfernen CUDA-Versionen, AMIs wenn sie nicht von einer unterstützten Framework-Version verwendet werden.

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

```
sudo dnf versionlock kernel*
```

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-04-24

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250424

Aktualisiert

- Der Nvidia-Treiber wurde von Version 570.86.15 auf 570.133.20 aktualisiert und entspricht nun der Adresse, die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom April 2025 CVEs enthalten ist
- CUDA12Der 3.8-Stack wurde mit NCCL 2.26.2 aktualisiert
- Standard-CUDA wurde von 12.6 auf 12.8 aktualisiert

Datum der Veröffentlichung: 2025-04-22

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU-AMI (Amazon Linux 2023) 20250421

Aktualisiert

- Der Nvidia-Treiber wurde von Version 570.124.06 auf 570.133.20 aktualisiert und entspricht nun der Adresse, die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom April 2025 CVEs enthalten ist

Datum der Veröffentlichung: 04.04.2025

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2023) 20250404

Aktualisiert

- Die Kernel-Version wurde von 6.1 auf 6.12 aktualisiert

Datum der Veröffentlichung: 2025-03-03

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2023) 20250303

Aktualisiert

- Nvidia-Treiber von 550.144.03 bis 570.86.15
- Der Standard-CUDA wurde von 1.4 auf 6 geändert. CUDA12 CUDA12

Hinzugefügt

- CUDA-Verzeichnis von 12.5 mit kompilierter NCCL-Version CUDA12 2.22.3+ .5 und cuDNN 9.7.1.26
- CUDA-Verzeichnis von 12.6 mit kompilierter NCCL-Version CUDA12 2.24.3+ .6 und cuDNN 9.7.1.26
- CUDA-Verzeichnis von 12.8 mit kompilierter NCCL-Version CUDA12 2.25.1+ .8 und cuDNN 9.7.1.26

Veröffentlichungsdatum: 2025-02-14

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber-GPU-AMI (Amazon Linux 2023) 20250214

Hinzugefügt

- Erste Version des Deep Learning ARM64 Base OSS DLAMI für Amazon Linux 2023

AWS ARM64 Deep-Learning-Base-GPU-AMI (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning ARM64 Base OSS Nvidia-Treiber-GPU-AMI (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2

- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: ARM64
- Linux-Kernel: 6.8.0-1027-aws
- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.4, 12.5, 12.6, 12.8 Stapel:
  - CUDA-, NCCL - und cuDDN-Installationsverzeichnisse:/-xx.x/ usr/local/cuda
    - Beispiel:/-12.8/ usr/local/cuda-12.8/ , /usr/local/cuda
  - Kompilierte NCCL-Version:
    - Für das CUDA-Verzeichnis von 12.4, kompilierte NCCL-Version 2.22.3+ .4 CUDA12
    - Für das CUDA-Verzeichnis 12.5, kompilierte NCCL-Version 2.22.3+ .5 CUDA12
    - Für das CUDA-Verzeichnis von 12.6, kompilierte NCCL-Version 2.24.3+ .6 CUDA12
    - Für das CUDA-Verzeichnis von 12.8, kompilierte NCCL-Version 2.26.2+ .8 CUDA12
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda-12.8/lib:/usr/local/cuda-12.8/lib64:/usr/local/cuda-12.8:/usr/local/cuda-12.8/targets/sbsa-linux/lib:/usr/local/cuda-12.8/nvvm/lib64:/usr/local/cuda-12.8/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda-12.8/bin/:/usr/local/cuda-12.8/include
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- AWS CLI v2 bei//2 und v1 bei/usr/local/bin/aws AWS CLI usr/bin/aws
- EBS-Volumetyp: gp3
- Nvidia-Container-Toolkit: 1.17.4
  - Versionsbefehl: -V nvidia-container-cli
- NVIDIA DCGM: 3.3
  - Versionsbefehl dcgmi -v
- Docker: 26.1.2
- Python:/usr/bin/python3.10
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):

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

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):

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

Hinweise

NVIDIA-Container-Toolkit 1.17.4

In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Unterstützung mehrerer ENI

- Ubuntu 22.04 richtet beim ersten Start automatisch das Quell-Routing auf mehreren NICs über Cloud-Init ein und konfiguriert es. Wenn Ihr Workflow attaching/detaching Ihre ENIs beinhaltet, während eine Instanz gestoppt ist, muss den Cloud-Init-Benutzerdaten eine zusätzliche Konfiguration hinzugefügt werden, um sicherzustellen, dass die NICs während dieser Ereignisse ordnungsgemäß konfiguriert werden. Ein Beispiel für die Cloud-Konfiguration finden Sie unten.
- Weitere Informationen zur Konfiguration der Cloud-Konfiguration für Ihre Instanzen finden Sie in dieser Canonical-Dokumentation - -/https://documentation.ubuntu.com/aws/en/latest/aws-how-to/instances/automaticallysetup-multiple-nics

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

Unterstützungspolitik

Diese AMIs Komponenten dieses AMI, wie CUDA-Versionen, können auf der Grundlage von Framework-Supportrichtlinien oder zur Optimierung der Leistung für Deep-Learning-Container oder zur Reduzierung der AMI-Größe in einer future Version ohne vorherige Ankündigung entfernt und geändert werden. Wir entfernen CUDA-Versionen, AMIs wenn sie nicht von einer unterstützten Framework-Version verwendet werden.

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

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

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-04-24

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250424

Aktualisiert

- Der Nvidia-Treiber wurde von Version 570.86.15 auf 570.133.20 aktualisiert, um der Tatsache Rechnung zu tragen, dass CVE im NVIDIA GPU Display Driver Security Bulletin für April 2025 auftaucht
- Der CUDA 12.8-Stack wurde mit NCCL 2.26.2 aktualisiert
- Standard-CUDA wurde von 12.6 auf 12.8 aktualisiert
- CUDA 12.3 wurde entfernt

Datum der Veröffentlichung: 2025-03-03

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250303

Aktualisiert

- Nvidia-Treiber von 550.144.03 bis 570.86.15
- Der Standard-CUDA wurde von .1 auf 6 geändert. CUDA12 CUDA12

Hinzugefügt

- CUDA-Verzeichnis von 12.4 mit kompilierter NCCL Version CUDA12 2.22.3+ .4 und cuDNN 9.7.1.26
- CUDA-Verzeichnis von 12.5 mit kompilierter NCCL-Version CUDA12 2.22.3+ .5 und cuDNN 9.7.1.26
- CUDA-Verzeichnis von 12.6 mit kompilierter NCCL-Version CUDA12 2.24.3+ .6 und cuDNN 9.7.1.26
- CUDA-Verzeichnis von 12.8 mit kompilierter NCCL-Version CUDA12 2.25.1+ .8 und cuDNN 9.7.1.26

Entfernt

- CUDA-Verzeichnis von 12.1 und 12.2

Datum der Veröffentlichung: 2025-02-17

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250214

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren,

dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Datum der Veröffentlichung: 17.01.2025

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20250117

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert und entspricht nun der Adresse CVEs , die im Sicherheitsbulletin für NVIDIA-GPU-Bildschirmtreiber vom Januar 2025 enthalten ist

Datum der Veröffentlichung: 2024-10-23

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20241023

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.90.07 auf 550.127.05 aktualisiert und entspricht nun der Adresse, die im NVIDIA GPU Display Security Bulletin für Oktober 2024 enthalten ist CVEs

Datum der Veröffentlichung: 2024-06-06

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20240606

Aktualisiert

- Die Nvidia-Treiberversion wurde von 535.161.08 auf 535.183.01 aktualisiert

Datum der Veröffentlichung: 2024-05-15

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04) 20240514

Hinzugefügt

- Erste Veröffentlichung des Deep Learning ARM64 Base OSS DLAMI für Ubuntu 22.04

AWS ARM64 Deep-Learning-Base-GPU-AMI (Amazon Linux 2)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning ARM64 Base OSS Nvidia-Treiber-GPU-AMI (Amazon Linux 2) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2
- Rechenarchitektur: ARM64
- Linux-Kernel: 5.10
- NVIDIA-Treiber: 550.144.03
- NVIDIA CUDA 12.1, 12.2, 12.3-Stapel:
  - Installationsverzeichnisse CUDA, NCCL und cuDDN:
    - Beispiel:/usr/local/cuda-12.1/ , /usr/local/cuda-12.1/
  - Kompilierte NCCL-Version:
    - Für das CUDA-Verzeichnis von 12.3, kompilierte NCCL-Version 2.21.5+. CUDA12
    - Für das CUDA-Verzeichnis 12.1, 12.2, kompilierte NCCL-Version 1.8.5+ 2. CUDA12
  - Standard-CUDA: 12.1
    - PATH//usr/local/cudazeigt auf CUDA 12.1
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1:/usr/local/cuda-12.1/targets/sbsa-linux/lib:/usr/local/cuda-12.1/nvvm/lib64:/usr/local/cuda-12.1/extras/CUPTI/lib

    - PATH soll//haben usr/local/cuda-12.1/bin/:/usr/local/cuda-12.1/include
    - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- AWS CLI v2 bei//2 und v1 bei/usr/local/bin/aws AWS CLI usr/bin/aws
- EBS-Volumetyp: gp3
- Nvidia-Container-Toolkit: 1.16.2
    - Versionsbefehl: -V nvidia-container-cli
- Docker: 26.1.2
- Python:/usr/bin/python3.10
- Fragen Sie die AMI-ID mit dem SSM-Parameter ab (Beispiel Region ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):

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

Hinweise

NVIDIA-Container-Toolkit 1.17.4

In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Unterstützungspolitik

Diese AMIs Komponenten dieses AMI, wie CUDA-Versionen, können auf der Grundlage von Framework-Supportrichtlinien oder zur Optimierung der Leistung für Deep-Learning-Container oder zur Reduzierung der AMI-Größe in einer future Version ohne vorherige Ankündigung entfernt und geändert werden. Wir entfernen CUDA-Versionen, AMIs wenn sie nicht von einer unterstützten Framework-Version verwendet werden.

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

```
sudo yum versionlock kernel*
```

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-02-17

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2) 20250214

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Datum der Veröffentlichung: 17.01.2025

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2) 20250117

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert und entspricht nun der Adresse, die im Sicherheitsbulletin für NVIDIA-GPU-Bildschirmtreiber vom Januar 2025 CVEs enthalten ist

Datum der Veröffentlichung: 22.10.2024

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2) 20241022

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.90.07 auf 550.127.05 aktualisiert und entspricht nun der Adresse CVEs , die im NVIDIA GPU Display Security Bulletin für Oktober 2024 enthalten ist

Veröffentlichungsdatum: 2024-10-08

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2) 20241008

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.16.1 auf 1.16.2 aktualisiert und die Sicherheitslücke CVE-2024-0133 behoben.

Veröffentlichungsdatum: 2024-06-06

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2) 20240606

Aktualisiert

- Die Nvidia-Treiberversion wurde von 535.161.08 auf 535.183.01 aktualisiert

Datum der Veröffentlichung: 2024-05-14

AMI-Name: Deep Learning ARM64 Base OSS Nvidia-Treiber GPU AMI (Amazon Linux 2) 20240514

Hinzugefügt

- Erste Version des Deep Learning ARM64 Base OSS DLAMI für Amazon Linux 2

Versionshinweise für Single Framework DLAMIs

Single Framework DLAMI Versionshinweise

- PyTorch DLAMIs
- TensorFlow DLAMIs

PyTorch DLAMIs

Versionshinweise zu Multi Framework DLAMI

- X86 PyTorch DLAMI Versionshinweise
- ARM64 PyTorch Versionshinweise zu DLAMI

X86 PyTorch DLAMI Versionshinweise

Im Folgenden finden Sie die Versionshinweise für X86: PyTorch DLAMIs

GPU

- AWS Deep-Learning-AMI-GPU PyTorch 2.7 (Amazon Linux 2023)
- AWS Deep-Learning-AMI-GPU PyTorch 2.7 (Ubuntu 22.04)
- AWS Deep-Learning-AMI-GPU PyTorch 2.6 (Amazon Linux 2023)
- AWS Deep-Learning-AMI-GPU PyTorch 2.6 (Ubuntu 22.04)
- AWS Deep-Learning-AMI-GPU PyTorch 2.5 (Amazon Linux 2023)
- AWS Deep-Learning-AMI-GPU PyTorch 2.5 (Ubuntu 22.04)
- AWS Deep-Learning-AMI-GPU PyTorch 2.4 (Ubuntu 22.04)

AWS Neuron

- Weitere Informationen finden Sie im Neuron DLAMI-Benutzerhandbuch

AWS Deep Learning OSS AMI GPU PyTorch 2.7 (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI
- G4dn, G5, G5, Gr6, P4, P4de, P5, P5e, P5en, P6-B200

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: x86
- Linux-Kernel: 6.1
- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.8-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationsverzeichnisse:/-12.8/ usr/local/cuda
  - Ort der NCCL-Tests:
    - all_reduce, all_gather und reduce_scatter:

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

    - Um NCCL-Tests auszuführen, wurde LD_LIBRARY_PATH bereits mit den benötigten Pfaden aktualisiert.
      - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs

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

      - LD_LIBRARY_PAT wurde mit CUDA-Versionspfaden aktualisiert:

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

  - Kompilierte NCCL-Version:
    - Für das CUDA-Verzeichnis von 12.8, kompilierte NCCL-Version 2.26.2+. CUDA12
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda/targets/x86_64-linux/lib
      - PATH, um//zu haben usr/local/cuda/bin/:/usr/local/cuda/include
- EFA-Installationsprogramm: 1.40.0
- Nvidia GDRCopy: 2.5
- AWS OFI NCCL: 1.14.2-aws
  - Installationspfad:/wird zu LD_LIBRARY_PATH hinzugefügt opt/amazon/ofi-nccl/. Path /opt/amazon/ofi-nccl/lib
- AWS CLI v2 bei/usr/local/bin/aws
- EBS-Volumetyp: gp3
- Nvidia-Container-Toolkit: 1.17.7
  - Versionsbefehl: -V nvidia-container-cli
- Docker: 25.0.8
- Python:/usr/bin/python3.12
- AMI-ID mit SSM-Parameter abfragen (Beispielregion ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispielregion ist us-east-1):

```
aws ec2 describe-images --region us-east-1 --owners amazon --filters
 'Name=name,Values=Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Amazon Linux
```

```
2023) ????????' 'Name=state,Values=available' --query 'reverse(sort_by(Images,
&CreationDate))[:1].ImageId' --output text
```

Hinweise

P6-B200-Instanzen

- P6-B200-Instanzen erfordern CUDA Version 12.8 oder höher und NVIDIA-Treiber 570 oder neuere Treiber.
- P6-B200 enthält 8 Netzwerkschnittstellenkarten und kann mit dem folgenden AWS CLI-Befehl gestartet werden:

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

P5/P5e-Instanzen

- DeviceIndex ist für jede Variable eindeutig NetworkCard und muss eine nicht negative Ganzzahl sein, die unter dem Grenzwert von per liegt. ENIs NetworkCard Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzigen gültigen Werte für 0 oder 1 DeviceIndex sind. Im Folgenden finden Sie ein Beispiel für einen Befehl zum Starten einer EC2 P5-Instanz mithilfe von awscli, der NetworkCardIndex für die Zahlen 0-31 und DeviceIndex als 0 für die erste Schnittstelle und 1 für die verbleibenden 31 Schnittstellen angezeigt wird.

```
aws ec2 run-instances --region $REGION \
```

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

```
sudo dnf versionlock kernel*
```

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es handelt sich um einen Sicherheitspatch), um die Kompatibilität mit den installierten Treibern und Paketversionen zu gewährleisten. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

PyTorch Veraltete Version von Anaconda Channel

Ab Version PyTorch 2.6 wird die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). PyTorch Infolgedessen werden Version PyTorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um das PyTorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

Datum der Veröffentlichung: 2025-05-22

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.7 (Amazon Linux 2023) 20250520

Hinzugefügt

- Erste Version der Deep Learning AMI GPU PyTorch 2.7 (Amazon Linux 2023) -Serie. Einschließlich einer virtuellen Python-Umgebung pytorch (source/opt/pytorch/bin/activate), ergänzt mit NVIDIA-Treiber R570, CUDA=12.8, cuDNN=9.10, NCCL=2.26.2 und EFA=1.40.0. PyTorch

AWS Deep-Learning-OSS-AMI-GPU PyTorch 2.7 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI
- G4dn, G5, G5, Gr6, P4, P4de, P5, P5e, P5en, P6-B200

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: x86
- Linux-Kernel: 6.8
- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.8-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationsverzeichnisse:/-12.8/ usr/local/cuda
  - Ort der NCCL-Tests:
    - all_reduce, all_gather und reduce_scatter:

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

    - Um NCCL-Tests auszuführen, wurde LD_LIBRARY_PATH bereits mit den benötigten Pfaden aktualisiert.
      - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs

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

      - LD_LIBRARY_PAT wurde mit CUDA-Versionspfaden aktualisiert:

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

  - Kompilierte NCCL-Version:
    - Für das CUDA-Verzeichnis von 12.8, kompilierte NCCL-Version 2.26.2+. CUDA12
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda/targets/x86_64-linux/lib
      - PATH, um//zu haben usr/local/cuda/bin/:/usr/local/cuda/include
- EFA-Installationsprogramm: 1.40.0
- Nvidia GDRCopy: 2.5
- Nvidia-Transformer-Engine: 1.11.0
- AWS OFI NCCL: 1.14.2-aws
  - Installationspfad:/wird zu LD_LIBRARY_PATH hinzugefügt opt/amazon/ofi-nccl/. Path /opt/amazon/ofi-nccl/lib
- AWS CLI v2 bei/usr/local/bin/aws
- EBS-Volumetyp: gp3
- Nvidia-Container-Toolkit: 1.17.7
  - Versionsbefehl: -V nvidia-container-cli
- Docker: 28.2.2
- Python:/usr/bin/python3.12
- AMI-ID mit SSM-Parameter abfragen (Beispielregion ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispielregion ist us-east-1):

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

Hinweise

Flash, Achtung

- Flash Attention hat noch keine offizielle Version für PyTorch 2.7. Aus diesem Grund wird es vorübergehend aus diesem AMI entfernt. Sobald eine offizielle Version für PyTorch 2.7 veröffentlicht wurde, werden wir sie in dieses AMI aufnehmen.
- Ohne Flash Attention verwendet die Transformer Engine standardmäßig cuDNN Fused Attention. Derzeit sind Probleme mit Fused Attention und GPUs Blackwell bekannt, z. B. bei P6-B200-Instances.
  - „Bei der Rechenkapazität sm10.0 (Blackwell-Architektur) enthält der FP8 Datentyp mit skalierter Aufmerksamkeit auf Punktprodukte einen Deadlock GPUs, der dazu führt, dass der Kernel unter bestimmten Umständen hängen bleibt, z. B. wenn das Problem groß ist oder wenn auf der GPU mehrere Kernel gleichzeitig ausgeführt werden. Ein Fix ist für eine future Version geplant.“ [CuDNN 9.10.0 Versionshinweise]
  - Benutzer, die P6-B200-Instances mit FP8 Daten und skalierter Aufmerksamkeit auf Punktprodukte ausführen möchten, sollten erwägen, Flash Attention manuell zu installieren.

P6-B200-Instanzen

- P6-B200-Instanzen erfordern CUDA Version 12.8 oder höher und NVIDIA-Treiber 570 oder neuere Treiber.

- P6-B200 enthält 8 Netzwerkschnittstellenkarten und kann mit dem folgenden AWS CLI-Befehl gestartet werden:

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

P5/P5e-Instanzen

- DeviceIndex ist für jede Variable eindeutig NetworkCard und muss eine nicht negative Ganzzahl sein, die unter dem Grenzwert von per liegt. ENIs NetworkCard Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzigen gültigen Werte für 0 oder 1 DeviceIndex sind. Im Folgenden finden Sie ein Beispiel für einen Befehl zum Starten einer EC2 P5-Instanz mithilfe von awscli, der NetworkCardIndex für die Zahlen 0-31 und DeviceIndex als 0 für die erste Schnittstelle und 1 für die verbleibenden 31 Schnittstellen angezeigt wird.

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

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

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

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

PyTorch Veraltete Version von Anaconda Channel

Ab Version PyTorch 2.6 wird die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). PyTorch Infolgedessen werden Version PyTorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um das PyTorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

Datum der Veröffentlichung: 2025-06-03

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.7 (Ubuntu 22.04) 20250602

Hinzugefügt

- Erste Version der Deep Learning AMI GPU PyTorch 2.7 (Ubuntu 22.04) -Serie. Einschließlich einer virtuellen Python-Umgebung pytorch (source/opt/pytorch/bin/activate), ergänzt mit NVIDIA-Treiber R570, CUDA=12.8, cuDNN=9.10, NCCL=2.26.5 und EFA=1.40.0. PyTorch

Bekannte Probleme

- „Mit der Rechenkapazität sm10.0 (Blackwell-Architektur) GPUs enthält der FP8 Datentyp mit skalierter Aufmerksamkeit auf das Punktprodukt einen Deadlock, der dazu führt, dass der Kernel unter bestimmten Umständen hängen bleibt, z. B. wenn das Problem groß ist oder die GPU mehrere Kernel gleichzeitig ausführt. Ein Fix ist für eine future Version geplant.“ [CuDNN 9.10.0 Versionshinweise]
  - Benutzer, die P6-B200-Instances mit FP8 Daten und skalierter Aufmerksamkeit auf Punktprodukte ausführen möchten, sollten erwägen, Flash Attention manuell zu installieren.

AWS Deep-Learning-AMI-GPU PyTorch 2.6 (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen:

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI
- Der NVIDIA-Treiber Deep Learning mit OSS unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

Das AMI umfasst Folgendes:

- Unterstützter AWS Dienst: EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: x86
- NVIDIA CUDA12 6.6-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.6/ usr/local/cuda
  - Standard-CUDA: 12.6
    - PFAD/-12.6/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:

        - LD_LIBRARY_PATH soll/haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
        - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
    - Kompilierte NCCL-Version für 12.6:2.24.3
- Ort der NCCL-Tests:
    - all_reduce, all_gather und reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
    - Um NCCL-Tests auszuführen, wurde LD_LIBRARY_PATH bereits mit den erforderlichen Pfaden aktualisiert.
        - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs
            - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
        - LD_LIBRARY_PATH wurde mit CUDA-Versionspfaden aktualisiert
            - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- EFA-Installationsprogramm: 1.38.0
- Nvidia: 2.4.1 GDRCopy
- AWS OFI NCCL: 1.13.2-aws
    - AWS OFI NCCL unterstützt jetzt mehrere NCCL-Versionen mit einem einzigen Build
    - Der Installationspfad:/opt/amazon/ofi-nccl/ . Path /opt/amazon/ofi-nccl/libwurde zu LD_LIBRARY_PATH hinzugefügt.
- Python-Version: 3.12
- Python:/opt/pytorch/bin/python
- NVIDIA-Treiber: 570.86.15
- AWS CLI v2 bei/usr/bin/aws
- EBS-Volumetyp: gp3
- NVMe Standort des Instance-Speichers (bei unterstützten EC2 Instances):/opt/dlami/nvme
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
    - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Hinweise

PyTorch Abwertung des Anaconda-Kanals

Ab Version PyTorch 2.6 wird die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). PyTorch Infolgedessen werden Version PyTorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um das PyTorch Venv zu aktivieren, verwenden Sie bitte source// opt/pytorch/bin/activateP5/P5e Instances:

- DeviceIndex ist für jedes Exemplar eindeutig und muss eine nicht negative Ganzzahl sein NetworkCard, die unter dem Grenzwert von per liegt. ENIs NetworkCard Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind. Im Folgenden finden Sie ein Beispiel für einen Befehl zum Starten einer EC2 P5-Instanz mithilfe von awscli, der NetworkCardIndex von der Nummer 0-31 und DeviceIndex als 0 für die erste Schnittstelle und DeviceIndex als 1 für die restlichen 31 Schnittstellen angezeigt wird.

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

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

```
sudo dnf versionlock kernel*
```

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-02-21

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.6.0 (Amazon Linux 2023) 20250220

Hinzugefügt

- Erste Version des Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6 für Amazon Linux 2023
  - Ab Version PyTorch 2.6 unterstützt Pytorch Conda nicht mehr. Infolgedessen werden Pytorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um den Pytorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

AWS Deep-Learning-AMI-GPU PyTorch 2.6 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: x86
- Python:/opt/pytorch/bin/python
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 570.86.15
- NVIDIA 1.1-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.6/ usr/local/cuda
  - Standard-CUDA: 12.6
    - PFAD/-12.6/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
      - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
  - Die kompilierte System-NCCL-Version ist unter/usr/local/cuda/vorhanden: 2.24.3
  - PyTorch Kompilierte NCCL-Version aus der Conda-Umgebung: 2.21.5 PyTorch
- Ort der NCCL-Tests:
  - all_reduce, all_gather und reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - Um NCCL-Tests auszuführen, wurde LD_LIBRARY_PATH bereits mit den erforderlichen Pfaden aktualisiert.

  - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs
  - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
  - LD_LIBRARY_PATH wurde mit CUDA-Versionspfaden aktualisiert
  - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- EFA-Installationsprogramm: 1.38.0
- Nvidia: 2.4.1 GDRCopy
- Nvidia-Transformer-Engine: v1.11.0
- AWS OFI NCCL: 1.13.2-aws
  - Installationspfad:/wird zu LD_LIBRARY_PATH hinzugefügt. opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/lib
  - Hinweis: Das PyTorch Paket enthält auch ein dynamisch verlinktes AWS OFI-NCCL-Plugin als Conda-Paketpaket und PyTorch verwendet dieses aws-ofi-nccl-dlc Paket anstelle von System-OFI-NCCL. AWS
- AWS CLI v2 als aws2 und v1 als aws AWS CLI
- EBS-Volumetyp: gp3
- Python-Version: 3.11
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Hinweise

PyTorch Einstellung des Anaconda-Kanals

Ab Version PyTorch 2.6 hat Pytorch die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). Infolgedessen werden Pytorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um Pytorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

P5/P5e-Instanzen:

- DeviceIndex ist für jedes Exemplar eindeutig und muss eine nicht negative Ganzzahl sein NetworkCard, die unter dem Grenzwert von per liegt. ENIs NetworkCard Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind. Im Folgenden finden Sie ein Beispiel für einen Befehl zum Starten einer EC2 P5-Instanz mithilfe von awscli, der NetworkCardIndex von der Nummer 0-31 und DeviceIndex als 0 für die erste Schnittstelle und DeviceIndex als 1 für die restlichen 31 Schnittstellen angezeigt wird.

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

```
echo linux-aws hold | sudo dpkg –set-selections
```

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

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-02-21

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.6.0 (Ubuntu 22.04) 20250220

Hinzugefügt

- Erste Version der Deep Learning AMI GPU PyTorch 2.6 (Ubuntu 22.04) -Serie. Einschließlich einer virtuellen Python-Umgebung pytorch (source/opt/pytorch/bin/activate), ergänzt durch den NVIDIA-Treiber R570, CUDA=12.6, cuDNN=9.7, NCCL=2.21.5 und EFA=1.38.0. PyTorch
  - Ab PyTorch Version 2.6 hat Pytorch die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). Infolgedessen werden Pytorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um Pytorch Venv zu aktivieren, aktivieren Sie es bitte mit source/opt/pytorch/bin/activate

AWS Deep-Learning-AMI-GPU PyTorch 2.5 (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

Das AMI umfasst Folgendes:

- Unterstützter AWS Dienst: EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: x86
- NVIDIA CUDA12 4.4-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.4/ usr/local/cuda
  - Standard-CUDA: 12.4
    - PFAD/-12.4/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
      - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
  - Kompilierte NCCL-Version für 12.4:2.21.5
- Ort der NCCL-Tests:
  - all_reduce, all_gather und reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - Um NCCL-Tests auszuführen, wurde LD_LIBRARY_PATH bereits mit den erforderlichen Pfaden aktualisiert.
    - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs
      - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
  - LD_LIBRARY_PATH wurde mit CUDA-Versionspfaden aktualisiert
    - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- EFA-Installationsprogramm: 1.38.0
- Nvidia: 2.4.1 GDRCopy
- AWS OFI NCCL: 1.13.2-aws
  - AWS OFI NCCL unterstützt jetzt mehrere NCCL-Versionen mit einem einzigen Build

  - Installationspfad:/opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/libwurde zu LD_LIBRARY_PATH hinzugefügt.
  - Testet den Pfad für Ring, message_transfer:/opt/aws-ofi-nccl/tests
- Python-Version: 3.11
- Python:/opt/conda/envs/pytorch/bin/python
- NVIDIA-Treiber: 560.35.03
- AWS CLI v2 bei/usr/bin/aws
- EBS-Volumetyp: gp3
- NVMe Speicherort des Instance-Speichers (auf unterstützten EC2 Instances):/opt/dlami/nvme
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Hinweise

P5/P5e-Instanzen:

- DeviceIndex ist für jedes Exemplar eindeutig und muss eine nicht negative Ganzzahl sein NetworkCard, die unter dem Grenzwert von per liegt. ENIs NetworkCard Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind. Im Folgenden finden Sie ein Beispiel für einen Befehl zum Starten einer EC2 P5-Instanz

mithilfe von awscli, der NetworkCardIndex von der Nummer 0-31 und DeviceIndex als 0 für die erste Schnittstelle und DeviceIndex als 1 für die restlichen 31 Schnittstellen angezeigt wird.

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

  ```
  sudo dnf versionlock kernel*
  ```

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-02-17

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) 20250216

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Veröffentlichungsdatum: 2025-01-08

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) 20250107

Hinzugefügt

- Support für G4dn-Instanzen hinzugefügt.

Veröffentlichungsdatum: 21.11.2024

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.5.1 (Amazon Linux 2023) 20241120

Hinzugefügt

- Erste Version des Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5 für Amazon Linux 2023

Bekannte Probleme

- Dieses DLAMI unterstützt derzeit keine G4dn- und EC2 G5-Instances. AWS ist sich einer Inkompatibilität bewusst, die zu CUDA-Initialisierungsfehlern führen kann, die sich sowohl auf die G4dn- als auch auf die G5-Instance-Familien auswirken, wenn die Open-Source-NVIDIA-Treiber zusammen mit einer Linux-Kernel-Version 6.1 oder neuer verwendet werden. Dieses Problem betrifft unter anderem Linux-Distributionen wie Amazon Linux 2023, Ubuntu 22.04 oder neuer oder SUSE Linux Enterprise Server 15 SP6 oder neuer.

AWS Deep-Learning-AMI-GPU PyTorch 2.5 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: x86
- Python:/opt/conda/envs/pytorch/bin/python
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.144.03
- NVIDIA 4.4-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.4/ usr/local/cuda
  - Standard-CUDA: 12.4
    - PFAD/-12.4/ usr/local/cuda points to /usr/local/cuda

    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
      - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
    - Die kompilierte System-NCCL-Version ist unter/usr/local/cuda/vorhanden: 2.21.5
    - PyTorch Kompilierte NCCL-Version aus der Conda-Umgebung: 2.21.5 PyTorch
- Ort der NCCL-Tests:
    - all_reduce, all_gather und reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
    - Um NCCL-Tests auszuführen, wurde LD_LIBRARY_PATH bereits mit den erforderlichen Pfaden aktualisiert.
      - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs
        - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
    - LD_LIBRARY_PATH wurde mit CUDA-Versionspfaden aktualisiert
      - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- EFA-Installationsprogramm: 1.34.0
- Nvidia: 2.4.1 GDRCopy
- Nvidia-Transformer-Engine: v1.11.0
- AWS OFI NCCL: 1.11.0-aws
    - Installationspfad:/wird zu LD_LIBRARY_PATH hinzugefügt. opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/lib
    - Testet den Pfad für Ring, message_transfer:/opt/aws-ofi-nccl/tests
    - Hinweis: Das PyTorch Paket enthält auch das dynamisch verknüpfte AWS OFI-NCCL-Plugin als Conda-Paketpaket und PyTorch verwendet dieses aws-ofi-nccl-dlc Paket anstelle von System-OFI-NCCL. AWS
- AWS CLI v2 als aws2 und v1 als aws AWS CLI
- EBS-Volumetyp: gp3
- Python-Version: 3.11
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
    - OSS Nvidia-Treiber:

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

```
        --name /aws/service/deeplearning/ami/x86_64/oss-nvidia-driver-gpu-
pytorch-2.5-ubuntu-22.04/latest/ami-id  \
        --query "Parameter.Value" \
        --output text
```

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Hinweise

P5/P5e-Instanzen:

- DeviceIndex ist für jedes Exemplar eindeutig und muss eine nicht negative Ganzzahl sein NetworkCard, die unter dem Grenzwert von per liegt. ENIs NetworkCard Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind. Im Folgenden finden Sie ein Beispiel für einen Befehl zum Starten einer EC2 P5-Instanz mithilfe von awscli, der NetworkCardIndex von der Nummer 0-31 und DeviceIndex als 0 für die erste Schnittstelle und DeviceIndex als 1 für die restlichen 31 Schnittstellen angezeigt wird.

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

Kernel

- Die Kernel-Version wird mit dem folgenden Befehl gepinnt:

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

- Wir empfehlen Benutzern, die Aktualisierung ihrer Kernel-Version zu vermeiden (es sei denn, es liegt ein Sicherheitspatch vor), um die Kompatibilität mit den installierten Treibern und Paketversionen sicherzustellen. Wenn Benutzer dennoch ein Update durchführen möchten, können sie die folgenden Befehle ausführen, um ihre Kernelversionen zu entsperren:

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

- Für jede neue Version von DLAMI wird der neueste verfügbare kompatible Kernel verwendet.

Veröffentlichungsdatum: 2025-02-17

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250216

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Datum der Veröffentlichung: 2025-01-21

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250119

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert und entspricht nun der Adresse CVEs , die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom Januar 2025 enthalten ist.

Veröffentlichungsdatum: 21.11.2021

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.5.1 (Ubuntu 22.04) 20241121

Hinzugefügt

- Erste Version der Deep Learning AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) -Serie. Einschließlich einer Conda-Umgebung mit Pytorch, ergänzt durch den NVIDIA-Treiber R550, CUDA=12.4.1, CUDNN=8.9.7, NCCL=2.21.5 und EFA=1.37.0. PyTorch

Fixed

- Aufgrund einer Änderung im Ubuntu-Kernel zur Behebung eines Fehlers in der KASLR-Funktionalität (Kernel Address Space Layout Randomization) können G4Dn/G5-Instances CUDA auf dem OSS-Nvidia-Treiber nicht ordnungsgemäß initialisieren. Um dieses Problem zu beheben, enthält dieses DLAMI Funktionen, die den proprietären Treiber für G4Dn- und G5-Instances dynamisch laden. Bitte rechnen Sie mit einer kurzen Initialisierungszeit für diesen Ladevorgang, um sicherzustellen, dass Ihre Instanzen ordnungsgemäß funktionieren.
  - Um den Status und den Zustand dieses Dienstes zu überprüfen, können Sie die folgenden Befehle verwenden:

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

```
active
```

AWS Deep-Learning-AMI-GPU PyTorch 2.4 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, P4, P4de, P5, P5e, P5en.

Das AMI umfasst Folgendes:

- Unterstützter AWS Dienst: EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: x86
- Python:/opt/conda/envs/pytorch/bin/python
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.144.03
- NVIDIA 1.1-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.4/ usr/local/cuda
  - Standard-CUDA: 12.4
    - PFAD/-12.4/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/x86_64-linux/lib
      - PATH, um//zu haben usr/local/cuda/bin/:/usr/local/cuda/include
  - Die kompilierte System-NCCL-Version ist unter/usr/local/cuda/vorhanden: 2.21.5
  - PyTorch Kompilierte NCCL-Version aus der Conda-Umgebung: 2.20.5 PyTorch

- Ort der NCCL-Tests:
  - all_reduce, all_gather und reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
  - Um NCCL-Tests auszuführen, wurde LD_LIBRARY_PATH bereits mit den erforderlichen Pfaden aktualisiert.
    - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs
      - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
  - LD_LIBRARY_PATH wurde mit CUDA-Versionspfaden aktualisiert
    - /usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cud/targets/x86_64-linux/lib
- EFA-Installationsprogramm: 1.34.0
- Nvidia: 2.4.1 GDRCopy
- Nvidia-Transformer-Engine: v1.11.0
- AWS OFI NCCL: 1.11.0-aws
  - Installationspfad:/wird zu LD_LIBRARY_PATH hinzugefügt. opt/aws-ofi-nccl/ . Path /opt/aws-ofi-nccl/lib
  - Testet den Pfad für Ring, message_transfer:/opt/aws-ofi-nccl/tests
  - Hinweis: Das PyTorch Paket enthält auch das dynamisch verknüpfte AWS OFI-NCCL-Plugin als Conda-Paketpaket und PyTorch verwendet dieses aws-ofi-nccl-dlc Paket anstelle von System-OFI-NCCL. AWS
- AWS CLI v2 als aws2 und v1 als aws AWS CLI
- EBS-Volumetyp: gp3
- Python-Version: 3.11
- Fragen Sie die AMI-ID mit dem SSM-Parameter ab (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

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

Hinweise

P5/P5e-Instanzen

- DeviceIndex ist für jede einzelne eindeutig und muss eine nicht negative Ganzzahl sein NetworkCard, die unter dem Grenzwert von per liegt. ENIs NetworkCard Auf P5 NetworkCard ist die Anzahl von ENIs per 2, was bedeutet, dass die einzig gültigen Werte für 0 oder 1 DeviceIndex sind. Im Folgenden finden Sie ein Beispiel für einen Befehl zum Starten einer EC2 P5-Instanz mithilfe von awscli, der NetworkCardIndex von der Nummer 0-31 und DeviceIndex als 0 für die erste Schnittstelle und DeviceIndex als 1 für die restlichen 31 Schnittstellen angezeigt wird.

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

Veröffentlichungsdatum: 2025-02-17

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI-GPU PyTorch 2.4.1 (Ubuntu 22.04) 20250216

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Veröffentlichungsdatum: 2025-01-21

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20250119

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert, um CVE-Probleme im NVIDIA GPU Display Driver Security Bulletin für Januar 2025 zu beheben.

Veröffentlichungsdatum: 18.11.2024-

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI-GPU PyTorch 2.4.1 (Ubuntu 22.04) 20241116

Fixed

- Aufgrund einer Änderung im Ubuntu-Kernel zur Behebung eines Fehlers in der KASLR-Funktionalität (Kernel Address Space Layout Randomization) können G4Dn/G5-Instances CUDA auf dem OSS-Nvidia-Treiber nicht ordnungsgemäß initialisieren. Um dieses Problem zu beheben, enthält dieses DLAMI Funktionen, die den proprietären Treiber für G4Dn- und G5-Instances dynamisch laden. Bitte rechnen Sie mit einer kurzen Initialisierungszeit für diesen Ladevorgang, um sicherzustellen, dass Ihre Instanzen ordnungsgemäß funktionieren.
  - Um den Status und den Zustand dieses Dienstes zu überprüfen, können Sie die folgenden Befehle verwenden:

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

Datum der Veröffentlichung: 2024-10-16

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI-GPU PyTorch 2.4.1 (Ubuntu 22.04) 20241016

Hinzugefügt

- Nvidia TransformerEngine v1.11.0 zur Beschleunigung von Transformer-Modellen hinzugefügt (Weitere Informationen finden Sie unter Transformer- .html) https://docs.nvidia.com/deeplearning/engine/user-guide/index

Veröffentlichungsdatum: 2024-09-30

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20240929

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.16.1 auf 1.16.2 aktualisiert und die Sicherheitslücke CVE-2024-0133 behoben.

Veröffentlichungsdatum: 2024-09-26

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) 20240925

Hinzugefügt

- Erste Version der Deep Learning AMI GPU PyTorch 2.4.1 (Ubuntu 22.04) -Serie. Einschließlich einer Conda-Umgebung mit Pytorch, ergänzt durch den NVIDIA-Treiber R550, CUDA=12.4.1, CUDNN=8.9.7, NCCL=2.20.5 und EFA=1.34.0. PyTorch

ARM64 PyTorch Versionshinweise zu DLAMI

Im Folgenden finden Sie die Versionshinweise für: ARM64 PyTorch DLAMIs

GPU

- AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.7 (Amazon Linux 2023)
- AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.7 (Ubuntu 22.04)
- AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.6 (Amazon Linux 2023)
- AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.6 (Ubuntu 22.04)

- AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.5 (Ubuntu 22.04)
- AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.4 (Ubuntu 22.04)

AWS Neuron

- Weitere Informationen finden Sie im Neuron DLAMI-Benutzerhandbuch

AWS ARM64 Deep-Learning-AMI OSS-GPU PyTorch 2.7 (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning ARM64 AMI OSS Nvidia-Treiber GPU PyTorch 2.7 (Amazon Linux 2023) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: ARM64
- Linux-Kernel: 6.12
- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.8-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationsverzeichnisse:/-12.8/ usr/local/cuda
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/- haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa
      - PATH, um//zu haben usr/local/cuda/bin/:/usr/local/cuda/include

- AWS CLI v2 bei/usr/local/bin/aws
- EBS-Volumetyp: gp3
- Nvidia-Container-Toolkit: 1.17.7
  - Versionsbefehl: -V nvidia-container-cli
- Docker: 25.0.8
- Python:/usr/bin/python3.12
- AMI-ID mit SSM-Parameter abfragen (Beispielregion ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispielregion ist us-east-1):

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

Hinweise

PyTorch Veraltete Version von Anaconda Channel

Ab Version PyTorch 2.6 wird die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). PyTorch Infolgedessen werden Version PyTorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um das PyTorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

Datum der Veröffentlichung: 2025-05-22

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber GPU PyTorch 2.7 (Amazon Linux 2023) 20250521

Hinzugefügt

- Erste Veröffentlichung der Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Amazon Linux 2023) -Serie. Einschließlich einer virtuellen Python-Umgebung pytorch (source/

opt/pytorch/bin/activate), ergänzt mit NVIDIA-Treiber R570, CUDA=12.8, cuDNN=9.10 und NCCL=2.26.2 PyTorch

AWS ARM64 Deep-Learning-AMI OSS-GPU PyTorch 2.7 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning ARM64 AMI OSS Nvidia-Treiber GPU PyTorch 2.7 (Ubuntu 22.04) $ {YYYY-MM-DD}

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: ARM64
- Linux-Kernel: 6.8
- NVIDIA-Treiber: 570.133.20
- NVIDIA CUDA 12.8-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationsverzeichnisse:/-12.8/ usr/local/cuda
  - Standard-CUDA: 12.8
    - PATH//usr/local/cudazeigt auf CUDA 12.8
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/- haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa
      - PATH, um//zu haben usr/local/cuda/bin/:/usr/local/cuda/include
- AWS CLI v2 bei/usr/local/bin/aws
- EBS-Volumetyp: gp3
- Nvidia-Container-Toolkit: 1.17.7

  - Versionsbefehl: -V nvidia-container-cli
- Docker: 28.2.2
- Python:/usr/bin/python3.12
- AMI-ID mit SSM-Parameter abfragen (Beispielregion ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispielregion ist us-east-1):

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

Hinweise

PyTorch Veraltete Version von Anaconda Channel

Ab Version PyTorch 2.6 wird die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). PyTorch Infolgedessen werden Version PyTorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um das PyTorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

Datum der Veröffentlichung: 2025-05-22

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber GPU PyTorch 2.7 (Ubuntu 22.04) 20250521

Hinzugefügt

- Erste Veröffentlichung der Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.7 (Ubuntu 22.04) -Serie. Einschließlich einer virtuellen Python-Umgebung pytorch (source/opt/pytorch/bin/activate), ergänzt mit NVIDIA-Treiber R570, CUDA=12.8, cuDNN=9.10 und NCCL=2.26.2 PyTorch

AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.6 (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Dienst: EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: ARM64
- Python:/opt/pytorch/bin/python
- Python-Version: 3.12
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 570.86.15
- CUDA12NVIDIA 6.6-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.6/ usr/local/cuda
  - Standard-CUDA: 12.6
    - PFAD/-12.6/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
  - Die kompilierte System-NCCL-Version ist unter/usr/local/cuda/vorhanden: 2.24.3
  - PyTorch Kompilierte NCCL-Version aus der Conda-Umgebung: 2.21.5 PyTorch
- AWS CLI v2 als aws2 und v1 als aws AWS CLI

- EBS-Volumetyp: gp3
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):

```
aws ssm get-parameter --region us-east-1 \
    --name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.6-
amazon-linux-2023/latest/ami-id  \
    --query "Parameter.Value" \
    --output text
```

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):

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

Hinweise

PyTorch Veraltete Version von Anaconda Channel

Ab Version PyTorch 2.6 wird die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). PyTorch Infolgedessen werden Version PyTorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um das PyTorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

Datum der Veröffentlichung: 2025-02-21

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.6.0 (Amazon Linux 2023) 20250221

Hinzugefügt

- Erste Version der Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Amazon Linux 2023) -Serie. Einschließlich einer virtuellen Python-Umgebung pytorch (source/opt/pytorch/bin/activate/), ergänzt durch den NVIDIA-Treiber R570, CUDA=12.6, cuDNN=9.7, NCCL=2.21.5. PyTorch

AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.6 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: ARM64
- Python:/opt/pytorch/bin/python
- Python-Version: 3.12
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 570.86.15
- NVIDIA 6.6-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.6/ usr/local/cuda
  - Standard-CUDA: 12.6
    - PFAD/-12.6/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
  - Die kompilierte System-NCCL-Version ist unter/usr/local/cuda/vorhanden: 2.24.3
  - PyTorch Kompilierte NCCL-Version aus der Venv-Umgebung: 2.21.5 PyTorch
- AWS CLI v2 als aws2 und v1 als aws AWS CLI

- EBS-Volumetyp: gp3
- AMI-ID mit SSM-Parameter abfragen (Beispielregion ist us-east-1):

```
aws ssm get-parameter --region us-east-1 \
    --name /aws/service/deeplearning/ami/arm64/oss-nvidia-driver-gpu-pytorch-2.6-
ubuntu-22.04/latest/ami-id  \
    --query "Parameter.Value" \
    --output text
```

- AMI-ID abfragen mit AWSCLI (Beispielregion ist us-east-1):

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

Hinweise

PyTorch Veraltete Version von Anaconda Channel

Ab Version PyTorch 2.6 wird die Unterstützung für Conda eingestellt (siehe offizielle Ankündigung). PyTorch Infolgedessen werden Version PyTorch 2.6 und höher zur Verwendung von virtuellen Python-Umgebungen übergehen. Um das PyTorch Venv zu aktivieren, verwenden Sie bitte source/opt/pytorch/bin/activate

Datum der Veröffentlichung: 2025-02-21

AMI-Name: Deep Learning ARM64 AMI OSS NVIDIA-Treiber-GPU PyTorch 2.6.0 (Ubuntu 22.04) 20250221

Hinzugefügt

- Erste Version der Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.6.0 (Ubuntu 22.04) -Serie. Einschließlich einer virtuellen Python-Umgebung pytorch (source/opt/pytorch/bin/activate), ergänzt durch den NVIDIA-Treiber R570, CUDA=12.6, cuDNN=9.7, NCCL=2.21.5. PyTorch

AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.5 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Deep Learning ARM64 AMI OSS Nvidia-Treiber GPU PyTorch 2.5. $ {PATCH-VERSION} (Ubuntu 22.04) $ {YYY-MM-DD}

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: ARM64
- Python:/opt/conda/envs/pytorch/bin/python
- Python-Version: 3.11
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.144.03
- NVIDIA 4.4-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.4/ usr/local/cuda
  - Standard-CUDA: 12.4
    - PFAD/-12.4/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
  - Die kompilierte System-NCCL-Version ist unter/usr/local/cuda/vorhanden: 2.21.5
  - PyTorch Kompilierte NCCL-Version aus der Conda-Umgebung: 2.21.5 PyTorch
- AWS CLI v2 als aws2 und v1 als aws AWS CLI

- EBS-Volumetyp: gp3
- Fragen Sie die AMI-ID mit dem SSM-Parameter ab (Beispiel Region ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):

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

Datum der Veröffentlichung: 2025-02-17

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250215

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Datum der Veröffentlichung: 2025-01-21

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.5.1 (Ubuntu 22.04) 20250117

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert und entspricht nun der Adresse CVEs , die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom Januar 2025 enthalten ist.

Veröffentlichungsdatum: 22.11.2024

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.5.1 (Ubuntu 22.04) 20241122

Hinzugefügt

- Erste Version der Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.5.1 (Ubuntu 22.04) -Serie. Einschließlich einer Conda-Umgebung mit Pytorch, ergänzt durch den NVIDIA-Treiber R550, CUDA=12.4, CUDNN=8.9.7, NCCL=2.21.5. PyTorch

AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.4 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- G5g

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: ARM64
- Python:/opt/conda/envs/pytorch/bin/python
- Python-Version: 3.11
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.144.03
- NVIDIA 1.1-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.4/ usr/local/cuda
  - Standard-CUDA: 12.4
    - PFAD/-12.4/ usr/local/cuda points to /usr/local/cuda
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/64 haben usr/local/cuda/lib:/usr/local/cuda/lib64:/usr/local/cuda:/usr/local/cuda/targets/sbsa-linux/lib:/usr/local/cuda/nvvm/lib64:/usr/local/cuda/extras/CUPTI/lib
      - PATH soll//haben usr/local/cuda/bin/:/usr/local/cuda/include
  - Die kompilierte System-NCCL-Version ist unter/usr/local/cuda/vorhanden: 2.21.5
  - PyTorch Kompilierte NCCL-Version aus der Conda-Umgebung: 2.20.5 PyTorch
- AWS CLI v2 als aws2 und v1 als aws AWS CLI
- EBS-Volumetyp: gp3
- Fragen Sie die AMI-ID mit dem SSM-Parameter ab (Beispiel Region ist us-east-1):

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):

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

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

Datum der Veröffentlichung: 2025-02-17

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.4.0 (Ubuntu 22.04) 20250215

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Datum der Veröffentlichung: 2025-01-21

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.4.0 (Ubuntu 22.04) 20250117

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert und entspricht nun der Adresse CVEs , die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom Januar 2025 enthalten ist.

Veröffentlichungsdatum: 30.09.2020

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.4.0 (Ubuntu 22.04) 20240927

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.16.1 auf 1.16.2 aktualisiert und die Sicherheitslücke CVE-2024-0133 behoben.

Veröffentlichungsdatum: 2024-09-26

AMI-Name: Deep Learning ARM64 AMI OSS Nvidia-Treiber-GPU PyTorch 2.4.0 (Ubuntu 22.04) 20240926

Hinzugefügt

- Erste Version der Deep Learning ARM64 AMI OSS Nvidia Driver GPU PyTorch 2.4.0 (Ubuntu 22.04) -Serie. Einschließlich einer Conda-Umgebung mit Pytorch, ergänzt durch den NVIDIA-Treiber R550, CUDA=12.4, CUDNN=8.9.7, NCCL=2.20.5. PyTorch

TensorFlow DLAMIs

TensorFlow Versionshinweise zu DLAMI

- X86 TensorFlow DLAMI Versionshinweise

X86 TensorFlow DLAMI Versionshinweise

Im Folgenden finden Sie die Versionshinweise für X86 TensorFlow DLAMI:

GPU

- AWS Deep-Learning-AMI-GPU TensorFlow 2.18 (Amazon Linux 2023)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.18 (Ubuntu 22.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.17 (Ubuntu 22.04)

AWS Neuron

- Weitere Informationen finden Sie im Neuron DLAMI Guide.

Deep-Learning-AMI-GPU TensorFlow 2.18 (Amazon Linux 2023)

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2023
- Rechenarchitektur: x86
- Python:/opt/tensorflow/bin/python3.12
- TensorFlow Ausführung: 2.18
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 560.35.03
- NVIDIA-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.5/ usr/local/cuda
- EFA-Installationsprogramm: 1.37.0
- AWS CLI v2 als aws2 und v1 als aws AWS CLI
- EBS-Volumetyp: gp3
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

```
    --output text
```

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Veröffentlichungsdatum: 2025-02-17

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.18 (Amazon Linux 2023) 20241215

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Veröffentlichungsdatum: 2024-12-09

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.18 (Amazon Linux 2023) 20241206

Hinzugefügt

- Erste Version der Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023) -Serie.
  - Die Software umfasst Folgendes:
    - „nvidia-driver=560.35.03"
    - „Fabric-Manager=560.35.03"
    - „cuda=12,5“
    - „cudnn=9,5.1"
    - „efa=1,37,0“
    - „nccl=2,23,4“
    - „aws-nccl-ofi-plugin=v1.13.0-aws“
- Die virtuelle Tensorflow-Umgebung (Aktivierungsbefehlsquelle/) umfasst Folgendes: opt/tensorflow/bin/activate
  - „tensorflow = 2.18.0“

Deep-Learning-AMI-GPU TensorFlow 2.18 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unter. Erste Schritte mit DLAMI

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e, P5en.

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: x86
- Python:/opt/tensorflow/bin/python3.12

- TensorFlow Ausführung: 2.18
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.144.03
- CUDA12 NVIDIA-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.5/ usr/local/cuda
- EFA-Installationsprogramm: 1.37.0
- AWS CLI v2 als aws2 und v1 als aws AWS CLI
- EBS-Volumetyp: gp3
- Fragen Sie die AMI-ID mit dem SSM-Parameter ab (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Datum der Veröffentlichung: 2025-02-17

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.18 (Ubuntu 22.04) 20250215

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows

sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Datum der Veröffentlichung: 20.01.2025

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.18 (Ubuntu 22.04) 20250118

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.90.07 auf 550.127.05 aktualisiert und entspricht nun der Adresse, die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom Januar 2025 enthalten ist CVEs

Veröffentlichungsdatum: 2024-12-09

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.18 (Ubuntu 22.04) 20241206

Hinzugefügt

- Erste Version der Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Ubuntu 22.04) - Serie.
  - Die Software umfasst Folgendes:
    - „nvidia-driver=550.127.05"
    - „Fabric-Manager=550.127.05"
    - „cuda=12,5“
    - „cudnn=9,5.1"
    - „efa=1,37,0“
    - „nccl=2,23,4“
    - „aws-nccl-ofi-plugin=v1.13.0-aws“

- Die virtuelle Tensorflow-Umgebung (Aktivierungsbefehlsquelle/) umfasst Folgendes: opt/tensorflow/bin/activate
  - „tensorflow = 2.18.0“

Fixed

- Aufgrund einer Änderung im Ubuntu-Kernel zur Behebung eines Fehlers in der KASLR-Funktionalität (Kernel Address Space Layout Randomization) können G4Dn/G5-Instances CUDA auf dem OSS-Nvidia-Treiber nicht ordnungsgemäß initialisieren. Um dieses Problem zu beheben, enthält dieses DLAMI Funktionen, die den proprietären Treiber für G4Dn- und G5-Instances dynamisch laden. Bitte rechnen Sie mit einer kurzen Initialisierungszeit für diesen Ladevorgang, um sicherzustellen, dass Ihre Instanzen ordnungsgemäß funktionieren.
  - Um den Status und den Zustand dieses Dienstes zu überprüfen, können Sie die folgenden Befehle verwenden:

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

Deep-Learning-AMI-GPU TensorFlow 2.17 (Ubuntu 22.04)

Hilfe zu den ersten Schritten finden Sie unter. Erste Schritte mit DLAMI

AMI-Namensformat

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

EC2 Unterstützte Instanzen

- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5, P5e. P5en.

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Ubuntu 22.04
- Rechenarchitektur: x86

- Python:/opt/tensorflow/bin/python3.12
- TensorFlow Ausführung: 2.17
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.144.03
- CUDA12 NVIDIA-Stapel:
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-12.3/ usr/local/cuda
- EFA-Installationsprogramm: 1.34.0
- AWS CLI v2 als aws2 und v1 als aws AWS CLI
- EBS-Volumetyp: gp3
- AMI-ID mit SSM-Parameter abfragen (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispiel Region ist us-east-1):
  - OSS Nvidia-Treiber:

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

Datum der Veröffentlichung: 2025-02-17

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20250215

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v

- In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial Wenn Sie eine CUDA-Kompatibilitätsschicht verwenden gezeigt.

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Datum der Veröffentlichung: 20.01.2025

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20250118

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert und entspricht nun der Adresse, die im Sicherheitsbulletin für NVIDIA GPU-Bildschirmtreiber vom Januar 2025 enthalten ist CVEs

Version 2.17.1

Veröffentlichungsdatum: 2024-11-18

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20241115

Fixed

- Aufgrund einer Änderung im Ubuntu-Kernel zur Behebung eines Fehlers in der KASLR-Funktionalität (Kernel Address Space Layout Randomization) können G4Dn/G5-Instances CUDA auf dem OSS-Nvidia-Treiber nicht ordnungsgemäß initialisieren. Um dieses Problem zu beheben, enthält dieses DLAMI Funktionen, die den proprietären Treiber für G4Dn- und G5-Instances dynamisch laden. Bitte rechnen Sie mit einer kurzen Initialisierungszeit für diesen Ladevorgang, um sicherzustellen, dass Ihre Instanzen ordnungsgemäß funktionieren.
  - Um den Status und den Zustand dieses Dienstes zu überprüfen, können Sie die folgenden Befehle verwenden:

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

Version 2.17.0

Veröffentlichungsdatum: 2024-09-25

AMI-Name: Deep Learning OSS Nvidia-Treiber AMI GPU TensorFlow 2.17 (Ubuntu 22.04) 20240924

Hinzugefügt

- Erste Version der Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.17 (Ubuntu 22.04) - Serie.
  - Die Software umfasst Folgendes:
    - „nvidia-driver=550.90.07"
    - „Fabric-Manager=550.90.07"
    - „cuda=12,3“
    - „cudnn=8,9,7"
    - „efa=1,34,0"
    - „nccl=2,22,3“
    - „aws-nccl-ofi-plugin=v1.11.0-aws“
- Die virtuelle Tensorflow-Umgebung (Aktivierungsbefehlsquelle/) umfasst Folgendes: opt/tensorflow/bin/activate
  - „tensorflow = 2.17.0“

Versionshinweise für Multi-Framework DLAMIs

> **Tip**
>
> Wenn Sie nur ein Framework für maschinelles Lernen verwenden, empfehlen wir ein DLAMI mit einem einzigen Framework.

Versionshinweise zu Multi Framework DLAMI

- Versionshinweise zu Multi-Framework DLAMI

Versionshinweise zu Multi-Framework DLAMI

Im Folgenden finden Sie die Versionshinweise für Multi-Framework X86 DLAMI:

GPU

- AWS Deep-Learning-AMI (Amazon Linux 2)

AWS Neuron

- Weitere Informationen finden Sie im Neuron DLAMI-Benutzerhandbuch

AWS Deep-Learning-AMI (Amazon Linux 2)

> **Tip**
>
> Kunden, die ein einzelnes Framework verwenden, wie das TensorFlow hier PyTorch erwähnte einzelne Framework DLAMIs , oder es wird empfohlen, es zu verwenden

Hilfe zu den ersten Schritten finden Sie unterErste Schritte mit DLAMI.

AMI-Namensformat

- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2), Version $ {XX.X}
- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version $ {XX.X}

Unterstützte Instanzen EC2

- Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI.
- Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, G6, Gr6, G6e, P4d, P4de, P5
- Deep Learning mit proprietärem Nvidia-Treiber unterstützt G3 (G3.16x nicht unterstützt), P3, P3dn

Das AMI umfasst Folgendes:

- Unterstützter AWS Service: Amazon EC2
- Betriebssystem: Amazon Linux 2

- Rechenarchitektur: x86
- Framework- und Python-Versionen für Conda-Umgebungen:
  - Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2):
    - python3: Python 3.10
    - tensorflow2_p310:2.16, Python 3.10 TensorFlow
    - pytorch_p310:2.2, Python 3.10 PyTorch
  - Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2):
    - python3: Python 3.10
    - tensorflow2_p310:2.16, Python 3.10 TensorFlow
    - pytorch_p310:2.2, Python 3.10 PyTorch
- NVIDIA-Treiber:
  - OS Nvidia-Treiber: 550.163.01
  - Proprietärer Nvidia-Treiber: 550.163.01
- NVIDIA 1.1-12.4-Stapel: CUDA12
  - CUDA-, NCCL- und cuDDN-Installationspfad:/-xx.x/ usr/local/cuda
  - Standard-CUDA: 12.1
    - PATH//usr/local/cudazeigt auf 1. CUDA12
    - Die folgenden Umgebungsvariablen wurden aktualisiert:
      - LD_LIBRARY_PATH soll/haben usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1:/usr/local/cuda-12.1/targets/x86_64-linux/lib
      - PATH soll//haben usr/local/cuda-12.1/bin/:/usr/local/cuda-11.8/include
    - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
  - Kompilierte NCCL-Version für CUDA 12.1-12.4:2.22.3
  - Standort der NCCL-Tests:
    - all_reduce, all_gather und reduce_scatter:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
    - Um NCCL-Tests ausführen zu können, muss LD_LIBRARY_PATH mit den folgenden Aktualisierungen bestanden werden.
      - Häufig verwendete Dateien wurden bereits zu LD_LIBRARY_PATH hinzugefügt: PATHs
        - /opt/amazon/efa/lib:/opt/amazon/openmpi/lib:/opt/aws-ofi-nccl/lib:/usr/local/lib:/usr/lib
      - Für jede andere CUDA-Version aktualisieren Sie LD_LIBRARY_PATH bitte entsprechend.
- EFA-Installationsprogramm: 1.38.0

- GDRCopy: 2,4
- AWS OFI NCCL: 1.13.2
  - Standort des Systems:/usr/local/cuda-xx.x/efa
  - Dies wird hinzugefügt, um NCCL-Tests auszuführen, die sich unter/-cuda-xx.x/ befinden usr/local/cuda-xx.x/efa/test
  - Außerdem enthält PyTorch das Paket ein dynamisch verlinktes AWS OFI-NCCL-Plugin als Conda-Paketpaket und verwendet dieses aws-ofi-nccl-dlc Paket anstelle von System-OFI-NCCL. PyTorch AWS
- Ort der NCCL-Tests:/-cuda-xx.x/ usr/local/cuda-xx.x/efa/test
- AWS CLI v2 bei//2 und v1 bei/usr/local/bin/aws AWS CLI usr/local/bin/aws
- EBS-Volumetyp: gp3
- AMI-ID mit SSM-Parameter abfragen (Beispielregion ist us-east-1):
  - OSS Nvidia-Treiber:

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

  - Eigener Nvidia-Treiber:

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

- AMI-ID abfragen mit AWSCLI (Beispielregion ist us-east-1):
  - OSS Nvidia-Treiber:

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

  - Eigener Nvidia-Treiber:

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

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

Hinweise

EFA-Updates von 1.37 auf 1.38 (Veröffentlichung am 05.02.2025)

- EFA bündelt jetzt das AWS OFI-NCCL-Plugin, das jetzt in/-ofi-nccl/ zu finden ist. opt/amazon/ofi-nccl rather than the original /opt/aws Wenn Sie Ihre Variable LD_LIBRARY_PATH aktualisieren, stellen Sie bitte sicher, dass Sie Ihren OFI-NCCL-Speicherort korrekt ändern.

Entfernung der Neuron Conda-Umgebung

- Der proprietäre Nvidia-Treiber von Deep Learning, der nach dem 18. Juli 2024 AMIs veröffentlicht wurde, wird ohne Neuron-Conda-Umgebungen für und ausgeliefert. PyTorch TensorFlow Bitte verwenden Sie stattdessen das DLAMIs Neuron in den DLAMI-Versionshinweisen, um neuronale Umgebungen zu nutzen.

Entfernen von Audit-Paketen

- DLAMIs, die zwischen dem 26. März 2024 (2024-03-26) und dem 12. April 2024 (2024-04-12) veröffentlicht wurden, wurden ohne das Auditpaket ausgeliefert. Wenn Sie dieses spezielle Paket für Ihre Protokollierungs- und Überwachungsanforderungen benötigen, migrieren Sie Ihre Workflows bitte auf die neueste DLAMI-Version, um sie mit dem installierten Audit-Paket nutzen zu können.

Horovod

- Horovod wurde aus den aktuellen Conda-Umgebungen pytorch_p310 und tensorflow2_p310 auf dem DLAMI entfernt. Kunden werden in der Lage sein, die Horovod-Bibliotheken gemäß den Horovod-Richtlinien zu installieren und sie auf ihren verteilten Schulungsaufträgen zu installieren. DLAMIs

Datum der Veröffentlichung: 2025-04-22

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 81.2
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 81.2

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.144.03 auf 550.163.01 aktualisiert, um die im Sicherheitsbulletin für NVIDIA-GPU-Bildschirmtreiber vom CVEs April 2025 enthaltene Adresse zu ersetzen

Veröffentlichungsdatum: 2025-02-17

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 80.6
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 80.4

Aktualisiert

- Das NVIDIA Container Toolkit wurde von Version 1.17.3 auf Version 1.17.4 aktualisiert
  - Weitere Informationen finden Sie auf der Seite mit den Versionshinweisen hier:/1.17.4 https://github.com/NVIDIA/ nvidia-container-toolkit releases/tag/v
  - In der Container Toolkit-Version 1.17.4 ist das Mounten von CUDA-kompatiblen Bibliotheken jetzt deaktiviert. Um die Kompatibilität mit mehreren CUDA-Versionen in Container-Workflows sicherzustellen, stellen Sie bitte sicher, dass Sie Ihren LD_LIBRARY_PATH so aktualisieren, dass er Ihre CUDA-Kompatibilitätsbibliotheken enthält, wie im Tutorial „Wenn Sie eine CUDA-Kompatibilitätsebene verwenden“ hier beschrieben - -gpu-drivers.html# https://docs.aws.amazon.com/sagemaker/ latest/dg/inference collapsible-cuda-compat

Entfernt

- Die Benutzerbereichsbibliotheken cuobj und nvdisasm, die vom NVIDIA CUDA-Toolkit bereitgestellt wurden, um die im NVIDIA CUDA Toolkit Security Bulletin vom 18. Februar 2025 enthaltenen Probleme zu beheben, wurden entfernt CVEs

Veröffentlichungsdatum: 05.02.2025

AMI-Namen

- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 80.2
- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 80.4

Aktualisiert

- Die EFA-Version wurde von 1.37.0 auf 1.38.0 aktualisiert
  - EFA bündelt jetzt das AWS OFI-NCCL-Plugin, das sich jetzt in/-ofi-nccl/ befindet. opt/amazon/ofi-nccl rather than the original /opt/aws Wenn Sie Ihre Variable LD_LIBRARY_PATH aktualisieren, stellen Sie bitte sicher, dass Sie Ihren OFI-NCCL-Speicherort korrekt ändern.

Datum der Veröffentlichung: 15.01.2025

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 80.3
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 80.1

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.127.05 auf 550.144.03 aktualisiert, um die im Sicherheitsbulletin für NVIDIA-GPU-Bildschirmtreiber vom CVEs Januar 2025 enthaltene Adresse zu ersetzen

Veröffentlichungsdatum: 2024-12-09

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 80.1
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 79.9

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.17.0 auf 1.17.3 aktualisiert

Veröffentlichungsdatum: 2024-11-11

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 79.9
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 79.7

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.16.2 auf 1.17.0 aktualisiert und die Sicherheitslücke CVE-2024-0134 behoben.

Datum der Veröffentlichung: 22.10.2024

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 79.6
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 79.6

Aktualisiert

- Der Nvidia-Treiber wurde von Version 550.90.07 auf 550.127.05 aktualisiert und entspricht nun der Adresse, die im NVIDIA GPU-Display-Sicherheitsbulletin für CVEs Oktober 2024 enthalten ist

Veröffentlichungsdatum: 2024-10-03

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 79.3
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 79.3

Aktualisiert

- Das Nvidia Container Toolkit wurde von Version 1.16.1 auf 1.16.2 aktualisiert und die Sicherheitslücke CVE-2024-0133 behoben.

Veröffentlichungsdatum: 2024-07-18

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 78.6
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 78.7

Aktualisiert

- Die Conda-Umgebungen aws_neuron_pytorch_p38 und aws_neuron_tensorflow_p38 wurden aus dem proprietären Nvidia-Treiber-AMI von Deep Learning entfernt.
- Die Unterstützung der Inf1-Instance-Familie wurde aus dem proprietären Nvidia-Treiber-AMI von Deep Learning entfernt.

Datum der Veröffentlichung: 2024-06-06

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 78.5
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 78.5

Aktualisiert

- Die Nvidia-Treiberversion wurde von 535.161.08 auf 535.183.01 aktualisiert

Datum der Veröffentlichung: 2024-05-17

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 78.1
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 78.1

Aktualisiert

- Torchserve wurde in der Umgebung pytorch_p310 von v0.8.2 auf v0.11.0 aktualisiert.

Veröffentlichungsdatum: 2024-05-07

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 78.0
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 78.0

Aktualisiert

- TensorFlow Version wurde in der tensorflow2_p310-Umgebung von 2.15 auf 2.16 aktualisiert.
- Die EFA-Version wurde von Version 1.30 auf Version 1.32 aktualisiert
- Das AWS OFI NCCL-Plugin wurde von Version 1.7.4 auf Version 1.9.1 aktualisiert
- Das Nvidia-Container-Toolkit wurde von Version 1.13.5 auf Version 1.15.0 aktualisiert
  - HINWEIS: Version 1.15.0 enthält NICHT die Pakete und nvidia-docker2. nvidia-container-runtime Es wird empfohlen, nvidia-container-toolkit Pakete direkt zu verwenden, indem Sie den Dokumenten zum Nvidia-Container-Toolkit folgen.

Hinzugefügt

- CUDA123.3-Stack mit CUDA12 .3, NCCL 2.21.5, cuDNN 8.9.7 hinzugefügt

Entfernt

- Die Stapel .7, CUDA11 .0 wurden entfernt, die bei/-12.0 vorhanden waren CUDA12 usr/local/cuda-11.7 and /usr/local/cuda
- Das nvidia-docker2-Paket und sein Befehl nvidia-docker wurden als Teil des Nvidia-Container-Toolkit-Updates von 1.13.5 auf 1.15.0 entfernt, das NICHT die Pakete und nvidia-docker2 enthält. nvidia-container-runtime

Veröffentlichungsdatum: 2024-04-04

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 77.0
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 77.0

Aktualisiert

- PyTorch Version wurde in der Umgebung pytorch_p310 von 2.1 auf 2.2 aktualisiert.
- Für den OSS-Nvidia-Treiber wurde Unterstützung für DLAMIs G6- und Gr6-Instanzen hinzugefügt. EC2 Weitere Informationen finden Sie auf der Seite zur EC2 Instanzauswahl.

Veröffentlichungsdatum: 2024-03-29

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 76.8
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 76.9

Aktualisiert

- Der Nvidia-Treiber wurde sowohl im proprietären als auch im OSS-Nvidia-Treiber von 535.104.12 auf 535.161.08 aktualisiert. DLAMIs
- Die neuen unterstützten Instanzen für jedes DLAMI lauten wie folgt:
  - Deep Learning mit proprietärem Nvidia-Treiber unterstützt G3 (G3.16x nicht unterstützt), P3, P3dn, Inf1
  - Deep Learning mit OSS Der Nvidia-Treiber unterstützt G4dn, G5, P4d, P4de.

Entfernt

- Die Unterstützung für G4dn-, G5- und EC2 G3.16x-Instanzen wurde aus dem proprietären Nvidia-Treiber DLAMI entfernt.

Version 76.8

Datum der Veröffentlichung: 2024-03-20

AMI-Namen

- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 76.8

Hinzugefügt

- awscliv2 wurde dem AMI als//usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awsauf dem proprietären Nvidia-Treiber-AMI hinzugefügt

Version 76.7

Datum der Veröffentlichung: 2024-03-20

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 76.7

Hinzugefügt

- awscliv2 wurde im AMI als//usr/local/bin/aws2, alongside awscliv1 as /usr/local/bin/awsauf dem OSS Nvidia Driver AMI hinzugefügt
- Aktualisierter OSS Nvidia-Treiber DLAMI mit G4dn- und G5-Unterstützung, basierend darauf sieht die aktuelle Unterstützung wie folgt aus:
  - Das proprietäre Nvidia-Treiber-AMI von Deep Learning Base (Amazon Linux 2) unterstützt P3, P3dn, G3, G5, G4dn.
  - Das Deep Learning Base OSS Nvidia-Treiber-AMI (Amazon Linux 2) unterstützt G4dn, G5, P4, P5.
- DLAMIs Es wird empfohlen, die OSS-Nvidia-Treiber für G4dn, G5, P4, P5 zu verwenden.

Version 76.3

Datum der Veröffentlichung: 2024-02-14

Aktualisiert

- TensorFlow Von 2.13.0 auf 2.15.0 aktualisiert
- EFA wurde von 1.29.0 auf 1.30.0 aktualisiert
- -OFI-NCCL wurde von 1.7.3-aws auf 1.7.4-aws aktualisiert AWS
- Der Nvidia-Treiber wurde auf dem proprietären Deep Learning-Treiber-AMI auf 535.104.12 aktualisiert
- Aktualisierter Nvidia-Treiber auf 535.154.05 auf Deep Learning OSS Nvidia Driver AMI

Version 76.2

Datum der Veröffentlichung: 2024-02-02

AMI-Namen

- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 76.2
- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 76.4

Sicherheit

- Die Runc-Paketversion wurde aktualisiert, um den Patch für CVE-2024-21626 zu verwenden.

Version 7.6.1

Datum der Veröffentlichung: 2023-12-27

Aktualisiert

- PyTorch Von 2.0.1 auf 2.1.0 aktualisiert

Version 75.1

Datum der Veröffentlichung: 2023-11-17

Weitere Informationen finden Sie unter Wichtige Änderungen an DLAMI

AMI-Namen

- Deep Learning OSS Nvidia-Treiber-AMI (Amazon Linux 2) Version 75.1
- Proprietäres Nvidia-Treiber-AMI für Deep Learning (Amazon Linux 2) Version 75.1

Hinzugefügt

- AWS Deep Learning AMI (DLAMI) ist in zwei separate Gruppen aufgeteilt:
  - DLAMI, das den proprietären Treiber von Nvidia verwendet (zur Unterstützung von P3, P3dn, G3, G5, G4dn).
  - DLAMI, das den Nvidia OSS-Treiber verwendet, um EFA zu aktivieren (zur Unterstützung von P4, P5).

- Weitere Informationen zu DLAMI Split finden Sie in der öffentlichen Ankündigung.
- AWS CLI-Abfragen für oben finden Sie in den Versionshinweisen unter dem Aufzählungspunkt ABFRAGE AMI-ID mit AWSCLI (Beispielregion ist us-east-1)

Aktualisiert

- EFA wurde von 1.26.1 auf 1.29.0 aktualisiert
- GDRCopy aktualisiert von 2.3 auf 2.4

Version 74.4

Datum der Veröffentlichung: 2023-10-27

Aktualisiert

- AWS Das OFI NCCL Plugin wurde von Version 1.7.2 auf Version 1.7.3 aktualisiert
- Die CUDA 12.0-12.1-Verzeichnisse wurden mit der NCCL-Version 2.18.5 aktualisiert
- CUDA12.1 wurde als Standard-CUDA-Version aktualisiert
  - LD_LIBRARY_PATH wurde auf//aktualisiert usr/local/cuda-12.1/targets/x86_64-linux/lib/:/usr/local/cuda-12.1/lib:/usr/local/cuda-12.1/lib64:/usr/local/cuda-12.1 and PATH to have /usr/local/cuda-12.1/bin
  - Für Kunden, die zu einer anderen CUDA-Version wechseln möchten, definieren Sie bitte die Variablen LD_LIBRARY_PATH und PATH entsprechend.
- Pillow wurde von Version 9.4.0 auf 10.1.0 aktualisiert, um SNYK-PYTHON-PILLOW-5918878 in allen Conda-Umgebungen zu korrigieren
- Opencv-Python wurde von 4.8.0.74 auf 4.8.1.78 aktualisiert, um SNYK-PYTHON-OPENCVPYTHON-5926695 in allen Conda-Umgebungen zu reparieren

Hinzugefügt

- Kernel Live Patching ist jetzt aktiviert. Live-Patching ermöglicht es Kunden, Sicherheitslücken und kritische Bug-Patches auf einen laufenden Linux-Kernel anzuwenden, ohne Neustarts oder Unterbrechungen laufender Anwendungen.
  - Bitte beachten Sie, dass die Live-Patching-Unterstützung für Kernel 5.10.192 am 30.11.23 endet.

- Weitere Informationen finden Sie in den offiziellen Dokumenten unter 2-live-patching.html AWS https://docs.aws.amazon.com/AWSEC2/ latest/UserGuide/al

Version 74.0

Datum der Veröffentlichung: 2023-07-19

Aktualisiert

- Von 2.12 auf 2.13 aktualisiert TensorFlow
  - Horovod wurde in dieser Version aus der Conda-Umgebung entfernt. Einzelheiten zur Installation von Horovod finden Sie im Hinweis.

Version 7.3.1

Datum der Veröffentlichung: 2023-06-12

Aktualisiert

- Von 2.0.0 auf 2.0.1 aktualisiert PyTorch

Erste Schritte mit DLAMI

Dieser Leitfaden enthält Tipps zur Auswahl des für Sie geeigneten DLAMI, zur Auswahl eines Instance-Typs, der zu Ihrem Anwendungsfall und Budget passt, und Weitere Informationen über DLAMI beschreibt benutzerdefinierte Setups, die von Interesse sein könnten.

Wenn Sie Amazon noch nicht verwenden AWS oder verwenden EC2, beginnen Sie mit demDeep-Learning-AMI mit Conda. Wenn Sie mit Amazon EC2 und anderen AWS Diensten wie Amazon EMR, Amazon EFS oder Amazon S3 vertraut sind und daran interessiert sind, diese Services für Projekte zu integrieren, die verteilte Schulungen oder Inferenzen benötigen, dann schauen Sie Weitere Informationen über DLAMI nach, ob einer für Ihren Anwendungsfall geeignet ist.

Wir empfehlen Ihnen, sich zuerst über Einen DLAMI wählen zu informieren und so den am besten geeigneten Instance-Typ zu bestimmen.

Nächster Schritt

Einen DLAMI wählen

Einen DLAMI wählen

Wir bieten eine Reihe von DLAMI-Optionen an, wie in den GPU-DLAMI-Versionshinweisen erwähnt. Um Ihnen bei der Auswahl des richtigen DLAMI für Ihren Anwendungsfall zu helfen, gruppieren wir Images nach dem Hardwaretyp oder der Funktionalität, für die sie entwickelt wurden. Unsere Gruppierungen auf oberster Ebene sind:

- DLAMI-Typ: Basis, Einzelframework, Multi-Framework (Conda DLAMI)
- Rechenarchitektur: x86-basiertes, ARM64-basiertes Graviton AWS
- Prozessortyp: GPU, CPU, Inferentia, Trainium
- SDK : CUDA, Neuron AWS
- Betriebssystem: Amazon Linux, Ubuntu

Die restlichen Themen in diesem Handbuch helfen Ihnen dabei, Sie weiter zu informieren und näher zu erläutern.

Themen

- CUDA-Installationen und Framework-Bindungen
- Deep-Learning-Basis-AMI
- Deep-Learning-AMI mit Conda
- DLAMI-Architekturoptionen
- DLAMI-Betriebssystemoptionen

Nächstes Thema

Deep-Learning-AMI mit Conda

CUDA-Installationen und Framework-Bindungen

Deep Learning ist zwar ziemlich modern, aber jedes Framework bietet „stabile“ Versionen. Diese stabilen Versionen funktionieren unter Umständen nicht mit den neuesten CUDA- oder cuDNN-Implementierungen und -Funktionen. Ihr Anwendungsfall und die Funktionen, die Sie benötigen, können Ihnen bei der Auswahl eines Frameworks helfen. Wenn Sie sich nicht sicher sind, verwenden Sie das neueste Deep Learning AMI mit Conda. Es enthält offizielle `pip` Binärdateien für alle Frameworks mit CUDA, wobei die neueste Version verwendet wird, die von jedem Framework unterstützt wird. Wenn Sie die neuesten Versionen benötigen und Ihre Deep-Learning-Umgebung anpassen möchten, verwenden Sie das Deep Learning Base AMI.

Weitere Informationen finden Sie in unserem Handbuch in Stable versus Release-Kandidaten.

Wählen Sie ein DLAMI mit CUDA

Die Deep-Learning-Basis-AMI hat alle verfügbaren CUDA-Versionsserien

Die Deep-Learning-AMI mit Conda hat alle verfügbaren CUDA-Versionsserien

> **Note**
>
> Wir schließen die Umgebungen CNTK MXNet, Caffe, Caffe2, Theano, Chainer oder Keras Conda nicht mehr in die ein. AWS Deep Learning AMIs

Spezifische Framework-Versionsnummern finden Sie in der AMIs Versionshinweise zu Deep Learning

Wählen Sie diesen DLAMI-Typ oder erfahren Sie DLAMIs mit der Option Next Up mehr über die Unterschiede.

Wählen Sie eine der CUDA-Versionen aus und lesen Sie die vollständige Liste der Versionen, für DLAMIs die diese Version verfügbar ist, im Anhang, oder erfahren Sie mehr über die verschiedenen Versionen DLAMIs mit der Option Next Up.

Nächstes Thema

Deep-Learning-Basis-AMI

Verwandte Themen

- Anweisungen zum Wechseln zwischen CUDA-Versionen finden Sie im Verwenden des Deep Learning Base AMI-Tutorial.

Deep-Learning-Basis-AMI

Das Deep Learning Base AMI ist wie eine leere Leinwand für Deep Learning. Es enthält alles, was Sie bis zur Installation eines bestimmten Frameworks benötigen, und bietet eine Auswahl an CUDA-Versionen Ihrer Wahl.

Warum sollten Sie sich für das Base DLAMI entscheiden

Diese AMI-Gruppe ist nützlich für Projektbeteiligte, die ein Deep-Learning-Projekt aufspalten und das aktuelle erstellen möchten. Es ist für Entwickler gedacht, die ihre eigene Umgebung mit der Gewissheit bereitstellen möchten, dass die neueste NVIDIA-Software installiert ist und funktioniert, damit sie ungestört auswählen können, welche Frameworks und Versionen sie installieren möchten.

Wählen Sie diesen DLAMI-Typ oder erfahren Sie DLAMIs mit der Option Next Up mehr über die Unterschiede.

Nächstes Thema

DLAMI mit Conda

Verwandte Themen

- Verwenden des Deep Learning Base AMI

Deep-Learning-AMI mit Conda

Das Conda DLAMI verwendet conda virtuelle Umgebungen, sie sind entweder in mehreren Frameworks oder in einem einzigen Framework vorhanden. DLAMIs Diese Umgebungen sind so konfiguriert, dass die verschiedenen Framework-Installationen getrennt bleiben und der Wechsel zwischen Frameworks rationalisiert wird. Dies ist ideal, um mit allen Frameworks zu lernen und zu experimentieren, die das DLAMI zu bieten hat. Die meisten Benutzer finden, dass das neue Deep Learning AMI mit Conda perfekt für sie ist.

Sie werden häufig mit den neuesten Versionen der Frameworks aktualisiert und verfügen über die neuesten GPU-Treiber und -Software. Sie werden in den meisten Dokumenten allgemein als „die AWS Deep Learning AMIs “ bezeichnet. Diese DLAMIs unterstützen die Betriebssysteme Ubuntu 20.04, Ubuntu 22.04, Amazon Linux 2 und Amazon Linux 2023. Die Unterstützung von Betriebssystemen hängt von der Unterstützung durch das vorgelagerte Betriebssystem ab.

Stable versus Release-Kandidaten

Die Conda AMIs verwenden optimierte Binärdateien der neuesten formellen Versionen aus jedem Framework. Release-Kandidaten und experimentelle Funktionen sind nicht zu erwarten. Die Optimierungen hängen von der Unterstützung des Frameworks für Beschleunigungstechnologien wie Intels MKL DNN ab, wodurch das Training und die Inferenz für C5- und C4-CPU-Instance-Typen beschleunigt werden. Die Binärdateien wurden außerdem so kompiliert, dass sie erweiterte Intel-Befehlssätze unterstützen, einschließlich, aber nicht beschränkt auf AVX, AVX-2, .1 und .2. SSE4 SSE4 Diese beschleunigen Vektor- und Gleitkommaoperationen auf Intel-CPU-Architekturen. Darüber hinaus werden CUDA und cuDNN für GPU-Instance-Typen mit der Version aktualisiert, die die neueste offizielle Version unterstützt.

Das Deep Learning AMI mit Conda installiert bei der ersten Aktivierung des Frameworks automatisch die optimierteste Version des Frameworks für Ihre EC2 Amazon-Instance. Weitere Informationen finden Sie unter Verwenden des Deep Learning-AMI mit Conda.

Wenn Sie aus dem Quellcode installieren und dabei benutzerdefinierte oder optimierte Build-Optionen verwenden möchten, Deep-Learning-Basis-AMI ist s möglicherweise die bessere Option für Sie.

Beendigung von Python 2

Die Python-Open-Source-Community hat die Unterstützung für Python 2 am 1.Januar 2020 offiziell beendet. Die TensorFlow PyTorch AND-Community hat angekündigt, dass die Versionen TensorFlow

2.1 und PyTorch 1.4 die letzten sind, die Python 2 unterstützen. Frühere Versionen des DLAMI (v26, v25 usw.), die Python 2 Conda-Umgebungen enthalten, sind weiterhin verfügbar. Wir stellen jedoch Updates für die Python 2 Conda-Umgebungen auf zuvor veröffentlichten DLAMI-Versionen nur bereit, wenn von der Open-Source-Community Sicherheitsupdates für diese Versionen veröffentlicht wurden. DLAMI-Versionen mit den neuesten Versionen der PyTorch Frameworks TensorFlow und enthalten nicht die Python 2 Conda-Umgebungen.

CUDA Support

Spezifische CUDA-Versionsnummern finden Sie in den GPU-DLAMI-Versionshinweisen.

Nächstes Thema

DLAMI-Architekturoptionen

Verwandte Themen

- Ein Tutorial zur Verwendung eines Deep Learning-AMI mit Conda finden Sie im Verwenden des Deep Learning-AMI mit Conda Tutorial.

DLAMI-Architekturoptionen

AWS Deep Learning AMIss werden entweder mit x86-basierten oder ARM64-basierten Graviton2-Architekturen angeboten.AWS

Hinweise zu den ersten Schritten mit dem ARM64 GPU-DLAMI finden Sie unter. Das ARM64 DLAMI Weitere Informationen zu verfügbaren Instance-Typen finden Sie unter. Auswahl eines DLAMI-Instanztyps

Nächstes Thema

DLAMI-Betriebssystemoptionen

DLAMI-Betriebssystemoptionen

DLAMIs werden in den folgenden Betriebssystemen angeboten.

- Amazon Linux 2
- Amazon Linux 2023
- Ubuntu 20.04

- Ubuntu 22.04

Ältere Versionen von Betriebssystemen sind als veraltet DLAMIs verfügbar. Weitere Informationen zur Veralterung von DLAMI finden Sie unter Veraltete Versionen für DLAMI

Bevor Sie sich für ein DLAMI entscheiden, sollten Sie abwägen, welchen Instance-Typ Sie benötigen, und Ihre AWS Region identifizieren.

Nächstes Thema

Auswahl eines DLAMI-Instanztyps

Auswahl eines DLAMI-Instanztyps

Generell sollten Sie bei der Auswahl eines Instance-Typs für ein DLAMI Folgendes beachten.

- Wenn Sie mit Deep Learning noch nicht vertraut sind, könnte eine Instanz mit einer einzelnen GPU Ihren Anforderungen entsprechen.
- Wenn Sie budgetbewusst sind, können Sie Instances verwenden, die nur auf CPUs basieren.
- Wenn Sie die hohe Leistung und Kosteneffizienz für die Deep-Learning-Modellinferenz optimieren möchten, können Sie Instances mit Inferentia-Chips verwenden. AWS
- Wenn Sie nach einer leistungsstarken GPU-Instanz mit einer ARM64-basierten CPU-Architektur suchen, können Sie den Instance-Typ G5g verwenden.
- Wenn Sie daran interessiert sind, ein vortrainiertes Modell für Inferenz und Vorhersagen auszuführen, können Sie Ihrer Amazon-Instance eine Amazon Elastic Inference hinzufügen. EC2 Amazon Elastic Inference bietet Ihnen Zugriff auf einen Beschleuniger mit einem Bruchteil einer GPU.
- Für umfangreiche Inferenzdienste könnte eine einzelne CPU-Instance mit viel Speicher oder ein Cluster solcher Instances die bessere Lösung sein.
- Wenn Sie ein großes Modell mit vielen Daten oder einer hohen Batchgröße verwenden, benötigen Sie eine größere Instanz mit mehr Speicher. Sie können Ihr Modell auch auf einen Cluster von verteilen GPUs. Möglicherweise stellt auch eine Instance mit weniger Speicher eine bessere Lösung für Sie dar, wenn Sie Ihre Batchgröße verringern. Dies kann sich auf die Genauigkeit und Schulungsgeschwindigkeit auswirken.
- Wenn Sie daran interessiert sind, Anwendungen für maschinelles Lernen mithilfe der NVIDIA Collective Communications Library (NCCL) auszuführen, die ein hohes Maß an Kommunikation

zwischen den Knoten in großem Umfang erfordern, sollten Sie den Elastic Fabric Adapter (EFA) verwenden.

Weitere Informationen zu Instances finden Sie unter Instance-Typen und . EC2

Die folgenden Themen enthalten Informationen zu Überlegungen zu Instanztypen.

> **Important**
>
> Deep Learning AMIs umfasst Treiber, Software oder Toolkits, die von der NVIDIA Corporation entwickelt wurden, im Eigentum der NVIDIA Corporation stehen oder von ihr bereitgestellt werden. Sie erklären sich damit einverstanden, diese NVIDIA-Treiber, -Software oder Toolkits nur auf EC2 Amazon-Instances zu verwenden, die NVIDIA-Hardware enthalten.

Themen

- Preise für das DLAMI
- Verfügbarkeit in der DLAMI Region
- Empfohlene GPU-Instances
- Empfohlene CPU-Instances
- Empfohlene Inferenzinstanzen
- Empfohlene Trainium-Instances

Preise für das DLAMI

Die im DLAMI enthaltenen Deep-Learning-Frameworks sind kostenlos und verfügen jeweils über eigene Open-Source-Lizenzen. Obwohl die im DLAMI enthaltene Software kostenlos ist, müssen Sie dennoch für die zugrunde liegende EC2 Amazon-Instance-Hardware bezahlen.

Einige EC2 Amazon-Instance-Typen sind als kostenlos gekennzeichnet. Es ist möglich, das DLAMI auf einer dieser kostenlosen Instanzen auszuführen. Das bedeutet, dass die Nutzung des DLAMI völlig kostenlos ist, wenn Sie nur die Kapazität dieser Instanz nutzen. Wenn Sie eine leistungsfähigere Instanz mit mehr CPU-Kernen, mehr Festplattenspeicher, mehr RAM oder einer oder mehreren benötigen, benötigen Sie eine Instanz GPUs, die nicht zur Instance-Klasse des kostenlosen Tiers gehört.

Weitere Informationen zur Instance-Auswahl und Preisgestaltung finden Sie unter EC2 Amazon-Preise.

Verfügbarkeit in der DLAMI Region

Jede Region unterstützt eine andere Reihe von Instance-Typen, und oft fallen für einen Instance-Typ in verschiedenen Regionen leicht unterschiedliche Kosten an. DLAMIs sind nicht in jeder Region verfügbar, aber es ist möglich, in die Region Ihrer Wahl DLAMIs zu kopieren. Weitere Informationen finden Sie unter AMI kopieren. Beachten Sie die Auswahlliste für Regionen und achten Sie darauf, dass Sie eine Region auswählen, die sich in Ihrer Nähe oder in der Nähe Ihrer Kunden befindet. Wenn Sie planen, mehr als ein DLAMI zu verwenden und möglicherweise einen Cluster zu erstellen, stellen Sie sicher, dass Sie dieselbe Region für alle Knoten im Cluster verwenden.

Weitere Informationen zu Regionen finden Sie unter — Amazon EC2 Service Endpoints.

Nächstes Thema

Empfohlene GPU-Instances

Empfohlene GPU-Instances

Für die meisten Deep-Learning-Zwecke empfehlen wir eine GPU-Instanz. Das Trainieren neuer Modelle ist auf einer GPU-Instanz schneller als auf einer CPU-Instanz. Sie können sublinear skalieren, wenn Sie mehrere GPU-Instanzen haben oder wenn Sie verteiltes Training auf viele Instanzen mit verwenden. GPUs

Die folgenden Instance-Typen unterstützen das DLAMI. Informationen zu den Optionen für GPU-Instanztypen und deren Verwendung finden Sie unter und wählen Sie Accelerated Computing aus.

> **Note**
>
> Die Größe Ihres Modells sollte bei der Auswahl einer Instanz eine Rolle spielen. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instanz überschreitet, wählen Sie einen anderen Instance-Typ mit ausreichend Arbeitsspeicher für Ihre Anwendung.

- Amazon EC2 P6-Instances verfügen über bis zu 8 NVIDIA Blackwell B200. GPUs
- Amazon EC2 P5e-Instances verfügen über bis zu 8 NVIDIA Tesla H200. GPUs
- Amazon EC2 P5-Instances verfügen über bis zu 8 NVIDIA Tesla GPUs H100.

- Amazon EC2 P4-Instances verfügen über bis zu 8 NVIDIA Tesla GPUs A100.
- Amazon EC2 P3-Instances verfügen über bis zu 8 NVIDIA Tesla GPUs V100.
- Amazon EC2 G3-Instances verfügen über bis zu 4 NVIDIA Tesla GPUs M60.
- Amazon EC2 G4-Instances verfügen über bis zu 4 NVIDIA GPUs T4.
- Amazon EC2 G5-Instances verfügen über bis zu 8 NVIDIA GPUs A10G.
- Amazon EC2 G6-Instances verfügen über bis zu 8 NVIDIA GPUs L4.
- Amazon EC2 G6e-Instances verfügen über bis zu 8 NVIDIA L40S Tensor Core. GPUs
- Amazon EC2 G5G-Instances verfügen über ARM64-basierte AWS Graviton2-Prozessoren.

DLAMI-Instanzen bieten Tools zur Überwachung und Optimierung Ihrer GPU-Prozesse. Weitere Informationen zur Überwachung Ihrer GPU-Prozesse finden Sie unter. GPU-Überwachung und -Optimierung

Spezifische Tutorials zur Arbeit mit G5G-Instances finden Sie unterDas ARM64 DLAMI.

Nächstes Thema

Empfohlene CPU-Instances

Empfohlene CPU-Instances

Egal, ob Sie nur ein begrenztes Budget haben, etwas über Deep Learning lernen oder einfach nur einen Vorrausage-Service betreiben möchten, Sie haben viele günstige Optionen in der CPU-Kategorie. Einige Frameworks nutzen das MKL DNN von Intel, das das Training und die Inferenz für C5-CPU-Instance-Typen (nicht in allen Regionen verfügbar) beschleunigt. Informationen zu CPU-Instance-Typen finden Sie unter Instance-Typen und . Wählen Sie dort EC2 Compute Optimized aus.

> **Note**
>
> Die Größe Ihres Modells sollte ein Faktor bei der Auswahl einer Instanz sein. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instanz überschreitet, wählen Sie einen anderen Instance-Typ mit ausreichend Arbeitsspeicher für Ihre Anwendung.

- Amazon EC2 C5-Instances haben bis zu 72 Intel v. CPUs C5-Instances zeichnen sich durch wissenschaftliche Modellierung, Stapelverarbeitung, verteilte Analytik, Hochleistungsrechnen (HPC) sowie Inferenz für maschinelles Lernen und Deep Learning aus.

Nächstes Thema

Empfohlene Inferenzinstanzen

Empfohlene Inferenzinstanzen

AWS Inferentia-Instances sind so konzipiert, dass sie eine hohe Leistung und Kosteneffizienz für Inferenz-Workloads mit Deep-Learning-Modellen bieten. Insbesondere Inf2-Instance-Typen verwenden AWS Inferentia-Chips und das AWS Neuron SDK, das in beliebte Frameworks für maschinelles Lernen wie und integriert ist. TensorFlow PyTorch

Kunden können Inf2-Instances verwenden, um umfangreiche Inferenzanwendungen für maschinelles Lernen wie Suche, Empfehlungsmaschinen, Computer Vision, Spracherkennung, Verarbeitung natürlicher Sprache, Personalisierung und Betrugserkennung zu den niedrigsten Kosten in der Cloud auszuführen.

> **Note**
>
> Die Größe Ihres Modells sollte ein Faktor bei der Auswahl einer Instanz sein. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instanz überschreitet, wählen Sie einen anderen Instance-Typ mit ausreichend Arbeitsspeicher für Ihre Anwendung.

- Amazon EC2 Inf2-Instances verfügen über bis zu 16 AWS Inferentia-Chips und einen Netzwerkdurchsatz von 100 Gbit/s.

Weitere Informationen zu den ersten Schritten mit AWS Inferentia finden Sie unter. DLAMIs Der AWS Inferentia-Chip mit DLAMI

Nächstes Thema

Empfohlene Trainium-Instances

Empfohlene Trainium-Instances

AWS Trainium-Instances sind darauf ausgelegt, hohe Leistung und Kosteneffizienz für Inferenz-Workloads mit Deep-Learning-Modellen zu bieten. Insbesondere Trn1-Instance-Typen verwenden AWS Trainium-Chips und das AWS Neuron SDK, das in beliebte Frameworks für maschinelles Lernen wie und integriert ist. TensorFlow PyTorch

Kunden können Trn1-Instances verwenden, um umfangreiche Inferenzanwendungen für maschinelles Lernen wie Suche, Empfehlungsmaschinen, Computer Vision, Spracherkennung, Verarbeitung natürlicher Sprache, Personalisierung und Betrugserkennung zu den niedrigsten Kosten in der Cloud auszuführen.

> **Note**
>
> Die Größe Ihres Modells sollte ein Faktor bei der Auswahl einer Instanz sein. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instanz überschreitet, wählen Sie einen anderen Instance-Typ mit ausreichend Arbeitsspeicher für Ihre Anwendung.

- Amazon EC2 Trn1-Instances verfügen über bis zu 16 AWS Trainium-Chips und einen Netzwerkdurchsatz von 100 Gbit/s.

Einrichtung einer DLAMI-Instanz

Nachdem Sie ein DLAMI und einen Amazon Elastic Compute Cloud (Amazon EC2) -Instanztyp ausgewählt haben, den Sie verwenden möchten, können Sie Ihre neue DLAMI-Instanz einrichten.

Wenn Sie noch kein DLAMI und keinen EC2 Instanztyp ausgewählt haben, finden Sie weitere Informationen unter. Erste Schritte mit DLAMI

Themen

- Die ID eines DLAMI ermitteln
- Eine DLAMI-Instanz starten
- Verbindung zu einer DLAMI-Instanz herstellen
- Einrichtung eines Jupyter Notebook-Servers auf einer DLAMI-Instanz
- Bereinigen einer DLAMI-Instanz

Die ID eines DLAMI ermitteln

Jedes DLAMI hat eine eindeutige Kennung (ID). Wenn Sie eine DLAMI-Instance über die EC2 Amazon-Konsole starten, können Sie optional die DLAMI-ID verwenden, um nach dem DLAMI zu suchen, das Sie verwenden möchten. Wenn Sie eine DLAMI-Instanz mit AWS Command Line Interface (AWS CLI) starten, ist diese ID erforderlich.

Sie können die ID für das DLAMI Ihrer Wahl finden, indem Sie einen AWS CLI Befehl für Amazon EC2 oder Parameter Store verwenden, eine Funktion von. AWS Systems Manager Anweisungen zur Installation und Konfiguration von finden Sie unter Erste Schritte mit dem AWS CLI im AWS Command Line Interface Benutzerhandbuch. AWS CLI

Using Parameter Store

> Um eine DLAMI-ID zu finden mit ssm get-parameter
>
> Im folgenden ssm get-parameterBefehl lautet das Format des Parameternamens für die `--name` Option. */aws/service/deeplearning/ami/$architecture/$ami_type/latest/ami-id* In diesem Namensformat *`architecture`* kann entweder **`x86_64`** oder sein**`arm64`**. Geben Sie das an, *`ami_type`* indem Sie den DLAMI-Namen verwenden und die Schlüsselwörter „deep“, „learning“ und „ami“ entfernen. Der AMI-Name befindet sich in AMIs Versionshinweise zu Deep Learning.

> **Important**
>
> Um diesen Befehl verwenden zu können, muss der AWS Identity and Access Management (IAM-) Principal, den Sie verwenden, über die `ssm:GetParameter` entsprechende Berechtigung verfügen. Weitere Informationen zu IAM-Prinzipalen finden Sie im Abschnitt Zusätzliche Ressourcen unter IAM-Rollen im IAM-Benutzerhandbuch.

- ```
  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
  ```

  Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

  ```
  ami-09ee1a996ac214ce7
  ```

  > **Tip**
  >
  > Für einige derzeit unterstützte DLAMI-Frameworks finden Sie spezifischere ssm get-parameter Beispielbefehle unter. AMIs Versionshinweise zu Deep Learning Wählen Sie den Link zu den Versionshinweisen des von Ihnen ausgewählten DLAMI und suchen Sie dann in den Versionshinweisen nach der entsprechenden ID-Abfrage.

Using Amazon EC2 CLI

Um eine DLAMI-ID zu finden mit ec2 describe-images

Geben Sie im folgenden ec2 describe-imagesBefehl für den Wert des Filters `Name=name` den DLAMI-Namen ein. Sie können eine Release-Version für ein bestimmtes Framework angeben, oder Sie können die neueste Version abrufen, indem Sie die Versionsnummer durch ein Fragezeichen (?) ersetzen.

- ```
  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
  ```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
ami-09ee1a996ac214ce7
```

> **Tip**
>
> Einen ec2 describe-images Beispielbefehl, der spezifisch für das DLAMI Ihrer Wahl ist, finden Sie unter. AMIs Versionshinweise zu Deep Learning Wählen Sie den Link zu den Versionshinweisen des von Ihnen ausgewählten DLAMI und suchen Sie dann in den Versionshinweisen nach der entsprechenden ID-Abfrage.

Nächster Schritt

Eine DLAMI-Instanz starten

Eine DLAMI-Instanz starten

Nachdem Sie die ID des DLAMI gefunden haben, das Sie zum Starten einer DLAMI-Instanz verwenden möchten, können Sie die Instance starten. Um es zu starten, können Sie entweder die EC2 Amazon-Konsole oder die AWS Command Line Interface (AWS CLI) verwenden.

> **Note**
>
> Für diese exemplarische Vorgehensweise verweisen wir möglicherweise speziell auf das Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04). Auch wenn Sie ein anderes DLAMI auswählen, sollten Sie in der Lage sein, dieser Anleitung zu folgen.

EC2 console

> **Note**
>
> Um Anwendungen für Hochleistungscomputing (HPC) und maschinelles Lernen zu beschleunigen, können Sie Ihre DLAMI-Instanz mit einem Elastic Fabric Adapter (EFA) starten. Spezifische Anweisungen finden Sie unter. Starten einer Instance mit EFA AWS Deep Learning AMIs

1. Öffnen Sie die EC2 Konsole.
2. Notieren Sie sich Ihre aktuelle AWS-Region Position in der obersten Navigationsleiste. Wenn dies nicht Ihre gewünschte Region ist, ändern Sie diese Option, bevor Sie fortfahren. Weitere Informationen finden Sie unter der EC2 Amazon-Servicendpunkte in der Allgemeine Amazon Web Services-Referenz.
3. Wählen Sie Launch Instance aus.
4. Geben Sie einen Namen für Ihre Instanz ein und wählen Sie das für Sie passende DLAMI aus.

   a. Suchen Sie in „Mein“ nach einem vorhandenen DLAMI AMIs oder wählen Sie „Schnellstart“.

   b. Suche nach DLAMI-ID. Durchsuchen Sie die Optionen und wählen Sie dann Ihre Wahl aus.
5. Wählen Sie einen Instance-Type. Die empfohlenen Instanzfamilien für Ihr DLAMI finden Sie unter. AMIs Versionshinweise zu Deep Learning Allgemeine Empfehlungen zu DLAMI-Instanztypen finden Sie unter. Auswahl eines DLAMI-Instanztyps
6. Wählen Sie Launch Instance aus.

AWS CLI

- Um das verwenden zu können AWS CLI, benötigen Sie die ID des DLAMIs, das Sie verwenden möchten, den EC2 Instanztyp AWS-Region und Ihre Sicherheitstoken-Informationen. Anschließend können Sie die Instance mit dem ec2 run-instances AWS CLI Befehl starten.

  Anweisungen zur Installation und Konfiguration von finden Sie unter Erste Schritte mit dem AWS CLI im AWS Command Line Interface Benutzerhandbuch. AWS CLI Weitere Informationen, einschließlich Befehlsbeispielen, finden Sie unter Starten, Auflisten und Schließen von EC2 Amazon-Instances für die AWS CLI.

Nachdem Sie Ihre Instance entweder über die EC2 Amazon-Konsole oder gestartet haben AWS CLI, warten Sie, bis die Instance bereit ist. Dieser Vorgang dauert einige Minuten. Sie können den Status der Instance in der EC2 Amazon-Konsole überprüfen. Weitere Informationen finden Sie unter Statuschecks für EC2 Amazon-Instances im EC2 Amazon-Benutzerhandbuch.

Nächster Schritt

Verbindung zu einer DLAMI-Instanz herstellen

Verbindung zu einer DLAMI-Instanz herstellen

Nachdem Sie eine DLAMI-Instanz gestartet haben und die Instanz läuft, können Sie über SSH von einem Client (Windows, macOS oder Linux) aus eine Verbindung zu ihr herstellen. Anweisungen finden Sie im EC2 Amazon-Benutzerhandbuch unter Herstellen einer Connect zu Ihrer Linux-Instance mithilfe von SSH.

Halten Sie eine Kopie des SSH-Login-Befehls bereit, falls Sie nach der Anmeldung einen Jupyter Notebook-Server einrichten möchten. Um eine Verbindung zur Jupyter-Webseite herzustellen, verwenden Sie eine Variante dieses Befehls.

Nächster Schritt

Einrichtung eines Jupyter Notebook-Servers auf einer DLAMI-Instanz

Einrichtung eines Jupyter Notebook-Servers auf einer DLAMI-Instanz

Mit einem Jupyter Notebook-Server können Sie Jupyter-Notebooks von Ihrer DLAMI-Instanz aus erstellen und ausführen. Mit Jupyter-Notebooks können Sie Machine-Learning-Experimente (ML) für Training und Inferenz durchführen, während Sie die AWS Infrastruktur nutzen und auf die in der DLAMI integrierten Pakete zugreifen. Weitere Informationen zu Jupyter-Notebooks finden Sie unter The Jupyter Notebook auf der Website Jupyter User Documentation.

Um einen Jupyter Notebook-Server einzurichten, müssen Sie:

- Konfigurieren Sie den Jupyter Notebook-Server auf Ihrer DLAMI-Instanz.
- Konfigurieren Sie Ihren Client so, dass er eine Verbindung zum Jupyter Notebook-Server herstellt. Wir stellen Konfigurationsanweisungen für Windows-, MacOS- und Linux-Clients zur Verfügung.
- Testen Sie das Setup, indem Sie sich beim Jupyter Notebook-Server anmelden.

Folgen Sie den Anweisungen in den folgenden Themen, um diese Schritte abzuschließen. Nachdem Sie einen Jupyter Notebook-Server eingerichtet haben, können Sie die Beispiel-Notebook-Tutorials

ausführen, die im Lieferumfang von enthalten sind. DLAMIs Weitere Informationen finden Sie unter Ausführen von Jupyter-Notebook-Tutorials.

Themen

- Sicherung des Jupyter Notebook-Servers auf einer DLAMI-Instanz
- Den Jupyter Notebook-Server auf einer DLAMI-Instanz starten
- Einen Client mit dem Jupyter Notebook-Server auf einer DLAMI-Instanz verbinden
- Anmeldung am Jupyter Notebook-Server auf einer DLAMI-Instanz

Sicherung des Jupyter Notebook-Servers auf einer DLAMI-Instanz

Um Ihren Jupyter Notebook-Server zu schützen, empfehlen wir, ein Passwort einzurichten und ein SSL-Zertifikat für den Server zu erstellen. Um ein Passwort und SSL zu konfigurieren, stellen Sie zunächst eine Verbindung zu Ihrer DLAMI-Instanz her und folgen Sie dann diesen Anweisungen.

Um den Jupyter Notebook-Server zu sichern

1. Jupyter bietet ein Passwort-Dienstprogramm. Führen Sie den folgenden Befehl aus und geben Sie Ihr bevorzugtes Passwort ein, wenn Sie dazu aufgefordert werden.

   ```
   $ jupyter notebook password
   ```

   Das Ergebnis sieht etwa folgendermaßen aus:

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

2. Erstellen Sie ein selbstsigniertes SSL-Zertifikat Folgen Sie den Anweisungen, um Ihren Ortsnamen entsprechend einzutragen. Geben Sie `.` ein, wenn Sie eine Eingabeaufforderung leer lassen möchten. Ihre Antworten wirken sich nicht auf die Funktionalität des Zertifikats aus.

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

> **Note**
>
> Möglicherweise möchten Sie ein reguläres SSL-Zertifikat erstellen, das von einem Drittanbieter signiert ist und nicht dazu führt, dass der Browser Ihnen eine Sicherheitswarnung ausgibt. Dieser Prozess ist wesentlich aufwendiger. Weitere Informationen finden Sie unter Sichern eines Notebook-Servers in der Jupyter Notebook-Benutzerdokumentation.

Nächster Schritt

Den Jupyter Notebook-Server auf einer DLAMI-Instanz starten

Den Jupyter Notebook-Server auf einer DLAMI-Instanz starten

Nachdem Sie Ihren Jupyter Notebook-Server mit einem Passwort und SSL gesichert haben, können Sie den Server starten. Melden Sie sich bei Ihrer DLAMI-Instanz an und führen Sie den folgenden Befehl aus, der das zuvor erstellte SSL-Zertifikat verwendet.

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

Wenn der Server gestartet ist, können Sie sich von Ihrem Client-Computer aus über einen SSH-Tunnel mit ihm verbinden. Wenn der Server ausgeführt wird, sehen Sie eine Ausgabe von Jupyter. Diese bestätigt, dass der Server ausgeführt wird. Ignorieren Sie an dieser Stelle den Hinweis, dass Sie über eine lokale Host-URL auf den Server zugreifen können, da dies erst funktioniert, wenn Sie den Tunnel erstellt haben.

> **Note**
>
> Jupyter übernimmt das Wechseln von Umgebungen, wenn Sie die Frameworks mithilfe der Jupyter-Web-Schnittstelle wechseln. Weitere Informationen finden Sie unter Wechseln von Umgebungen mit Jupyter.

Nächster Schritt

Einen Client mit dem Jupyter Notebook-Server auf einer DLAMI-Instanz verbinden

Einen Client mit dem Jupyter Notebook-Server auf einer DLAMI-Instanz verbinden

Nachdem Sie den Jupyter Notebook-Server auf Ihrer DLAMI-Instanz gestartet haben, konfigurieren Sie Ihren Windows-, macOS- oder Linux-Client so, dass er eine Verbindung zum Server herstellt. Wenn Sie eine Verbindung herstellen, können Sie Jupyter-Notebooks auf dem Server in Ihrem Workspace erstellen und darauf zugreifen und Ihren Deep-Learning-Code auf dem Server ausführen.

Voraussetzungen

Stellen Sie sicher, dass Sie über Folgendes verfügen, das Sie für die Einrichtung eines SSH-Tunnels benötigen:

- Der öffentliche DNS-Name Ihrer EC2 Amazon-Instance. Weitere Informationen finden Sie unter Hostnamentypen für EC2 Amazon-Instances im EC2 Amazon-Benutzerhandbuch.
- Das Schlüsselpaar für die private Schlüsseldatei. Weitere Informationen zum Zugriff auf Ihr key pair finden Sie unter EC2Amazon-Schlüsselpaare und EC2 Amazon-Instances im EC2 Amazon-Benutzerhandbuch.

Stellen Sie von einem Windows-, macOS- oder Linux-Client aus eine Verbindung her

Um von einem Windows-, macOS- oder Linux-Client aus eine Verbindung zu Ihrer DLAMI-Instanz herzustellen, folgen Sie den Anweisungen für das Betriebssystem Ihres Clients.

Windows

So stellen Sie über SSH von einem Windows-Client aus eine Verbindung zu Ihrer DLAMI-Instanz her

1. Verwenden Sie einen SSH-Client für Windows, z. B. PuTTY. Anweisungen finden Sie unter Connect zu Ihrer Linux-Instance mithilfe von PuTTY im EC2 Amazon-Benutzerhandbuch. Weitere SSH-Verbindungsoptionen finden Sie unter Mit SSH Connect zu Ihrer Linux-Instance herstellen.
2. (Optional) Erstellen Sie einen SSH-Tunnel zu einem laufenden Jupyter-Server. Installieren Sie Git Bash auf Ihrem Windows-Client und folgen Sie dann den Verbindungsanweisungen für macOS- und Linux-Clients.

macOS or Linux

So stellen Sie über SSH von einem macOS- oder Linux-Client aus eine Verbindung zu Ihrer DLAMI-Instanz her

1. Öffnen Sie ein -Terminalfenster.
2. Führen Sie den folgenden Befehl aus, um alle Anfragen auf dem lokalen Port 8888 an den Port 8888 auf Ihrer EC2 Amazon-Remoteinstanz weiterzuleiten. Aktualisieren Sie den Befehl, indem Sie den Speicherort Ihres Schlüssels für den Zugriff auf die EC2 Amazon-Instance und den öffentlichen DNS-Namen Ihrer EC2 Amazon-Instance ersetzen. Hinweis: Bei einem Amazon Linux-AMI lautet der Benutzername `ec2-user` anstelle von `ubuntu`.

   ```
   $ ssh -i ~/mykeypair.pem -N -f -L 8888:localhost:8888 ubuntu@ec2-###-##-##-
   ###.compute-1.amazonaws.com
   ```

   Dieser Befehl öffnet einen Tunnel zwischen Ihrem Client und der entfernten EC2 Amazon-Instance, auf der der Jupyter Notebook-Server ausgeführt wird.

Nächster Schritt

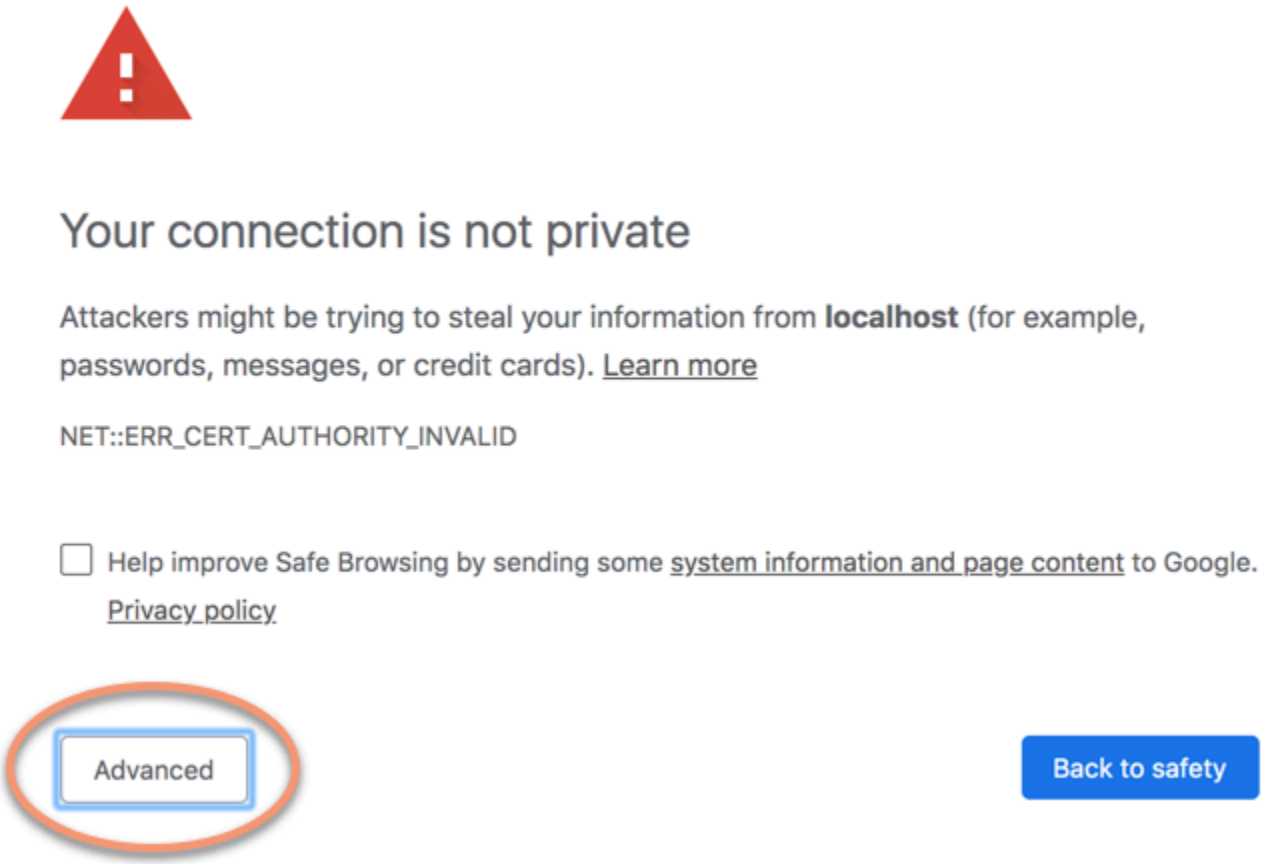
Anmeldung am Jupyter Notebook-Server auf einer DLAMI-Instanz

Anmeldung am Jupyter Notebook-Server auf einer DLAMI-Instanz

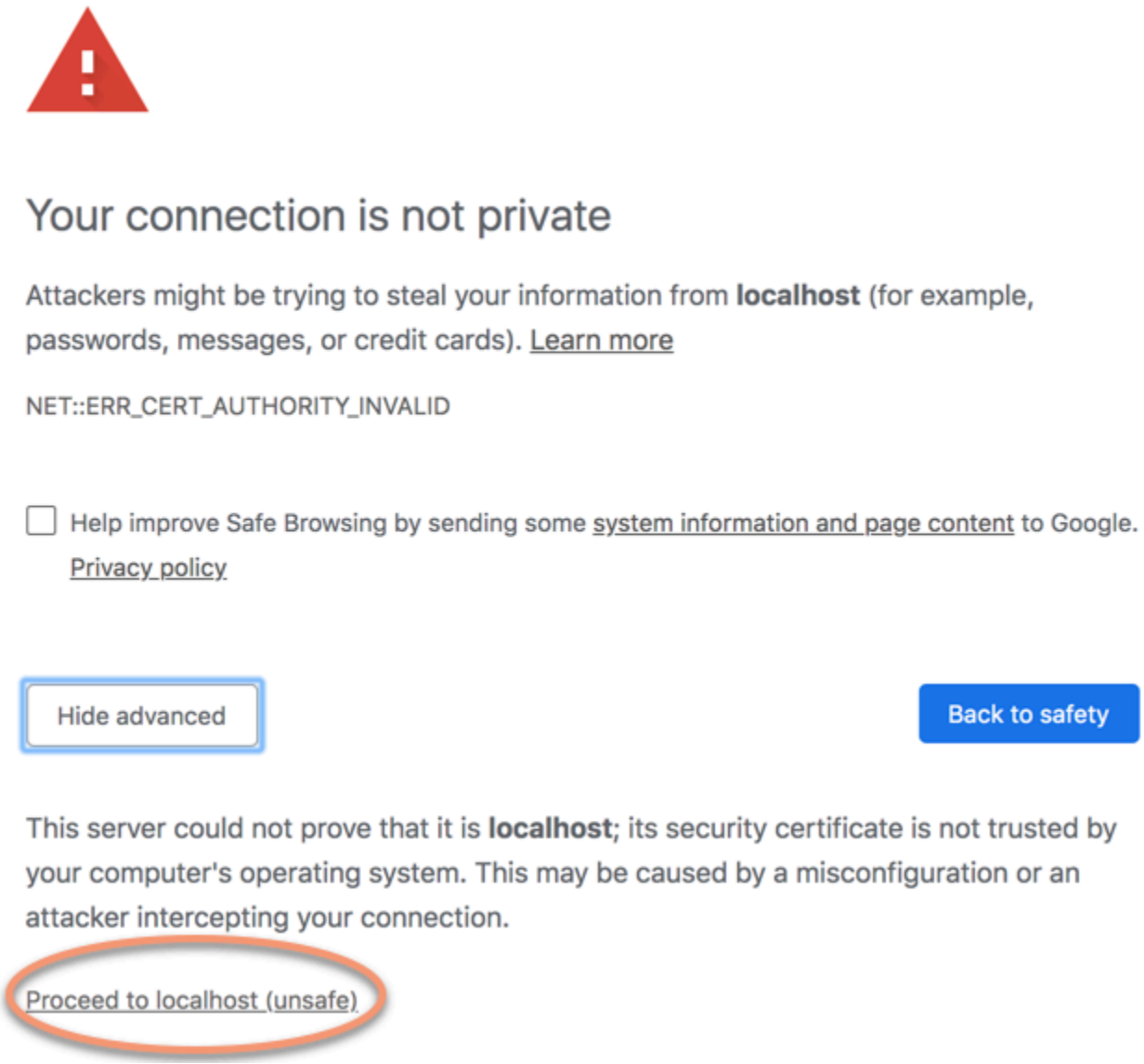
Nachdem Sie Ihren Client mit dem Jupyter Notebook-Server auf Ihrer DLAMI-Instanz verbunden haben, können Sie sich beim Server anmelden.

Um sich in Ihrem Browser beim Server anzumelden

1. Geben Sie in der Adressleiste Ihres Browsers die folgende URL ein oder klicken Sie auf diesen Link: https://localhost:8888
2. Bei einem selbstsignierten SSL-Zertifikat warnt Sie Ihr Browser und fordert Sie auf, den weiteren Besuch der Website zu vermeiden.



Da Sie dies selbst eingerichtet haben, können Sie bedenkenlos fortfahren. Abhängig von Ihrem Browser wird die Schaltfläche "Erweitert", "Details anzeigen" oder eine ähnliche angezeigt.



Klicken Sie darauf und anschließend auf den Link "Weiter zu localhost". Wenn die Verbindung erfolgreich ist, wird die Webseite des Jupyter Notebook-Servers angezeigt. An dieser Stelle werden Sie nach dem Passwort gefragt, das Sie zuvor eingerichtet haben.

Jetzt haben Sie Zugriff auf den Jupyter Notebook-Server, der auf der DLAMI-Instanz läuft. Sie können nun neue Notebooks erstellen oder die bereitgestellten Tutorials ausführen.

Bereinigen einer DLAMI-Instanz

Wenn Sie Ihre DLAMI-Instance nicht mehr benötigen, können Sie sie bei Amazon beenden oder kündigen, um EC2 unerwartete Gebühren zu vermeiden.

Wenn Sie eine Instance beenden, können Sie sie behalten und sie später wieder starten, wenn Sie sie erneut verwenden möchten. Ihre Konfigurationen, Dateien und andere nichtflüchtige Informationen werden in einem Volume auf Amazon Simple Storage Service (Amazon S3) gespeichert. Solange Ihre Instance gestoppt ist, fallen für die Aufbewahrung des Volumes S3-Gebühren an, für Rechenressourcen fallen jedoch keine Gebühren an. Wenn Sie die Instance erneut starten, wird das Speichervolume mit Ihren Daten bereitgestellt.

Wenn Sie eine Instance beenden, ist sie weg und Sie können sie nicht erneut starten. Natürlich fallen bei einer beendeten Instance keine weiteren Gebühren für die Rechenressourcen an. Ihre Daten befinden sich jedoch weiterhin auf Amazon S3, und es können weiterhin S3-Gebühren anfallen. Um alle weiteren Gebühren im Zusammenhang mit Ihrer beendeten Instance zu vermeiden, müssen Sie auch das Speichervolumen auf Amazon S3 löschen. Anweisungen finden Sie unter EC2 Amazon-Instances beenden im EC2 Amazon-Benutzerhandbuch.

Weitere Informationen zu EC2 Amazon-Instanzstatus, wie z. B. `stopped` und`terminated`, finden Sie unter Änderungen des EC2 Amazon-Instanzstatus im EC2 Amazon-Benutzerhandbuch.

Verwendung eines DLAMI

Themen

- Verwenden des Deep Learning-AMI mit Conda
- Verwenden des Deep Learning Base AMI
- Ausführen von Jupyter-Notebook-Tutorials
- Tutorials

In den folgenden Abschnitten wird beschrieben, wie das Deep Learning-AMI mit Conda verwendet werden kann, um Umgebungen zu wechseln, Beispielcode aus jedem der Frameworks auszuführen und Jupyter auszuführen, sodass Sie verschiedene Notebook-Tutorials ausprobieren können.

Verwenden des Deep Learning-AMI mit Conda

Themen

- Einführung in das Deep Learning AMI mit Conda
- Loggen Sie sich in Ihr DLAMI ein
- Starten Sie die Umgebung TensorFlow
- Wechseln Sie zur PyTorch Python-3-Umgebung
- Entfernen von Umgebungen

Einführung in das Deep Learning AMI mit Conda

Conda ist ein Open-Source-Paket- und Umgebungsverwaltungssystem, das unter Windows, MacOS und Linux ausgeführt werden kann. Conda installiert, startet und aktualisiert Pakete und deren Abhängigkeiten. Conda erstellt, speichert, lädt und wechselt Umgebungen auf Ihrem lokalen Computer.

Das Deep Learning AMI mit Conda wurde so konfiguriert, dass Sie einfach zwischen Deep-Learning-Umgebungen wechseln können. Die folgenden Anweisungen zeigen Ihnen einige grundlegende Befehle in `conda`. Sie unterstützen Sie bei der Überprüfung der korrekten Funktionsweise des grundlegenden Imports des Frameworks und der Ausführung einiger einfacher Operationen. Sie

können dann zu ausführlicheren Tutorials übergehen, die mit dem DLAMI bereitgestellt werden, oder zu den Frameworks-Beispielen, die Sie auf der Projektseite der einzelnen Frameworks finden.

Loggen Sie sich in Ihr DLAMI ein

Nachdem Sie sich an Ihrem Server angemeldet haben, sehen Sie eine Server-MOTD (Message Of The Day) mit verschiedenen Conda-Befehlen, mit denen Sie zwischen den verschiedenen Deep-Learning-Frameworks wechseln können. Unten folgt eine Beispiel-MOTD. Ihr spezifisches MOTD kann variieren, wenn neue Versionen des DLAMI veröffentlicht werden.

```
=============================================================================
        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
    =============================================================================
```

Starten Sie die Umgebung TensorFlow

> **Note**
>
> Wenn Sie Ihre erste Conda-Umgebung starten, haben Sie bitte etwas Geduld, während diese geladen wird. Das Deep Learning AMI mit Conda installiert bei der ersten Aktivierung des

Frameworks automatisch die optimierteste Version des Frameworks für Ihre EC2 Instance. Es sollten keine weiteren Verzögerungen auftreten.

1. Aktivieren Sie die TensorFlow virtuelle Umgebung für Python 3.

```
$ source activate tensorflow2_p310
```

2. Starten Sie das iPython-Terminal.

```
(tensorflow2_p310)$ ipython
```

3. Führen Sie ein schnelles TensorFlow Programm aus.

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Es sollte "Hello, Tensorflow!" angezeigt werden.

Nächstes Thema

Ausführen von Jupyter-Notebook-Tutorials

Wechseln Sie zur PyTorch Python-3-Umgebung

Wenn Sie sich noch in der IPython-Konsole befinden, verwenden Sie und bereiten Sie sich dann darauf vor`quit()`, die Umgebung zu wechseln.

- Aktivieren Sie die PyTorch virtuelle Umgebung für Python 3.

```
$ source activate pytorch_p310
```

Testen Sie etwas PyTorch Code

Um Ihre Installation zu testen, verwenden Sie Python, um PyTorch Code zu schreiben, der ein Array erstellt und ausgibt.

1. Starten Sie das iPython-Terminal.

```
(pytorch_p310)$ ipython
```

2. Importieren PyTorch.

```
import torch
```

Möglicherweise wird eine Warnmeldung zu einem Paket eines Drittanbieters angezeigt. Sie können sie ignorieren.

3. Erstellen Sie eine 5x3-Matrix mit zufällig initialisierten Elementen. Drucken Sie das Array.

```
x = torch.rand(5, 3)
print(x)
```

Überprüfen Sie das Ergebnis.

```
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]])
```

Entfernen von Umgebungen

Wenn Ihnen der Speicherplatz auf dem DLAMI ausgeht, können Sie Conda-Pakete, die Sie nicht verwenden, deinstallieren:

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

Verwenden des Deep Learning Base AMI

Verwenden des Deep Learning Base AMI

Das Basis-AMI wird mit einer grundlegenden Plattform von GPU-Treibern und Beschleunigungsbibliotheken geliefert, um Ihre eigene, angepasste Deep-Learning-Umgebung zu

implementieren. Standardmäßig ist das AMI mit einer beliebigen NVIDIA CUDA-Versionsumgebung konfiguriert. Sie können auch zwischen verschiedenen Versionen von CUDA wechseln. Weitere Informationen zur Vorgehensweise finden Sie in den folgenden Anweisungen.

CUDA-Versionen konfigurieren

Sie können die CUDA-Version überprüfen, indem Sie das NVIDIA-Programm ausführen. `nvcc`

```
nvcc --version
```

Sie können eine bestimmte CUDA-Version mit dem folgenden Bash-Befehl auswählen und überprüfen:

```
sudo rm /usr/local/cuda
sudo ln -s /usr/local/cuda-12.0 /usr/local/cuda
```

Weitere Informationen finden Sie in den Versionshinweisen zu Base DLAMI.

Ausführen von Jupyter-Notebook-Tutorials

Tutorials und Beispiele sind im Quellcode der Deep-Learning-Projekte enthalten und laufen in den meisten Fällen auf jedem DLAMI. Wenn Sie das Deep-Learning-AMI mit Conda auswählen, erhalten Sie zusätzlich einige ausgesuchte Tutorials, die bereits eingerichtet sind und sofort ausprobiert werden können.

> **Important**
>
> Um die auf dem DLAMI installierten Jupyter-Notebook-Tutorials auszuführen, müssen Sie Einrichtung eines Jupyter Notebook-Servers auf einer DLAMI-Instanz

Sobald der Jupyter-Server ausgeführt wird, können Sie die Tutorials über Ihren Webbrowser aufrufen. Wenn Sie das Deep Learning-AMI mit Conda ausführen oder Python-Umgebungen eingerichtet haben, können Sie Python-Kernel über die Jupyter-Notebook-Oberfläche wechseln. Wählen Sie den entsprechenden Kernel aus, bevor Sie versuchen, ein Tutorial für ein bestimmtes Framework auszuführen. Weitere Beispiele hierfür werden Benutzern des Deep Learning AMI mit Conda zur Verfügung gestellt.

> **Note**
>
> Viele Tutorials erfordern zusätzliche Python-Module, die möglicherweise nicht auf Ihrem DLAMI eingerichtet sind. Wenn Sie eine Fehlermeldung erhalten wie"`xyz module not found`", melden Sie sich bei der DLAMI an, aktivieren Sie die Umgebung wie oben beschrieben und installieren Sie dann die erforderlichen Module.

> **Tip**
>
> Deep-Learning-Tutorials und -Beispiele basieren oft auf einem oder mehreren. GPUs Wenn Ihr Instance-Typ keine GPU hat, müssen Sie möglicherweise Codeabschnitte des Beispiels ändern, um dieses ausführen zu können.

Navigation der installierten Tutorials

Sobald Sie beim Jupyter-Server angemeldet sind und das Verzeichnis der Tutorials sehen können (nur auf Deep Learning AMI mit Conda), werden Ihnen Ordner mit Tutorials für jeden Framework-Namen angezeigt. Wenn ein Framework nicht aufgeführt ist, sind auf Ihrem aktuellen DLAMI keine Tutorials für dieses Framework verfügbar. Klicken Sie auf den Namen des Frameworks, um die aufgeführten Tutorials zu sehen. Wählen Sie eines davon per Klick aus, wenn Sie es starten möchten.

Wenn Sie ein Notebook zum ersten Mal auf dem Deep Learning AMI mit Conda ausführen, möchte es wissen, welche Umgebung Sie verwenden möchten. Es wird eine Liste zur Auswahl bereitgestellt. Jede Umgebung hat einen Namen, der diesem Muster entspricht:

`Environment (conda_framework_python-version)`

Zum Beispiel könnten Sie sehen`Environment (conda_mxnet_p36)`, was bedeutet, dass die Umgebung Python 3 hat MXNet . Die andere Variante davon wäre`Environment (conda_mxnet_p27)`, was bedeutet, dass die Umgebung Python 2 hat MXNet.

> **Tip**
>
> Wenn Sie sich Sorgen darüber machen, welche Version von CUDA aktiv ist, können Sie sie unter anderem im MOTD sehen, wenn Sie sich zum ersten Mal beim DLAMI anmelden.

Wechseln von Umgebungen mit Jupyter

Wenn Sie ein Tutorial für ein anderes Framework ausprobieren möchten, überprüfen Sie, welcher Kernel aktuell ausgeführt wird. Diese Information finden Sie oben rechts in der Jupyter-Benutzeroberfläche, direkt unter der Schaltfläche zum Abmelden. Sie können den Kernel in einem beliebigen geöffneten Notebook ändern, indem Sie auf die Jupyter-Menüelemente Kernel, Change Kernel und auf die Umgebung klicken, die für das ausgeführte Notebook geeignet ist.

Nach diesem Schritt müssen Sie alle Zellen erneut ausführen, da eine Änderung des Kernels den Zustand von allen Elementen löscht, die Sie zuvor ausgeführt haben.

> **Tip**
>
> Der Wechsel zwischen Frameworks kann Spaß machen und lehrreich sein. Es kann aber vorkommen, dass Sie nicht mehr über genügend Speicher verfügen. Wenn Fehler auftreten, prüfen Sie das Terminal-Fenster, in dem der Jupyter-Server ausgeführt wird. Hier finden Sie hilfreiche Meldungen und Fehlerprotokollierung, und möglicherweise wird ein Fehler angezeigt. out-of-memory Zum Beheben dieses Problems können Sie zur Startseites Ihres Jupyter-Servers wechseln, die Registerkarte Running auswählen und für die einzelnen Tutorials, die wahrscheinlich noch im Hintergrund ausgeführt werden und den gesamten Speicher beanspruchen, auf Shutdown klicken.

Tutorials

Im Folgenden finden Sie Tutorials zur Verwendung des Deep Learning AMI mit der Software von Conda.

Themen

- Aktivieren von Frameworks
- Verteiltes Training mit Elastic Fabric Adapter
- GPU-Überwachung und -Optimierung
- Der AWS Inferentia-Chip mit DLAMI
- Das ARM64 DLAMI
- Inferenz
- Modellbereitstellung

Aktivieren von Frameworks

Im Folgenden sind die Deep-Learning-Frameworks aufgeführt, die auf dem Deep Learning-AMI mit Conda installiert sind. Klicken Sie auf ein Framework, um zu erfahren, wie Sie es aktivieren.

Themen

- PyTorch
- TensorFlow 2

PyTorch

Wird aktiviert PyTorch

Wenn ein stabiles Conda-Paket eines Frameworks veröffentlicht wird, wird es getestet und auf dem DLAMI vorinstalliert. Wenn Sie den neuesten, nicht getesteten Nightly Build ausführen möchten, können Sie PyTorchInstall's Nightly Build (experimentell) manuell ausführen.

Um das aktuell installierte Framework zu aktivieren, folgen Sie diesen Anweisungen auf Ihrem Deep Learning AMI mit Conda.

Führen PyTorch Sie für Python 3 mit CUDA und MKL-DNN diesen Befehl aus:

```
$ source activate pytorch_p310
```

Starten Sie das iPython-Terminal.

```
(pytorch_p310)$ ipython
```

Führen Sie ein schnelles Programm aus. PyTorch

```
import torch
x = torch.rand(5, 3)
print(x)
print(x.size())
y = torch.rand(5, 3)
print(torch.add(x, y))
```

Das anfängliche zufällige Array sollte angezeigt werden. Danach wird seine Größe angezeigt und dann ein weiteres zufälliges Array.

PyTorchInstall's Nightly Build (experimentell)

Wie installiert man PyTorch von einem Nightly-Build

Sie können den neuesten PyTorch Build in einer oder beiden PyTorch Conda-Umgebungen auf Ihrem Deep Learning-AMI mit Conda installieren.

1. • (Option für Python 3) — Aktiviere die PyTorch Python-3-Umgebung:

```
$ source activate pytorch_p310
```

2. Für die restlichen Schritte wird davon ausgegangen, dass Sie die `pytorch_p310`-Umgebung verwenden. Entfernen Sie das aktuell installierte PyTorch:

```
(pytorch_p310)$ pip uninstall torch
```

3. • (Option für GPU-Instanzen) — Installieren Sie den neuesten nächtlichen Build von PyTorch mit CUDA.0:

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

   • (Option für CPU-Instanzen) — Installieren Sie den neuesten nächtlichen Build von PyTorch für Instances ohne: GPUs

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

4. Um zu überprüfen, ob Sie den neuesten Nightly Build erfolgreich installiert haben, starten Sie das IPython Terminal und überprüfen Sie die Version von. PyTorch

```
(pytorch_p310)$ ipython
```

```
import torch
print (torch.__version__)
```

   Die Druckausgabe sollte in etwa wie folgt aussehen: `1.0.0.dev20180922`

5. Um zu überprüfen, ob der PyTorch Nightly-Build gut mit dem MNIST-Beispiel funktioniert, können Sie ein Testskript aus dem PyTorch Beispiel-Repository ausführen:

```
(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
```

Weitere Tutorials

Weitere Tutorials und Beispiele finden Sie in den offiziellen Dokumenten, der PyTorch Dokumentation und auf der Website des PyTorchFrameworks.

TensorFlow 2

Dieses Tutorial zeigt, wie Sie TensorFlow 2 auf einer Instance aktivieren, auf der das Deep Learning AMI mit Conda (DLAMI on Conda) ausgeführt wird, und ein 2-Programm ausführen. TensorFlow

Wenn ein stabiles Conda-Paket eines Frameworks veröffentlicht wird, wird es getestet und auf dem DLAMI vorinstalliert.

Aktiviert 2 TensorFlow

Um mit TensorFlow Conda auf dem DLAMI zu laufen

1. Um TensorFlow 2 zu aktivieren, öffnen Sie eine Amazon Elastic Compute Cloud (Amazon EC2) - Instanz des DLAMI mit Conda.
2. Führen Sie für TensorFlow 2 und Keras 2 auf Python 3 mit CUDA 10.1 und MKL-DNN diesen Befehl aus:

   ```
   $ source activate tensorflow2_p310
   ```
3. Starten Sie das iPython-Terminal:

   ```
   (tensorflow2_p310)$ ipython
   ```
4. Führen Sie ein TensorFlow 2-Programm aus, um zu überprüfen, ob es ordnungsgemäß funktioniert:

   ```
   import tensorflow as tf
   ```

```
hello = tf.constant('Hello, TensorFlow!')
tf.print(hello)
```

Auf Ihrem Bildschirm sollte `Hello, TensorFlow!` angezeigt werden.

Weitere Tutorials

Weitere Tutorials und Beispiele finden Sie in der TensorFlow Dokumentation zur TensorFlow Python-API oder TensorFlowauf der Website.

Verteiltes Training mit Elastic Fabric Adapter

Ein Elastic Fabric Adapter (EFA) ist ein Netzwerkgerät, das Sie an Ihre DLAMI-Instanz anschließen können, um High Performance Computing (HPC) -Anwendungen zu beschleunigen. EFA ermöglicht es Ihnen, die Anwendungsleistung eines lokalen HPC-Clusters mit der Skalierbarkeit, Flexibilität und Elastizität der Cloud zu erreichen. AWS

Die folgenden Themen zeigen Ihnen, wie Sie mit der Verwendung von EFA mit dem DLAMI beginnen können.

> **Note**
>
> Wählen Sie Ihr DLAMI aus dieser Base-GPU-DLAMI-Liste

Themen

- Starten einer Instance mit EFA AWS Deep Learning AMIs
- EFA auf dem DLAMI verwenden

Starten einer Instance mit EFA AWS Deep Learning AMIs

Das neueste Base DLAMI ist sofort mit EFA einsatzbereit und wird mit den erforderlichen Treibern, Kernelmodulen, libfabric, openmpi und dem NCCL OFI-Plugin für GPU-Instanzen geliefert.

Die unterstützten CUDA-Versionen eines Basis-DLAMI finden Sie in den Versionshinweisen.

Hinweis:

- Wenn Sie eine NCCL-Anwendung `mpirun` auf EFA ausführen, müssen Sie den vollständigen Pfad zu der von EFA unterstützten Installation wie folgt angeben:

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

- Damit Ihre Anwendung EFA verwenden kann, fügen Sie `FI_PROVIDER="efa"` dem `mpirun`-Befehl hinzu, wie unter EFA auf dem DLAMI verwenden gezeigt.

Themen

- Bereiten Sie eine EFA-fähige Sicherheitsgruppe vor
- Starten Ihrer Instance
- Überprüfen Sie den EFA-Anhang

Bereiten Sie eine EFA-fähige Sicherheitsgruppe vor

EFA benötigt eine Sicherheitsgruppe, die den gesamten ein- und ausgehenden Datenverkehr zur und von der Sicherheitsgruppe selbst zulässt. Weitere Informationen finden Sie in der EFA-Dokumentation.

1. Öffnen Sie die EC2 Amazon-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie im Navigationsbereich Security Groups (Sicherheitsgruppen) und anschließend Create Security Group (Sicherheitsgruppe erstellen) aus.
3. Führen Sie im Fenster Create Security Group Folgendes aus:
   - Geben Sie für Security group name (Name der Sicherheitsgruppe) einen beschreibenden Namen für die Sicherheitsgruppe ein, wie etwa `EFA-enabled security group`.
   - (Optional:) Geben Sie unter Description (Beschreibung) eine kurze Beschreibung der Sicherheitsgruppe ein.
   - Wählen Sie bei VPC die VPC aus, in der Sie Ihre EFA-fähigen Instances starten möchten.
   - Wählen Sie Create (Erstellen) aus.
4. Wählen Sie die von Ihnen erstellte Sicherheitsgruppe aus und kopieren Sie dann auf der Registerkarte Description (Beschreibung) die Group ID (Gruppen-ID).
5. Gehen Sie auf den Tabs Inbound und Outbound wie folgt vor:
   - Wählen Sie Edit aus.

- Wählen Sie für Type (Typ) die Option All traffic (Gesamter Datenverkehr) aus.
- Wählen Sie für Source (Quelle) die Option Custom (Benutzerdefiniert) aus.
- Fügen Sie die Sicherheitsgruppen-ID, die Sie kopiert haben, in das Feld ein.
- Wählen Sie Speichern.

6. Aktivieren Sie eingehenden Datenverkehr, indem Sie entsprechend der Informationen unter Autorisieren von eingehendem Datenverkehr für Linux-Instances vorgehen. Wenn Sie diesen Schritt überspringen, können Sie nicht mit Ihrer DLAMI-Instanz kommunizieren.

Starten Ihrer Instance

EFA on the AWS Deep Learning AMIs wird derzeit von den folgenden Instance-Typen und Betriebssystemen unterstützt:

- P3dn: Amazon Linux 2, Ubuntu 20.04
- P4d, P4de: Amazon Linux 2, Amazon Linux 2023, Ubuntu 20.04, Ubuntu 22.04
- P5, P5e, P5en: Amazon Linux 2, Amazon Linux 2023, Ubuntu 20.04, Ubuntu 22.04

Der folgende Abschnitt zeigt, wie eine EFA-fähige DLAMI-Instanz gestartet wird. Weitere Informationen zum Starten einer EFA-fähigen Instance finden Sie unter Starten von EFA-fähigen Instances in einer Cluster Placement-Gruppe.

1. Öffnen Sie die EC2 Amazon-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie Launch Instance aus.
3. Wählen Sie auf der Seite „AMI auswählen“ ein unterstütztes DLAMI aus, das Sie auf der Seite mit den DLAMI-Versionshinweisen finden
4. Wählen Sie auf der Seite „Instance-Typ auswählen“ einen der folgenden unterstützten Instance-Typen aus und klicken Sie dann auf Weiter: Instance-Details konfigurieren. Unter diesem Link finden Sie eine Liste der unterstützten Instances: Erste Schritte mit EFA und MPI
5. Führen Sie auf der Seite Configure Instance Details (Instance-Details konfigurieren) die folgenden Schritte aus:

   - Geben Sie für Number of instances (Anzahl der Instances) die Anzahl der EFA-aktivierten Instances ein, die gestartet werden sollen.
   - Wählen Sie für Network (Netzwerk) und Subnet (Subnetz) die VPC und das Subnetz an, in das Sie die Instances starten möchten.

- [Optional] Wählen Sie für Placement-Gruppe die Option Instance zur Placement-Gruppe hinzufügen aus. Um eine optimale Leistung zu erzielen, starten Sie die Instances innerhalb einer Platzierungsgruppe.
- [Optional] Wählen Sie für Placement-Gruppenname die Option Zu einer neuen Placement-Gruppe hinzufügen aus, geben Sie einen aussagekräftigen Namen für die Placement-Gruppe ein und wählen Sie dann für Placement-Gruppenstrategie die Option Cluster aus.
- Achten Sie darauf, den „Elastic Fabric Adapter“ auf dieser Seite zu aktivieren. Wenn diese Option deaktiviert ist, ändern Sie das Subnetz in ein Subnetz, das den ausgewählten Instance-Typ unterstützt.
- Wählen Sie im Abschnitt Network Interfaces (Netzwerkschnittstellen für das Gerät eth0 die Option New network interface (Neue Netzwerkschnittstelle) aus. Sie können optional eine primäre IPv4 Adresse und eine oder mehrere sekundäre IPv4 Adressen angeben. Wenn Sie die Instance in einem Subnetz starten, dem ein IPv6 CIDR-Block zugeordnet ist, können Sie optional eine primäre IPv6 Adresse und eine oder mehrere sekundäre IPv6 Adressen angeben.
- Wählen Sie Next: Add Storage aus.

6. Auf der Seite Add Storage (Speicher hinzufügen) können Sie neben den durch das AMI festgelegten Volumes (wie z. B. dem Root-Gerät-Volume) auch Datenträger angeben, die an die Instance angefügt werden sollen. Wählen Sie anschließend Next: Add Tags (Weiter: Tags (Markierungen) hinzufügen) aus.
7. Geben Sie auf der Seite Add Tags (Tags (Markierungen) hinzufügen) Tags (Markierungen) für die Instances an, z. B. einen benutzerfreundlichen Namen, und wählen Sie anschließend Next: Configure Security Group (Weiter: Sicherheitsgruppe konfigurieren).
8. Wählen Sie auf der Seite Sicherheitsgruppe konfigurieren für Sicherheitsgruppe zuweisen die Option Eine bestehende Sicherheitsgruppe auswählen und wählen Sie dann die Sicherheitsgruppe aus, die Sie zuvor erstellt haben.
9. Klicken Sie auf Review and Launch.
10. Überprüfen Sie auf der Seite Review Instance Launch (Instance-Start überprüfen) Ihre Einstellungen und wählen Sie anschließend Launch (Starten) aus, um ein Schlüsselpaar auszuwählen und Ihre Instance zu starten.

Überprüfen Sie den EFA-Anhang

Über die Konsole

Überprüfen Sie nach dem Start der Instance die Instance-Details in der AWS Konsole. Wählen Sie dazu die Instanz in der EC2 Konsole aus und schauen Sie sich die Registerkarte Beschreibung im unteren Bereich der Seite an. Suchen Sie den Parameter "Network Interfaces: eth0" und klicken Sie auf "eth0", wodurch ein Pop-up geöffnet wird. Stellen Sie sicher, dass der 'Elastic Fabric Adapter' aktiviert ist.

Wenn EFA nicht aktiviert ist, können Sie dies wie folgt beheben:

- Beenden Sie die EC2 Instance und starten Sie eine neue mit den gleichen Schritten. Stellen Sie sicher, dass die EFA angehängt ist.
- Hängen Sie EFA an eine bestehende Instanz an.

  1. Gehen Sie in der EC2 Konsole zu Netzwerkschnittstellen.
  2. Klicken Sie auf "Create a Network Interface (Netzwerkschnittstelle erstellen)".
  3. Wählen Sie dasselbe Subnetz aus, in dem sich Ihre Instance befindet.
  4. Stellen Sie sicher, dass der 'Elastic Fabric Adapter' aktiviert ist, und klicken Sie auf Erstellen.
  5. Gehen Sie zurück zum Tab EC2 Instances und wählen Sie Ihre Instance aus.
  6. Gehen Sie zu Aktionen: Instanzstatus und beenden Sie die Instance, bevor Sie EFA anhängen.
  7. Wählen Sie unter "Actions (Aktionen)" die Option "Networking: Attach Network Interface (Netzwerk: Netzwerkschnittstelle anfügen)" aus.
  8. Wählen Sie die soeben erstellte Schnittstelle aus und klicken Sie auf "Attach (Anfügen)".
  9. Starten Sie Ihre Instance neu.

Über die Instance

Das folgende Testskript ist bereits auf dem DLAMI vorhanden. Führen Sie es aus, um sicherzustellen, dass die Kernel-Module ordnungsgemäß geladen sind.

```
$ fi_info -p efa
```

Ihre Ausgabe sollte in etwa wie folgt aussehen.

```
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
```

Überprüfen der Sicherheitsgruppenkonfiguration

Das folgende Testskript ist bereits auf dem DLAMI vorhanden. Führen Sie es aus, um sicherzustellen, dass die von Ihnen erstellte Sicherheitsgruppe ordnungsgemäß konfiguriert ist.

```
$ cd /opt/amazon/efa/test/
$ ./efa_test.sh
```

Ihre Ausgabe sollte in etwa wie folgt aussehen.

```
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
```

Wenn es nicht mehr reagiert oder nicht vollständig ausgeführt wird, stellen Sie sicher, dass Ihre Sicherheitsgruppe über die richtigen inbound/outbound Regeln verfügt.

EFA auf dem DLAMI verwenden

Im folgenden Abschnitt wird beschrieben, wie EFA verwendet wird, um Anwendungen mit mehreren Knoten auf dem auszuführen. AWS Deep Learning AMIs

Ausführen von Anwendungen mit mehreren Knoten mit EFA

Um eine Anwendung auf einem Knotencluster auszuführen, ist die folgende Konfiguration erforderlich

Themen

- Aktivieren von passwortloser SSH
- Erstellen einer Hosts-Datei
- NCCL-Tests

Aktivieren von passwortloser SSH

Wählen Sie einen Knoten im Cluster als Führungsknoten aus. Die verbleibenden Knoten werden als Mitgliedsknoten bezeichnet.

1. Generieren Sie auf dem Führungsknoten das RSA-Tastenpaar.

   ```
   ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
   ```

2. Ändern Sie die Berechtigungen des privaten Schlüssels auf dem Führungsknoten.

   ```
   chmod 600 ~/.ssh/id_rsa
   ```

3. Kopieren Sie den öffentlichen Schlüssel `~/.ssh/id_rsa.pub` auf die Mitgliedsknoten im Cluster und hängen Sie ihn an. `~/.ssh/authorized_keys`
4. Sie sollten nun in der Lage sein, sich direkt mithilfe der privaten IP bei den Mitgliedsknoten über den Führungsknoten anzumelden.

   ```
   ssh <member private ip>
   ```

5. Deaktivieren Sie die strictHostKey Überprüfung und aktivieren Sie die Agentenweiterleitung auf dem Leader-Knoten, indem Sie der Datei ~/.ssh/config auf dem Leader-Knoten Folgendes hinzufügen:

   ```
   Host *
   ```

```
    ForwardAgent yes
Host *
    StrictHostKeyChecking no
```

6. Führen Sie auf Amazon Linux 2-Instances den folgenden Befehl auf dem Leader-Knoten aus, um die richtigen Berechtigungen für die Konfigurationsdatei bereitzustellen:

```
chmod 600 ~/.ssh/config
```

Erstellen einer Hosts-Datei

Erstellen Sie auf dem Führungsknoten eine Hosts-Datei, um die Knoten im Cluster zu identifizieren. Die Hosts-Datei muss für jeden Knoten im Cluster einen Eintrag aufweisen. Erstellen Sie eine Datei "~/hosts" und fügen Sie jeden Knoten mithilfe der privaten IP wie folgt hinzu:

```
localhost slots=8
<private ip of node 1> slots=8
<private ip of node 2> slots=8
```

NCCL-Tests

> **Note**
>
> Diese Tests wurden mit der EFA-Version 1.38.0 und dem OFI NCCL Plugin 1.13.2 ausgeführt.

Im Folgenden ist eine Teilmenge der von Nvidia bereitgestellten NCCL-Tests aufgeführt, mit denen sowohl Funktionalität als auch Leistung über mehrere Rechenknoten hinweg getestet werden können

Unterstützte Instanzen: P3dn, P4, P5, P5e, P5en

Leistungstests

NCCL-Leistungstest mit mehreren Knoten auf P4D.24xlarge

Um die NCCL-Leistung mit EFA zu überprüfen, führen Sie den standardmäßigen NCCL-Leistungstest aus, der im offiziellen NCCL-Tests Repo verfügbar ist. Das DLAMI enthält diesen Test, der bereits für CUDA XX.X erstellt wurde. Sie können auf ähnliche Weise Ihr eigenes Skript mit EFA ausführen.

Wenn Sie Ihr eigenes Skript erstellen, beachten Sie dabei die folgenden Anweisungen:

- Verwenden Sie den vollständigen Pfad zu mpirun, wie im Beispiel gezeigt, während Sie NCCL-Anwendungen mit EFA ausführen.
- Ändern Sie die Parameter np und N basierend auf der Anzahl der Instanzen und in Ihrem Cluster. GPUs
- Fügen Sie das Flag NCCL_DEBUG=INFO hinzu und stellen Sie sicher, dass in den Protokollen die EFA-Nutzung als „Ausgewählter Anbieter ist EFA“ angegeben ist.
- Legen Sie den Speicherort des Trainingsprotokolls fest, der zur Validierung analysiert werden soll

```
TRAINING_LOG="testEFA_$(date +"%N").log"
```

Verwenden Sie den Befehl `watch nvidia-smi` auf einem der Mitgliedsknoten, um die GPU-Nutzung zu überwachen. Die folgenden `watch nvidia-smi` Befehle gelten für eine generische CUDA xx.x-Version und hängen vom Betriebssystem Ihrer Instanz ab. Sie können die Befehle für jede verfügbare CUDA-Version in Ihrer EC2 Amazon-Instance ausführen, indem Sie die CUDA-Version im Skript ersetzen.

- Amazon Linux 2, Amazon Linux 2023:

```
 $ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib64:/opt/amazon/openmpi/
lib64:$LD_LIBRARY_PATH \
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
 none \
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
 100 | tee ${TRAINING_LOG}
```

- Ubuntu 20.04, Ubuntu 20.04:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib:/opt/amazon/openmpi/
lib:$LD_LIBRARY_PATH \
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
 none \
```

```
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
 100 | tee ${TRAINING_LOG}
```

Die Ausgabe sollte folgendermaßen aussehen:

```
# nThread 1 nGpus 1 minBytes 8 maxBytes 1073741824 step: 2(factor) warmup iters: 5
 iters: 100 agg iters: 1 validation: 1 graph: 0
#
# Using devices
#  Rank  0 Group  0 Pid  33378 on ip-172-31-42-25 device  0 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank  1 Group  0 Pid  33379 on ip-172-31-42-25 device  1 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank  2 Group  0 Pid  33380 on ip-172-31-42-25 device  2 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank  3 Group  0 Pid  33381 on ip-172-31-42-25 device  3 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank  4 Group  0 Pid  33382 on ip-172-31-42-25 device  4 [0x90] NVIDIA A100-
SXM4-40GB
#  Rank  5 Group  0 Pid  33383 on ip-172-31-42-25 device  5 [0x90] NVIDIA A100-
SXM4-40GB
#  Rank  6 Group  0 Pid  33384 on ip-172-31-42-25 device  6 [0xa0] NVIDIA A100-
SXM4-40GB
#  Rank  7 Group  0 Pid  33385 on ip-172-31-42-25 device  7 [0xa0] NVIDIA A100-
SXM4-40GB
#  Rank  8 Group  0 Pid  30378 on ip-172-31-43-8 device  0 [0x10] NVIDIA A100-SXM4-40GB
#  Rank  9 Group  0 Pid  30379 on ip-172-31-43-8 device  1 [0x10] NVIDIA A100-SXM4-40GB
#  Rank 10 Group  0 Pid  30380 on ip-172-31-43-8 device  2 [0x20] NVIDIA A100-SXM4-40GB
#  Rank 11 Group  0 Pid  30381 on ip-172-31-43-8 device  3 [0x20] NVIDIA A100-SXM4-40GB
#  Rank 12 Group  0 Pid  30382 on ip-172-31-43-8 device  4 [0x90] NVIDIA A100-SXM4-40GB
#  Rank 13 Group  0 Pid  30383 on ip-172-31-43-8 device  5 [0x90] NVIDIA A100-SXM4-40GB
#  Rank 14 Group  0 Pid  30384 on ip-172-31-43-8 device  6 [0xa0] NVIDIA A100-SXM4-40GB
#  Rank 15 Group  0 Pid  30385 on ip-172-31-43-8 device  7 [0xa0] NVIDIA A100-SXM4-40GB
ip-172-31-42-25:33385:33385 [7] NCCL INFO cudaDriverVersion 12060
ip-172-31-43-8:30383:30383 [5] NCCL INFO Bootstrap : Using ens32:172.31.43.8
ip-172-31-43-8:30383:30383 [5] NCCL INFO NCCL version 2.23.4+cuda12.5
...
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.13.2-aws
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Using Libfabric version 1.22
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Using CUDA driver version 12060 with
 runtime 12050
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Configuring AWS-specific options
```

```
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
 variable to 1
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting NCCL_NVLSTREE_MAX_CHUNKSIZE
 to 512KiB
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting NCCL_NVLS_CHUNKSIZE to 512KiB
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Running on p4d.24xlarge platform,
 Setting NCCL_TOPO_FILE environment variable to /opt/amazon/ofi-nccl/share/aws-ofi-
nccl/xml/p4d-24xl-topo.xml
...
------------------------------some output truncated----------------------------------
#                                                              out-of-place
          in-place
#       size         count      type   redop    root     time   algbw   busbw #wrong
  time   algbw   busbw #wrong
#        (B)    (elements)                               (us)  (GB/s)  (GB/s)
  (us)  (GB/s)  (GB/s)
           8             2     float     sum      -1    180.3    0.00    0.00      0
 179.3    0.00    0.00      0
          16             4     float     sum      -1    178.1    0.00    0.00      0
 177.6    0.00    0.00      0
          32             8     float     sum      -1    178.5    0.00    0.00      0
 177.9    0.00    0.00      0
          64            16     float     sum      -1    178.8    0.00    0.00      0
 178.7    0.00    0.00      0
         128            32     float     sum      -1    178.2    0.00    0.00      0
 177.8    0.00    0.00      0
         256            64     float     sum      -1    178.6    0.00    0.00      0
 178.8    0.00    0.00      0
         512           128     float     sum      -1    177.2    0.00    0.01      0
 177.1    0.00    0.01      0
        1024           256     float     sum      -1    179.2    0.01    0.01      0
 179.3    0.01    0.01      0
        2048           512     float     sum      -1    181.3    0.01    0.02      0
 181.2    0.01    0.02      0
        4096          1024     float     sum      -1    184.2    0.02    0.04      0
 183.9    0.02    0.04      0
        8192          2048     float     sum      -1    191.2    0.04    0.08      0
 190.6    0.04    0.08      0
       16384          4096     float     sum      -1    202.5    0.08    0.15      0
 202.3    0.08    0.15      0
       32768          8192     float     sum      -1    233.0    0.14    0.26      0
 232.1    0.14    0.26      0
```

```
      65536         16384     float     sum      -1    238.6    0.27    0.51      0
 235.1    0.28    0.52      0
     131072         32768     float     sum      -1    237.2    0.55    1.04      0
 236.8    0.55    1.04      0
     262144         65536     float     sum      -1    248.3    1.06    1.98      0
 247.0    1.06    1.99      0
     524288        131072     float     sum      -1    309.2    1.70    3.18      0
 307.7    1.70    3.20      0
    1048576        262144     float     sum      -1    408.7    2.57    4.81      0
 404.3    2.59    4.86      0
    2097152        524288     float     sum      -1    613.5    3.42    6.41      0
 607.9    3.45    6.47      0
    4194304       1048576     float     sum      -1    924.5    4.54    8.51      0
 914.8    4.58    8.60      0
    8388608       2097152     float     sum      -1   1059.5    7.92   14.85      0
 1054.3    7.96   14.92      0
   16777216       4194304     float     sum      -1   1269.9   13.21   24.77      0
 1272.0   13.19   24.73      0
   33554432       8388608     float     sum      -1   1642.7   20.43   38.30      0
 1636.7   20.50   38.44      0
   67108864      16777216     float     sum      -1   2446.7   27.43   51.43      0
 2445.8   27.44   51.45      0
  134217728      33554432     float     sum      -1   4143.6   32.39   60.73      0
 4142.4   32.40   60.75      0
  268435456      67108864     float     sum      -1   7351.9   36.51   68.46      0
 7346.7   36.54   68.51      0
  536870912     134217728     float     sum      -1    13717   39.14   73.39      0
 13703   39.18   73.46      0
 1073741824     268435456     float     sum      -1    26416   40.65   76.21      0
 26420   40.64   76.20      0
...
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 15.5514
```

Validierungstests

Um zu überprüfen, ob die EFA-Tests ein gültiges Ergebnis geliefert haben, verwenden Sie bitte die folgenden Tests zur Bestätigung:

- Rufen Sie den Instanztyp mithilfe der EC2 Instanz-Metadaten ab:

```
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)
```

- Ausführen des Leistungstestss
- Stellen Sie die folgenden Parameter ein

```
CUDA_VERSION
CUDA_RUNTIME_VERSION
NCCL_VERSION
```

- Validieren Sie die Ergebnisse wie gezeigt:

```
RETURN_VAL=`echo $?`
if [ ${RETURN_VAL} -eq 0 ]; then

    # [0] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.13.2-aws
    # [0] NCCL INFO NET/OFI Using CUDA driver version 12060 with runtime 12010

    # cudaDriverVersion 12060  --> This is max supported cuda version by nvidia
 driver
    # NCCL version 2.23.4+cuda12.5 --> This is NCCL version compiled with cuda
 version

    # Validation of logs
    grep "NET/OFI Configuring AWS-specific options" ${TRAINING_LOG} || { echo "AWS-
specific options text not found"; exit 1; }
    grep "busbw" ${TRAINING_LOG} || { echo "busbw text not found"; exit 1; }
    grep "Avg bus bandwidth " ${TRAINING_LOG} || { echo "Avg bus bandwidth text not
 found"; exit 1; }
    grep "NCCL version $NCCL_VERSION" ${TRAINING_LOG} || { echo "Text not found: NCCL
 version $NCCL_VERSION"; exit 1; }
    if [[ ${INSTANCE_TYPE} == "p4d.24xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Text not found: NET/
Libfabric/0/GDRDMA"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
 { echo "Selected Provider is efa text not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p4de.24xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
 text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
 { echo "Avg bus bandwidth text not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p5.48xlarge" ]]; then
```

```
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
 text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
 { echo "Avg bus bandwidth text not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p5e.48xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
 text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
 { echo "Avg bus bandwidth text not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p5en.48xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
 text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 16 nics)" ${TRAINING_LOG} ||
 { echo "Avg bus bandwidth text not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p3dn.24xlarge" ]]; then
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
 { echo "Selected Provider is efa text not found"; exit 1; }
    fi
    echo "***************************** check_efa_nccl_all_reduce passed for cuda
 version ${CUDA_VERSION} *****************************"
else
    echo "***************************** check_efa_nccl_all_reduce failed for cuda
 version ${CUDA_VERSION} *****************************"
fi
```

- Um auf die Benchmark-Daten zuzugreifen, können wir die letzte Zeile der Tabellenausgabe aus dem All_Reduce-Test mit mehreren Knoten analysieren:

```
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}"
```

GPU-Überwachung und -Optimierung

Der folgende Abschnitt führt Sie durch GPU-Optimierungs und -Überwachungsoptionen. Dieser Abschnitt ist wie ein typischer Workflow mit Überwachung, Beaufsichtigung, Vorverarbeitung und Schulung aufgebaut.

- Überwachen
  - Überwachen Sie mit GPUs CloudWatch
- Optimierung
  - Vorverarbeitung
  - Training

Überwachen

Auf Ihrem DLAMI sind mehrere GPU-Überwachungstools vorinstalliert. Diese Anleitung erwähnt auch Tools, die heruntergeladen und installiert werden können.

- Überwachen Sie mit GPUs CloudWatch- ein vorinstalliertes Hilfsprogramm, das Statistiken zur GPU-Nutzung an Amazon CloudWatch meldet.
- nvidia-CLI – ein Dienstprogramm zur Überwachung der allgemeinen GPU-Rechenleistungs- und -Speichernutzung. Dies ist auf Ihrem AWS Deep Learning AMIs (DLAMI) vorinstalliert.
- NVML C-Bibliothek - eine auf C basierende API für den direkten Zugriff auf GPU-Überwachungs- und Verwaltungsfunktionen. Dies wird von der nvidia-smi-CLI intern verwendet und ist auf Ihrem DLAMI vorinstalliert. Dazu gehören weiterhin Python- und Perl-Anbindungen zur Unterstützung der Bereitstellung in diesen Sprachen. Das auf Ihrem DLAMI vorinstallierte Hilfsprogramm gpumon.py verwendet das Paket pynvml von. nvidia-ml-py
- NVIDIA DCGM - Ein Cluster-Management-Tool. Besuchen Sie die Entwicklerseite, um zu erfahren, wie Sie dieses Tool installieren und konfigurieren.

> **Tip**
>
> Im Entwickler-Blog von NVIDIA finden Sie die neuesten Informationen zur Verwendung der auf Ihrem DLAMI installierten CUDA-Tools:
>
> - Überwachung TensorCore der Auslastung mit Nsight IDE und nvprof.

Überwachen Sie mit GPUs CloudWatch

Wenn Sie Ihre DLAMI mit einem GPU verwenden, stellen Sie möglicherweise fest, dass Sie auf der Suche nach Möglichkeiten zur Nachverfolgung der Nutzung während der Schulung oder Inferenz sind. Dies kann nützlich sein, um Ihre Datenpipeline zu optimieren und Ihr Deep Learning-Netzwerk zu verfeinern.

Es gibt zwei Möglichkeiten, GPU-Metriken zu konfigurieren mit CloudWatch:

- Metriken mit dem AWS CloudWatch Agenten konfigurieren (empfohlen)
- Konfigurieren Sie Metriken mit dem vorinstallierten Skript `gpumon.py`

Metriken mit dem AWS CloudWatch Agenten konfigurieren (empfohlen)

Integrieren Sie Ihr DLAMI in den Unified CloudWatch Agent, um GPU-Metriken zu konfigurieren und die Nutzung von GPU-Koprozessen in EC2 Amazon-beschleunigten Instances zu überwachen.

Es gibt vier Möglichkeiten, GPU-Metriken mit Ihrem DLAMI zu konfigurieren:

- Konfigurieren Sie minimale GPU-Metriken
- Konfigurieren Sie partielle GPU-Metriken
- Konfigurieren Sie alle verfügbaren GPU-Metriken
- Konfigurieren Sie benutzerdefinierte GPU-Metriken

Informationen zu Updates und Sicherheitspatches finden Sie unter Sicherheitspatches für den Agenten AWS CloudWatch

Voraussetzungen

Zu Beginn müssen Sie IAM-Berechtigungen für EC2 Amazon-Instances konfigurieren, an die Ihre Instance Metriken CloudWatch weiterleiten kann. Ausführliche Schritte finden Sie unter IAM-Rollen und -Benutzer für die Verwendung mit dem CloudWatch Agenten erstellen.

Konfigurieren Sie minimale GPU-Metriken

Konfigurieren Sie minimale GPU-Metriken mithilfe des `dlami-cloudwatch-agent@minimal systemd` Dienstes. Dieser Dienst konfiguriert die folgenden Metriken:

- `utilization_gpu`

- utilization_memory

Sie finden den systemd Dienst für minimale vorkonfigurierte GPU-Metriken an der folgenden Stelle:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-minimal.json
```

Aktivieren und starten Sie den systemd Dienst mit den folgenden Befehlen:

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

Konfigurieren Sie partielle GPU-Metriken

Konfigurieren Sie partielle GPU-Metriken mithilfe des dlami-cloudwatch-agent@partial systemd Dienstes. Dieser Dienst konfiguriert die folgenden Metriken:

- utilization_gpu
- utilization_memory
- memory_total
- memory_used
- memory_free

Sie finden den systemd Dienst für teilweise vorkonfigurierte GPU-Metriken an der folgenden Stelle:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-partial.json
```

Aktivieren und starten Sie den systemd Dienst mit den folgenden Befehlen:

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

Konfigurieren Sie alle verfügbaren GPU-Metriken

Konfigurieren Sie alle verfügbaren GPU-Metriken mithilfe des dlami-cloudwatch-agent@all systemd Dienstes. Dieser Dienst konfiguriert die folgenden Metriken:

- 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

Sie finden den systemd Dienst für alle verfügbaren vorkonfigurierten GPU-Metriken an der folgenden Stelle:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-all.json
```

Aktivieren und starten Sie den systemd Dienst mit den folgenden Befehlen:

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

Konfigurieren Sie benutzerdefinierte GPU-Metriken

Wenn die vorkonfigurierten Metriken Ihren Anforderungen nicht entsprechen, können Sie eine benutzerdefinierte CloudWatch Agentenkonfigurationsdatei erstellen.

Erstellen Sie eine benutzerdefinierte Konfigurationsdatei

Informationen zum Erstellen einer benutzerdefinierten Konfigurationsdatei finden Sie in den detaillierten Schritten unter Manuelles Erstellen oder Bearbeiten der CloudWatch Agentenkonfigurationsdatei.

Gehen Sie für dieses Beispiel davon aus, dass sich die Schemadefinition unter befindet`/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`.

Konfigurieren Sie Metriken mit Ihrer benutzerdefinierten Datei

Führen Sie den folgenden Befehl aus, um den CloudWatch Agenten entsprechend Ihrer benutzerdefinierten Datei zu konfigurieren:

```
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
```

Sicherheitspatches für den Agenten AWS CloudWatch

Neu veröffentlichte Versionen DLAMIs sind mit den neuesten verfügbaren AWS CloudWatch Agent-Sicherheitspatches konfiguriert. In den folgenden Abschnitten erfahren Sie, wie Sie Ihr aktuelles DLAMI je nach Betriebssystem Ihrer Wahl mit den neuesten Sicherheitspatches aktualisieren können.

Amazon Linux 2

Wird verwendet`yum`, um die neuesten AWS CloudWatch Agenten-Sicherheitspatches für ein Amazon Linux 2-DLAMI zu erhalten.

```
sudo yum update
```

Ubuntu

Um die neuesten AWS CloudWatch Sicherheitspatches für ein DLAMI mit Ubuntu zu erhalten, muss der AWS CloudWatch Agent über einen Amazon S3 S3-Download-Link neu installiert werden.

```
wget https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/
amazon-cloudwatch-agent.deb
```

Weitere Informationen zur Installation des AWS CloudWatch Agenten mithilfe von Amazon S3 S3-Download-Links finden Sie unter Installation und Ausführung des CloudWatch Agenten auf Ihren Servern.

Konfigurieren Sie Metriken mit dem vorinstallierten Skript **`gpumon.py`**

Das Dienstprogramm gpumon.py ist auf Ihrem DLAMI vorinstalliert. Es lässt sich in die Nutzung pro GPU integrieren CloudWatch und unterstützt deren Überwachung: GPU-Speicher, GPU-Temperatur und GPU-Leistung. Das Skript sendet die überwachten Daten regelmäßig an CloudWatch. Sie können die Granularität für die zu sendenden Daten konfigurieren, CloudWatch indem Sie einige Einstellungen im Skript ändern. Bevor Sie das Skript starten, müssen Sie jedoch einrichten, um die Metriken CloudWatch zu empfangen.

Wie richte ich die GPU-Überwachung ein und führe sie aus CloudWatch

1. Erstellen Sie einen IAM-Benutzer oder ändern Sie einen vorhandenen Benutzer, um eine Richtlinie für die Veröffentlichung der Metrik festzulegen CloudWatch. Wenn Sie einen neuen Benutzer erstellen, notieren Sie sich die Anmeldeinformationen, da Sie diese im nächsten Schritt benötigen.

   Die IAM-Richtlinie, nach der gesucht werden soll, lautet „cloudwatch:“. PutMetricData Die Richtlinie, die hinzugefügt wird, lautet wie folgt:

   ```
   {
      "Version": "2012-10-17",
      "Statement": [
           {
               "Action": [
                   "cloudwatch:PutMetricData"
                ],
                "Effect": "Allow",
                "Resource": "*"
           }
      ]
   }
   ```

   > **Tip**
   >
   > Weitere Informationen zum Erstellen eines IAM-Benutzers und zum Hinzufügen von Richtlinien für CloudWatch finden Sie in der Dokumentation. CloudWatch

2. Führen Sie auf Ihrem DLAMI AWS configure aus und geben Sie die IAM-Benutzeranmeldedaten an.

```
$ aws configure
```

3. Möglicherweise müssen Sie einige Änderungen am gpumon-Dienstprogramm vornehmen, bevor Sie es ausführen können. Sie finden das Gpumon-Hilfsprogramm und die README-Datei an dem im folgenden Codeblock definierten Speicherort. Weitere Informationen zum `gpumon.py` Skript finden Sie im Amazon S3 S3-Speicherort des Skripts.

```
Folder: ~/tools/GPUCloudWatchMonitor
Files:  ~/tools/GPUCloudWatchMonitor/gpumon.py
       ~/tools/GPUCloudWatchMonitor/README
```

   Optionen:

   - Ändern Sie die Region in gpumon.py, wenn sich Ihre Instance NICHT in us-east-1 befindet.
   - Ändern Sie andere Parameter wie den CloudWatch `namespace` oder den Berichtszeitraum mit`store_reso`.

4. Derzeit unterstützt das Skript nur Python 3. Aktivieren Sie die Python-3-Umgebung Ihres bevorzugten Frameworks oder aktivieren Sie die allgemeine Python-3-Umgebung von DLAMI.

```
$ source activate python3
```

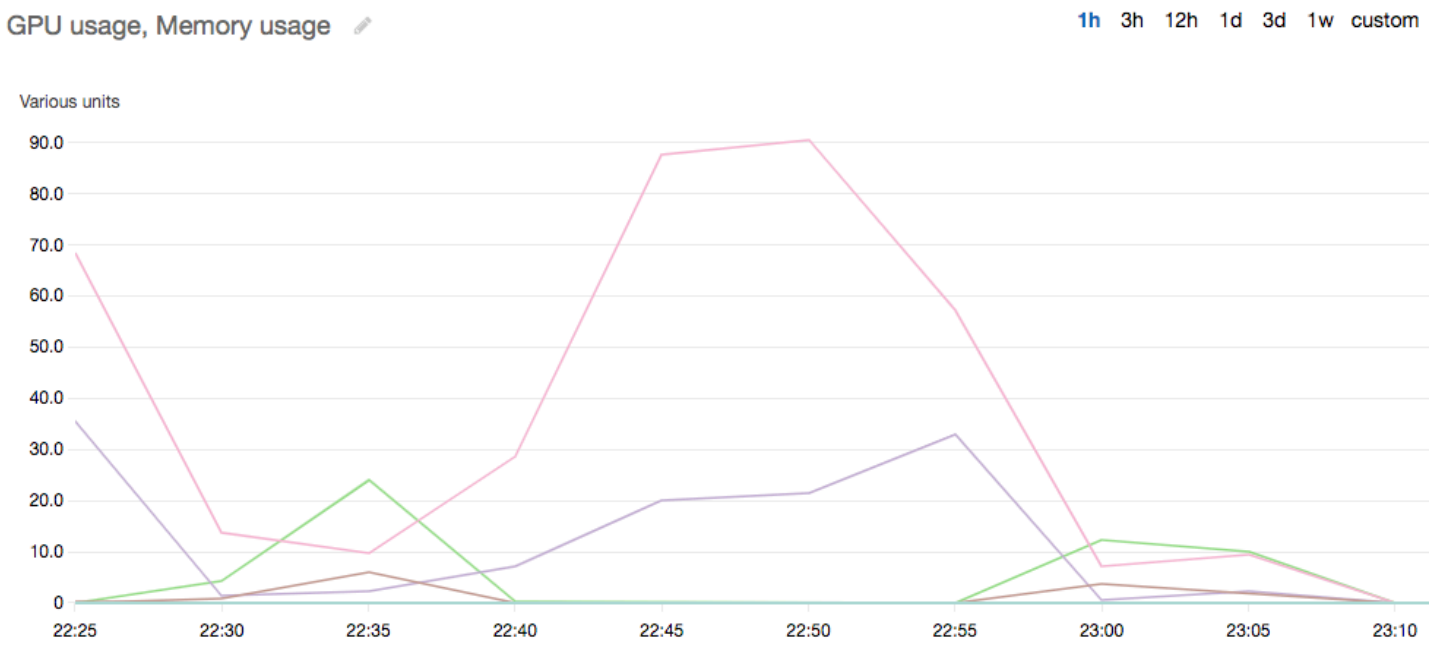
5. Führen Sie das gpumon-Dienstprogramm im Hintergrund aus.

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

6. Öffnen Sie Ihren Browser mit https://console.aws.amazon.com/cloudwatch/ und wählen Sie dann die Metrik. Sie wird einen Namespace " haben. DeepLearningTrain

> **Tip**
>
> Sie können den Namespace durch Modifizierung von gpumon.py ändern. Sie können auch das Berichtsintervall durch Anpassung von `store_reso` ändern.

Im Folgenden finden Sie ein CloudWatch Beispieldiagramm, das über einen Lauf von gpumon.py berichtet, der einen Trainingsjob auf der p2.8xlarge-Instance überwacht.



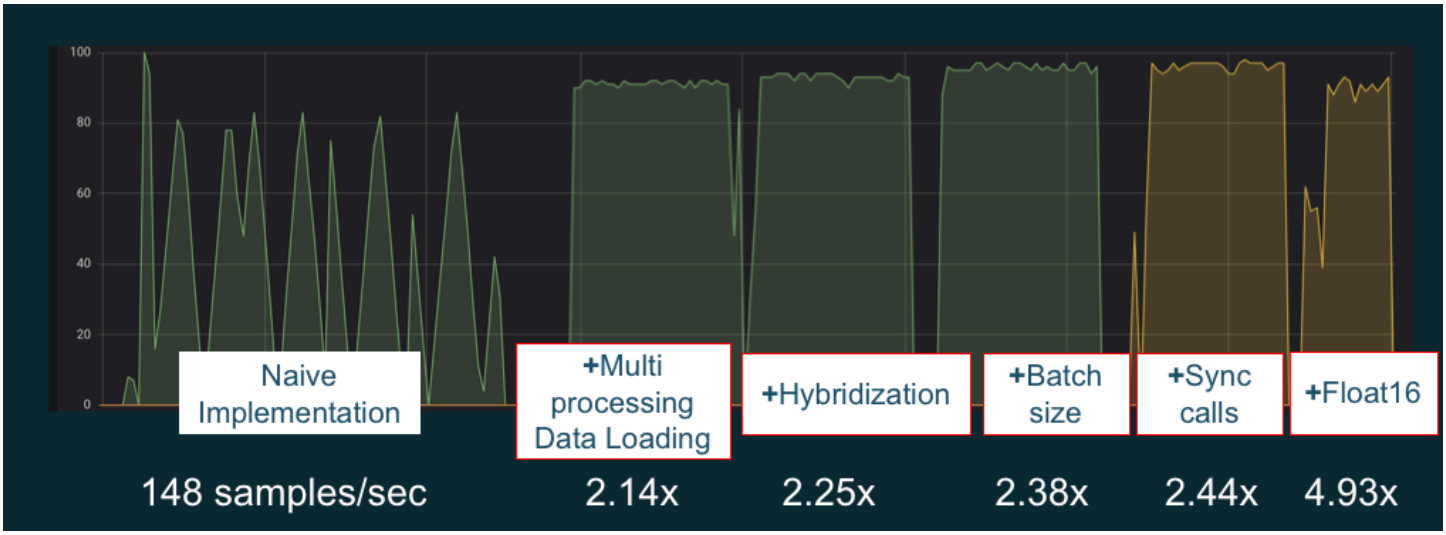
Möglicherweise sind diese weiteren Themen zur GPU-Überwachung und -Optimierung für Sie interessant:

- Überwachen
  - Überwachen Sie mit GPUs CloudWatch
- Optimierung
  - Vorverarbeitung
  - Training

Optimierung

Um das Beste aus Ihrem herauszuholen GPUs, können Sie Ihre Datenpipeline optimieren und Ihr Deep-Learning-Netzwerk optimieren. Wie das folgende Diagramm zeigt, nutzt eine „naive“ oder einfache Implementierung eines neuronalen Netzwerks die GPU möglicherweise inkonsistent und nicht mit ihrem vollen Potenzial. Wenn Sie Ihre Vorverarbeitung und das Laden der Daten optimieren, können Sie den Engpass von CPU zu GPU reduzieren. Sie können das neuronale Netzwerk selbst anpassen, indem Sie Hybridisierung (wenn vom Framework unterstützt) verwenden, die Stapelgröße anpassen sowie Aufrufe synchronisieren. Sie können in den meisten Frameworks auch Schulungen mit mehreren Präzisionen verwenden, was dramatische Verbesserungen des Durchsatzes mit sich bringen kann.

Das folgende Diagramm zeigt die kumulativen Leistungssteigerungen bei der Anwendung verschiedener Optimierungen. Ihre Ergebnisse hängen von den verarbeiteten Daten und dem optimierten Netzwerk ab.



Beispiel für GPU-Leistungsoptimierungen. Quelle des Diagramms: Leistungstricks mit MXNet Gluon

In den folgenden Anleitungen werden Optionen vorgestellt, die mit Ihrem DLAMI funktionieren und Ihnen helfen, die GPU-Leistung zu steigern.

Themen

- Vorverarbeitung
- Training

Vorverarbeitung

Die Vorverarbeitung von Daten durch Umwandlungen oder Erweiterungen ist oft ein CPU-gebundener Prozess, was zu einem Engpass in Ihrer allgemeinen Pipeline führen kann. Frameworks verfügen über integrierte Operatoren für die Abbildverarbeitung, DALI (Data Augmentation Library) zeigt jedoch eine verbesserte Leistung gegenüber den in Frameworks integrierten Optionen.

- NVIDIA Data Augmentation Library (DALI): DALI übergibt die Datenerweiterung an die GPU. Es ist nicht auf dem DLAMI vorinstalliert, aber Sie können darauf zugreifen, indem Sie es installieren oder einen unterstützten Framework-Container auf Ihrem DLAMI oder einer anderen Amazon Elastic Compute Cloud-Instanz laden. Einzelheiten finden Sie auf der DALI-Projektseite auf der NVIDIA-Website. Ein Anwendungsbeispiel und zum Herunterladen von Codebeispielen finden Sie im Beispiel Preprocessing Training Performance. SageMaker

- nvJPEG: eine GPU-beschleunigte JPEG-Decoder-Bibliothek für C-Programmierer. Sie unterstützt die Decodierung einzelner Abbilder oder Stapel sowie Transformationsoperationen, die für Deep Learning verbreitet sind. nvJPEG ist mit DALI integriert. Alternativ können Sie es von der nvjpeg-Seite der NVIDIA-Website herunterladen und separat verwenden.

Möglicherweise sind diese weiteren Themen zur GPU-Überwachung und -Optimierung für Sie interessant:

- Überwachen
  - Überwachen Sie mit GPUs CloudWatch
- Optimierung
  - Vorverarbeitung
  - Training

Training

Mit Schulungen mit gemischter Präzision können Sie größere Netzwerke mit derselben Speichergröße bereitstellen oder die Speichernutzung gegenüber Ihrem Netzwerk mit einzelner oder doppelter Präzision reduzieren - die Rechenleistung wird dadurch sicher gesteigert. Dazu kommt der Vorteil kleinerer und schnellerer Datenübertragungen, ein wichtiger Faktor für verteilte Schulungen mit mehreren Knoten. Um die Schulung mit gemischter Präzision nutzen zu können, müssen Sie das Data Casting und die Verlustskalierung anpassen. Nachfolgend finden Sie Anleitungen dazu, wie Sie dies für Frameworks tun können, die gemischte Präzisionen unterstützen.

- NVIDIA Deep Learning SDK — Dokumente auf der NVIDIA-Website, in denen die Implementierung mit gemischter Genauigkeit für, und beschrieben wird. MXNet PyTorch TensorFlow

> **Tip**
>
> Konsultieren Sie die Website für das von Ihnen gewählte Framework und suchen Sie nach „gemischte Präzision“ oder „fp16“ für die jeweils neuesten Optimierungstechniken. Hier finden Sie einige Anleitungen für gemischte Präzisionen, die möglicherweise nützlich für Sie sind:
>
> - Training mit gemischter Präzision mit TensorFlow (Video) — auf der NVIDIA-Blogseite.
> - Training mit gemischter Präzision mit Float16 mit MXNet - ein FAQ-Artikel auf der Website. MXNet

- NVIDIA Apex: ein Tool für einfaches Training mit gemischter Präzision mit PyTorch — einem Blogartikel auf der NVIDIA-Website.

Möglicherweise sind diese weiteren Themen zur GPU-Überwachung und -Optimierung für Sie interessant:

- Überwachen
  - Überwachen Sie mit GPUs CloudWatch
- Optimierung
  - Vorverarbeitung
  - Training

Der AWS Inferentia-Chip mit DLAMI

AWS Inferentia ist ein maßgeschneiderter Chip für maschinelles Lernen, der speziell für leistungsstarke AWS Inferenzvorhersagen entwickelt wurde. Um den Chip zu verwenden, richten Sie eine Amazon Elastic Compute Cloud-Instanz ein und verwenden Sie das AWS Neuron Software Development Kit (SDK), um den Inferentia-Chip aufzurufen. Um Kunden das beste Inferentia-Erlebnis zu bieten, wurde Neuron in das AWS Deep Learning AMIs (DLAMI) integriert.

Die folgenden Themen zeigen Ihnen, wie Sie mit der Verwendung von Inferentia mit dem DLAMI beginnen können.

Inhalt

- Starten einer DLAMI-Instanz mit Neuron AWS
- Verwendung des DLAMI mit Neuron AWS

Starten einer DLAMI-Instanz mit Neuron AWS

Das neueste DLAMI ist sofort mit AWS Inferentia einsatzbereit und wird mit dem AWS Neuron API-Paket geliefert. Informationen zum Starten einer DLAMI-Instanz finden Sie unter Starten und Konfigurieren einer DLAMI-Instanz. Nachdem Sie ein DLAMI haben, gehen Sie wie hier beschrieben vor, um sicherzustellen, dass Ihr AWS Inferentia-Chip und Ihre AWS Neuron-Ressourcen aktiv sind.

Inhalt

- Verifizieren der Instance
- Identifizieren von AWS Inferentia-Geräten
- Anzeigen des Ressourcenverbrauchs
- Verwendung von Neuron Monitor (Neuron-Monitor)
- Aktualisierung der Neuron-Software

Verifizieren der Instance

Bevor Sie Ihre Instance verwenden, stellen Sie sicher, dass sie ordnungsgemäß eingerichtet und mit Neuron konfiguriert ist.

Identifizieren von AWS Inferentia-Geräten

Verwenden Sie den folgenden Befehl, um die Anzahl der Inferentia-Geräte auf Ihrer Instance zu ermitteln:

```
neuron-ls
```

Wenn an Ihre Instance Inferentia-Geräte angeschlossen sind, sieht Ihre Ausgabe in etwa wie folgt aus:

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

Die bereitgestellte Ausgabe stammt aus einer INF1.6XLarge-Instance und umfasst die folgenden Spalten:

- NEURON DEVICE: Die logische ID, die dem zugewiesen ist. NeuronDevice Diese ID wird verwendet, wenn mehrere Laufzeiten für die Verwendung verschiedener Laufzeiten konfiguriert werden. NeuronDevices
- NEURON-KERNE: Die Anzahl der NeuronCores vorhandenen Kerne in der. NeuronDevice
- NEURONENSPEICHER: Die Menge an DRAM-Speicher im. NeuronDevice

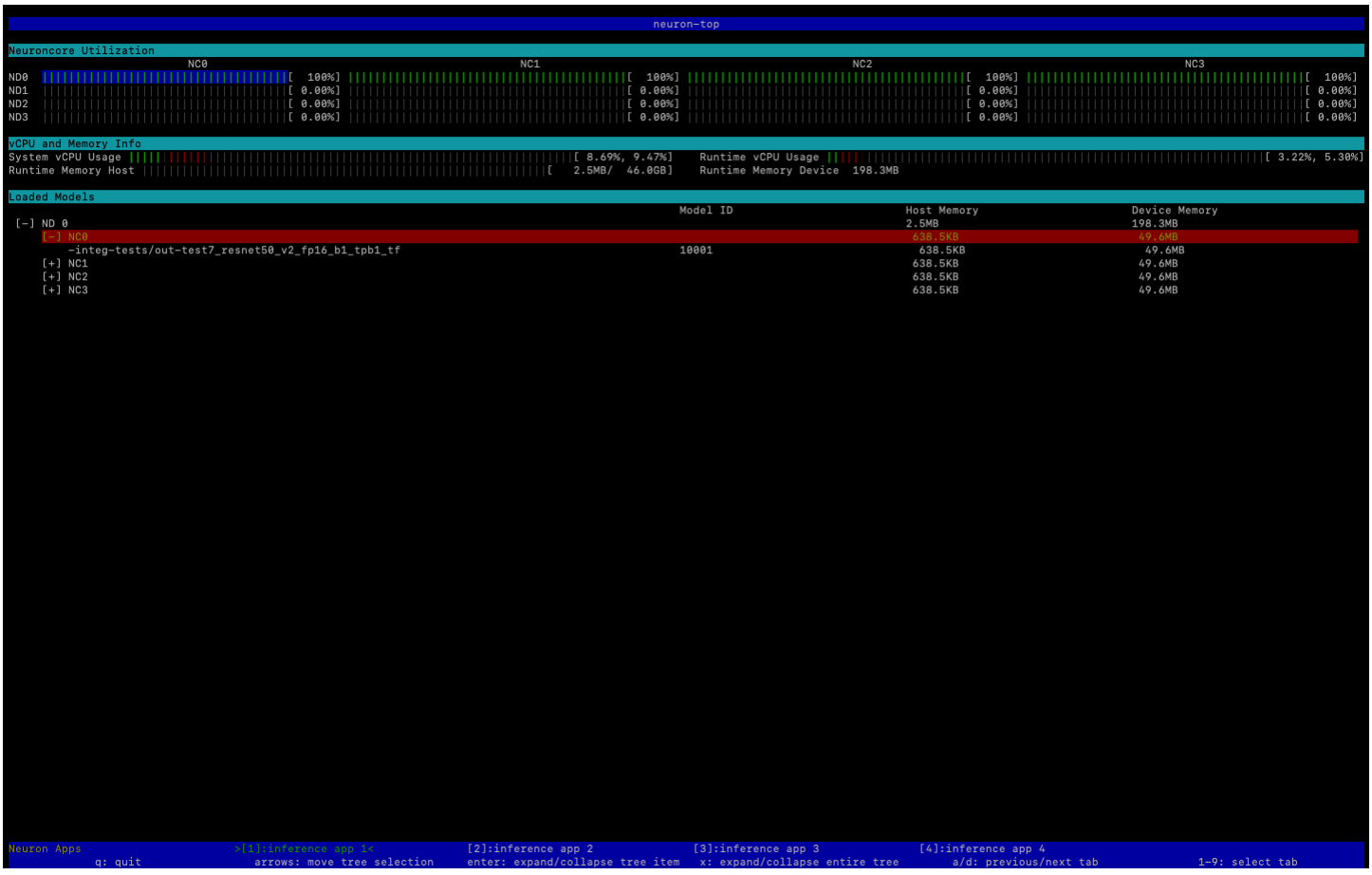
- ANGESCHLOSSENE GERÄTE: Andere, die NeuronDevices mit dem verbunden sind. NeuronDevice
- PCI BDF: Die PCI-Bus-Gerätefunktions-ID (BDF) des. NeuronDevice

Anzeigen des Ressourcenverbrauchs

Zeigen Sie mit dem Befehl nützliche Informationen NeuronCore zur vCPU-Auslastung, zur Speichernutzung, zu geladenen Modellen und Neuron-Anwendungen an`neuron-top`. Wenn Sie `neuron-top` ohne Argumente starten, werden Daten für alle maschinellen Lernanwendungen angezeigt, die dies verwenden. NeuronCores

```
neuron-top
```

Wenn eine Anwendung vier verwendet NeuronCores, sollte die Ausgabe der folgenden Abbildung ähneln:



Weitere Informationen zu Ressourcen zur Überwachung und Optimierung neuronaler Inferenzanwendungen finden Sie unter Neuron Tools.

Verwendung von Neuron Monitor (Neuron-Monitor)

Neuron Monitor sammelt Metriken aus den Neuron-Laufzeiten, die auf dem System laufen, und streamt die gesammelten Daten im JSON-Format auf stdout. Diese Metriken sind in Metrikgruppen organisiert, die Sie konfigurieren, indem Sie eine Konfigurationsdatei bereitstellen. Weitere Informationen zu Neuron Monitor finden Sie im Benutzerhandbuch für Neuron Monitor.

Aktualisierung der Neuron-Software

Informationen zum Aktualisieren der Neuron SDK-Software in DLAMI finden Sie im AWS Neuron Setup Guide.

Nächster Schritt

Verwendung des DLAMI mit Neuron AWS

Verwendung des DLAMI mit Neuron AWS

Ein typischer Arbeitsablauf mit dem AWS Neuron SDK besteht darin, ein zuvor trainiertes Modell für maschinelles Lernen auf einem Kompilierungsserver zu kompilieren. Verteilen Sie anschließend die Artefakte zur Ausführung an die Inf1-Instanzen. AWS Deep Learning AMIs (DLAMI) ist mit allem vorinstalliert, was Sie zum Kompilieren und Ausführen von Inferenzen in einer Inf1-Instanz benötigen, die Inferentia verwendet.

In den folgenden Abschnitten wird beschrieben, wie Sie das DLAMI mit Inferentia verwenden.

Inhalt

- Verwendung von TensorFlow -Neuron und dem Neuron Compiler AWS
- Verwenden von AWS Neuron Serving TensorFlow
- Verwendung von -Neuron MXNet und dem Neuron Compiler AWS
- Verwenden von MXNet -Neuron Model Serving
- Verwenden von PyTorch -Neuron und dem Neuron Compiler AWS

Verwendung von TensorFlow -Neuron und dem Neuron Compiler AWS

Dieses Tutorial zeigt, wie Sie mit dem AWS Neuron-Compiler das Keras ResNet -50-Modell kompilieren und als gespeichertes Modell im Format exportieren. SavedModel Dieses Format ist ein typisches TensorFlow austauschbares Modellformat. Sie lernen außerdem, wie die Inferenz für eine Inf1-Instance mit Beispieldaten ausgeführt wird.

Weitere Informationen zum Neuron SDK finden Sie in der AWS Neuron SDK-Dokumentation.

Inhalt

- Voraussetzungen
- Aktivieren der Conda-Umgebung
- Resnet50-Kompilierung
- ResNet50 Inferenz

Voraussetzungen

Bevor Sie dieses Tutorial verwenden, müssen Sie die Einrichtungsschritte in Starten einer DLAMI-Instanz mit Neuron AWS abgeschlossen haben. Sie sollten auch mit Deep Learning und der Verwendung des DLAMI vertraut sein.

Aktivieren der Conda-Umgebung

Aktivieren Sie die TensorFlow -Neuron Conda-Umgebung mit dem folgenden Befehl:

```
source activate aws_neuron_tensorflow_p36
```

Führen Sie den folgenden Befehl aus, um die aktuelle Conda-Umgebung zu verlassen:

```
source deactivate
```

Resnet50-Kompilierung

Erstellen Sie das Python-Skript **`tensorflow_compile_resnet50.py`** mit folgendem Inhalt. Dieses Python-Skript kompiliert das Keras ResNet 50-Modell und exportiert es als gespeichertes Modell.

```
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')
```

Kompilieren Sie das Modell mit dem folgenden Befehl:

```
python tensorflow_compile_resnet50.py
```

Der Kompilierungsprozess dauert einige Minuten. Nach Abschluss sollte die Ausgabe so aussehen:

```
...
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
```

```
...
```

Nach der Kompilierung wird das gespeicherte Modell gezippt: **`ws_resnet50/resnet50_neuron.zip`**. Entzippen Sie das Modell und laden Sie das Beispielbild mit den folgenden Befehlen herunter:

```
unzip ws_resnet50/resnet50_neuron.zip -d .
curl -O https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/images/kitten_small.jpg
```

ResNet50 Inferenz

Erstellen Sie das Python-Skript **`tensorflow_infer_resnet50.py`** mit dem folgenden Inhalt. Dieses Skript führt die Inferenz für das heruntergeladene Modell unter Verwendung eines zuvor kompilierten Inferenzmodells aus.

```
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])
```

Führen Sie die Inferenz für das Modell mit dem folgenden Befehl aus:

```
python tensorflow_infer_resnet50.py
```

Die Ausgabe sollte folgendermaßen aussehen:

```
...
[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
 'snow_leopard', 0.009290541)]
```

Nächster Schritt

Verwenden von AWS Neuron Serving TensorFlow

Verwenden von AWS Neuron Serving TensorFlow

Dieses Tutorial zeigt, wie Sie ein Diagramm erstellen und einen AWS Neuron-Kompilierungsschritt hinzufügen, bevor Sie das gespeicherte Modell zur Verwendung mit TensorFlow Serving exportieren. TensorFlow Serving ist ein Serversystem, mit dem Sie Inferenzen in einem Netzwerk skalieren können. Neuron TensorFlow Serving verwendet dieselbe API wie normales Serving. TensorFlow Der einzige Unterschied besteht darin, dass ein gespeichertes Modell für AWS Inferentia kompiliert werden muss und der Einstiegspunkt eine andere Binärdatei mit dem Namen ist. `tensorflow_model_server_neuron` Die Binärdatei befindet sich im DLAMI `/usr/local/bin/tensorflow_model_server_neuron` und ist dort vorinstalliert.

Weitere Informationen zum Neuron SDK finden Sie in der Neuron SDK-Dokumentation.AWS

Inhalt

- Voraussetzungen
- Aktivieren der Conda-Umgebung
- Kompilieren und Exportieren des gespeicherten Modells
- Bereitstellen des gespeicherten Modells
- Generieren von Inferenzanforderungen an den Modell-Server

Voraussetzungen

Bevor Sie dieses Tutorial verwenden, müssen Sie die Einrichtungsschritte in Starten einer DLAMI-Instanz mit Neuron AWS abgeschlossen haben. Sie sollten auch mit Deep Learning und der Verwendung des DLAMI vertraut sein.

Aktivieren der Conda-Umgebung

Aktivieren Sie die TensorFlow -Neuron Conda-Umgebung mit dem folgenden Befehl:

```
source activate aws_neuron_tensorflow_p36
```

Wenn Sie die aktuelle Conda-Umgebung verlassen müssen, führen Sie Folgendes aus:

```
source deactivate
```

Kompilieren und Exportieren des gespeicherten Modells

Erstellen Sie ein Python-Skript namens `tensorflow-model-server-compile.py` mit dem folgenden Inhalt. Dieses Skript erstellt ein Diagramm und kompiliert es mit Neuron. Anschließend wird das kompilierte Diagramm als gespeichertes Modell exportiert.

```
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)
```

Kompilieren Sie das Modell mit dem folgenden Befehl:

```
python tensorflow-model-server-compile.py
```

Die Ausgabe sollte folgendermaßen aussehen:

```
...
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
```

Bereitstellen des gespeicherten Modells

Nachdem das Modell kompiliert wurde, können Sie den folgenden Befehl verwenden, um das gespeicherte Modell mit der Binärdatei tensorflow_model_server_neuron bereitzustellen:

```
tensorflow_model_server_neuron --model_name=resnet50_inf1 \
    --model_base_path=$HOME/resnet50_inf1/ --port=8500 &
```

Die Ausgabe sollte wie folgt aussehen. Das kompilierte Modell wird vom Server im DRAM des Inferentia-Geräts gespeichert, um die Inferenz vorzubereiten.

```
...
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 ...
```

Generieren von Inferenzanforderungen an den Modell-Server

Erstellen Sie das Python-Skript namens `tensorflow-model-server-infer.py` mit folgendem Inhalt. Dieses Skript führt die Inferenz über gRPC (Service-Framework) aus.

```
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))
```

Führen Sie die Inferenz für das Modell aus, indem Sie gRPC mit dem folgenden Befehl verwenden:

```
python tensorflow-model-server-infer.py
```

Die Ausgabe sollte folgendermaßen aussehen:

```
[[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
 'snow_leopard', 0.009290541)]]
```

Verwendung von -Neuron MXNet und dem Neuron Compiler AWS

Die MXNet -Neuron-Kompilierungs-API bietet eine Methode zum Kompilieren eines Modelldiagramms, das Sie auf einem Inferentia-Gerät ausführen können. AWS

In diesem Beispiel verwenden Sie die API, um ein ResNet -50-Modell zu kompilieren und damit Inferenzen auszuführen.

Weitere Informationen zum Neuron SDK finden Sie in der Neuron SDK-Dokumentation AWS .

Inhalt

- Voraussetzungen
- Aktivieren der Conda-Umgebung
- Resnet50-Kompilierung
- ResNet50 Folgerung

Voraussetzungen

Bevor Sie dieses Tutorial verwenden, müssen Sie die Einrichtungsschritte in Starten einer DLAMI-Instanz mit Neuron AWS abgeschlossen haben. Sie sollten auch mit Deep Learning und der Verwendung des DLAMI vertraut sein.

Aktivieren der Conda-Umgebung

Aktivieren Sie die MXNet -Neuron Conda-Umgebung mit dem folgenden Befehl:

```
source activate aws_neuron_mxnet_p36
```

Führen Sie Folgendes aus, um die aktuelle Conda-Umgebung zu verlassen:

```
source deactivate
```

Resnet50-Kompilierung

Erstellen Sie das Python-Skript namens **`mxnet_compile_resnet50.py`** mit folgendem Inhalt. Dieses Skript verwendet die Python-API zur Kompilierung von MXNet -Neuron, um ein ResNet -50-Modell zu kompilieren.

```
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)
```

Kompilieren Sie das Modell mit dem folgenden Befehl:

```
python mxnet_compile_resnet50.py
```

Die Kompilierung dauert einige Minuten. Wenn die Kompilierung abgeschlossen ist, befinden sich die folgenden Dateien in Ihrem aktuellen Verzeichnis:

```
resnet-50-0000.params
resnet-50-symbol.json
compiled_resnet50-0000.params
compiled_resnet50-symbol.json
```

ResNet50 Folgerung

Erstellen Sie das Python-Skript namens **`mxnet_infer_resnet50.py`** mit folgendem Inhalt. Mit diesem Skript wird ein Beispiel-Image heruntergeladen, das dazu verwendet wird, um die Inferenz für das kompilierte Modell auszuführen.

```
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]))
```

Führen Sie die Inferenz mit folgendem Befehl mit dem kompilierten Modell aus:

```
python mxnet_infer_resnet50.py
```

Die Ausgabe sollte folgendermaßen aussehen:

```
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
```

Nächster Schritt

Verwenden von MXNet -Neuron Model Serving

Verwenden von MXNet -Neuron Model Serving

In diesem Tutorial lernen Sie, ein vortrainiertes MXNet Modell zu verwenden, um Bildklassifizierung in Echtzeit mit Multi Model Server (MMS) durchzuführen. MMS ist ein flexibles easy-to-use Tool zur Bereitstellung von Deep-Learning-Modellen, die mit einem beliebigen Framework für maschinelles Lernen oder Deep Learning trainiert wurden. Dieses Tutorial beinhaltet einen Kompilierungsschritt mit AWS Neuron und eine Implementierung von MMS mit. MXNet

Weitere Informationen zum Neuron SDK finden Sie in der Neuron SDK-Dokumentation AWS .

Inhalt

- Voraussetzungen
- Aktivieren der Conda-Umgebung
- Herunterladen des Beispielcodes
- Kompilieren des Modells
- Ausführen der Inferenz

Voraussetzungen

Bevor Sie dieses Tutorial verwenden, müssen Sie die Einrichtungsschritte in Starten einer DLAMI-Instanz mit Neuron AWS abgeschlossen haben. Sie sollten auch mit Deep Learning und der Verwendung des DLAMI vertraut sein.

Aktivieren der Conda-Umgebung

Aktivieren Sie die MXNet -Neuron Conda-Umgebung mit dem folgenden Befehl:

```
source activate aws_neuron_mxnet_p36
```

Führen Sie Folgendes aus, um die aktuelle Conda-Umgebung zu verlassen:

```
source deactivate
```

Herunterladen des Beispielcodes

Um dieses Beispiel auszuführen, laden Sie den Beispielcode mit den folgenden Befehlen herunter:

```
git clone https://github.com/awslabs/multi-model-server
cd multi-model-server/examples/mxnet_vision
```

Kompilieren des Modells

Erstellen Sie das Python-Skript namens `multi-model-server-compile.py` mit folgendem Inhalt. Dieses Skript kompiliert das Modell ResNet 50 für das Inferentia-Gerätеziel.

```
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)
```

Verwenden Sie den folgenden Befehl, um das Modell zu kompilieren:

```
python multi-model-server-compile.py
```

Die Ausgabe sollte folgendermaßen aussehen:

```
...
```

```
[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!
```

Erstellen Sie eine Datei namens `signature.json` mit dem folgenden Inhalt, um den Eingabenamen und das Shape zu konfigurieren:

```
{
  "inputs": [
    {
      "data_name": "data",
      "data_shape": [
        1,
        3,
        224,
        224
      ]
    }
  ]
}
```

Sie können die Datei `synset.txt` mit dem folgenden Befehl herunterladen. Diese Datei ist eine Liste mit Namen für ImageNet Vorhersageklassen.

```
curl -O https://s3.amazonaws.com/model-server/model_archive_1.0/examples/
squeezenet_v1.1/synset.txt
```

Erstellen Sie eine benutzerdefinierte Service-Klasse unter Verwendung der Vorlage im Ordner `model_server_template`. Kopieren Sie die Vorlage mit dem folgenden Befehl in das aktuelle Arbeitsverzeichnis:

```
cp -r ../model_service_template/* .
```

Bearbeiten Sie das Modul `mxnet_model_service.py`, indem Sie den Kontext `mx.cpu()` wie folgt durch den Kontext `mx.neuron()` ersetzen. Sie müssen auch die überflüssige Datenkopie für auskommentierenmodel_input, weil MXNet -Neuron das NDArray und Gluon nicht unterstützt. 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]
```

Erstellen Sie mit den folgenden Befehlen ein Paket des Modells mit model-archiver:

```
cd ~/multi-model-server/examples
model-archiver --force --model-name resnet-50_compiled --model-path mxnet_vision --
handler mxnet_vision_service:handle
```

Ausführen der Inferenz

Starten Sie den Multi Model Server und laden Sie das Modell, das die RESTful API verwendet, mithilfe der folgenden Befehle. Stellen Sie sicher, dass neuron-rtd mit den Standardeinstellungen ausgeführt wird.

```
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
```

Führen Sie die Inferenz mit den folgenden Befehlen mit einem Beispielbild aus:

```
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
```

Die Ausgabe sollte folgendermaßen aussehen:

```
[
  {
    "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"
  }
]
```

Um nach dem Test zu bereinigen, geben Sie über die RESTful API einen Löschbefehl aus und beenden Sie den Modellserver mit den folgenden Befehlen:

```
curl -X DELETE http://127.0.0.1:8081/models/resnet-50_compiled

multi-model-server --stop
```

Die Ausgabe sollte folgendermaßen aussehen:

```
{
  "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
```

Verwenden von PyTorch -Neuron und dem Neuron Compiler AWS

Die PyTorch -Neuron-Kompilierungs-API bietet eine Methode zum Kompilieren eines Modelldiagramms, das Sie auf einem Inferentia-Gerät ausführen können. AWS

Ein trainiertes Modell muss für ein Inferentia-Ziel kompiliert werden, bevor es auf Inf1-Instances bereitgestellt werden kann. Das folgende Tutorial kompiliert das Torchvision ResNet 50-Modell und exportiert es als gespeichertes Modul. TorchScript Dieses Modell wird dann zum Ausführen der Inferenz verwendet.

Der Einfachheit halber wird in diesem Tutorial eine Inf1-Instance sowohl für die Kompilierung als auch für die Inferenz verwendet. In der Praxis können Sie Ihr Modell mit einem anderen Instance-Typ kompilieren, z. B. mit der c5-Instance-Familie. Anschließend müssen Sie das kompilierte Modell auf dem Inf1-Inferenzserver bereitstellen. Weitere Informationen finden Sie in der AWS Neuron PyTorch SDK-Dokumentation.

Inhalt

- Voraussetzungen
- Aktivieren der Conda-Umgebung
- Resnet50-Kompilierung
- ResNet50 Inferenz

Voraussetzungen

Bevor Sie dieses Tutorial verwenden, müssen Sie die Einrichtungsschritte in Starten einer DLAMI-Instanz mit Neuron AWS abgeschlossen haben. Sie sollten auch mit Deep Learning und der Verwendung des DLAMI vertraut sein.

Aktivieren der Conda-Umgebung

Aktivieren Sie die PyTorch -Neuron Conda-Umgebung mit dem folgenden Befehl:

```
source activate aws_neuron_pytorch_p36
```

Führen Sie Folgendes aus, um die aktuelle Conda-Umgebung zu verlassen:

```
source deactivate
```

Resnet50-Kompilierung

Erstellen Sie das Python-Skript namens **`pytorch_trace_resnet50.py`** mit folgendem Inhalt. Dieses Skript verwendet die Python-API zur Kompilierung von PyTorch -Neuron, um ein ResNet -50-Modell zu kompilieren.

> **Note**
>
> Es besteht eine Abhängigkeit zwischen den Versionen von Torchvision und dem Torch-Paket, die Sie beim Kompilieren von Torchvision-Modellen beachten sollten. Diese

Abhängigkeitsregeln können über Pip verwaltet werden. Torchvision==0.6.1 entspricht der Version Torch==1.5.1, während Torchvision==0.8.2 der Version Torch==1.7.1 entspricht.

```
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")
```

Führen Sie das Kompilierungsskript aus.

```

python pytorch_trace_resnet50.py
```

Die Kompilierung wird ein paar Minuten dauern. Nach Abschluss der Kompilierung wird das kompilierte Modell wie `resnet50_neuron.pt` im lokalen Verzeichnis gespeichert.

ResNet50 Inferenz

Erstellen Sie das Python-Skript namens **`pytorch_infer_resnet50.py`** mit folgendem Inhalt. Mit diesem Skript wird ein Beispiel-Image heruntergeladen, das dazu verwendet wird, um die Inferenz für das kompilierte Modell auszuführen.

```
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) )
```

Führen Sie die Inferenz mit folgendem Befehl mit dem kompilierten Modell aus:

```
python pytorch_infer_resnet50.py
```

Die Ausgabe sollte folgendermaßen aussehen:

```
Top 5 labels:
 ['tiger', 'lynx', 'tiger_cat', 'Egyptian_cat', 'tabby']
```

Das ARM64 DLAMI

AWS ARM64 GPUs DLAMIs sind so konzipiert, dass sie eine hohe Leistung und Kosteneffizienz für Deep-Learning-Workloads bieten. Insbesondere der G5G-Instanztyp verfügt über den ARM64-basierten AWS Graviton2-Prozessor, der von Grund auf neu entwickelt AWS und dafür optimiert wurde, wie Kunden ihre Workloads in der Cloud ausführen. AWS ARM64 DLAMIs Die GPU ist mit Docker, NVIDIA Docker, NVIDIA Driver, CUDA, cuDNN, NCCL sowie beliebten Frameworks für maschinelles Lernen wie und vorkonfiguriert. TensorFlow PyTorch

Mit dem Instance-Typ G5G können Sie die Preis- und Leistungsvorteile von Graviton2 nutzen, um GPU-beschleunigte Deep-Learning-Modelle zu deutlich geringeren Kosten im Vergleich zu x86-basierten Instances mit GPU-Beschleunigung bereitzustellen.

Wählen Sie ein ARM64 DLAMI

Starten Sie eine G5g-Instance mit dem ARM64 DLAMI Ihrer Wahl.

step-by-stepAnweisungen zum Starten eines DLAMI finden Sie unter Starten und Konfigurieren eines DLAMI.

Eine Liste der neuesten ARM64 DLAMIs Versionen finden Sie in den Versionshinweisen für DLAMI.

Erste Schritte

Die folgenden Themen zeigen Ihnen, wie Sie mit der Verwendung des ARM64 DLAMI beginnen können.

Inhalt

- Verwenden der ARM64 GPU PyTorch DLAMI

Verwenden der ARM64 GPU PyTorch DLAMI

Der AWS Deep Learning AMIs ist sofort einsatzbereit mit dem Arm64-Prozessor und ist optimiert für. GPUs PyTorch Die ARM64 GPU PyTorch DLAMI umfasst eine Python-Umgebung, die mit PyTorchTorchVision, und TorchServefür Deep-Learning-Training und Inferenz-Anwendungsfälle vorkonfiguriert ist.

Inhalt

- PyTorch Python-Umgebung überprüfen
- Trainingsbeispiel ausführen mit PyTorch
- Führen Sie ein Inferenzbeispiel aus mit PyTorch

PyTorch Python-Umgebung überprüfen

Connect zu Ihrer G5g-Instance her und aktivieren Sie die Conda-Basisumgebung mit dem folgenden Befehl:

```
source activate base
```

Ihre Eingabeaufforderung sollte darauf hinweisen, dass Sie in der Conda-Basisumgebung arbeiten, die PyTorch TorchVision, und andere Bibliotheken enthält.

```
(base) $
```

Überprüfen Sie die Standard-Werkzeugpfade der PyTorch Umgebung:

```
(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)
```

Trainingsbeispiel ausführen mit PyTorch

Führen Sie einen Beispiel-MNIST-Trainingsjob aus:

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

Ihre Ausgabe sollte wie folgt aussehen:

```
...
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%)
```

Führen Sie ein Inferenzbeispiel aus mit PyTorch

Verwenden Sie die folgenden Befehle, um ein vortrainiertes densenet161-Modell herunterzuladen und die Inferenz auszuführen mit: 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
```

Ihre Ausgabe sollte wie folgt aussehen:

```
{
  "tiger_cat": 0.4693363308906555,
  "tabby": 0.4633873701095581,
  "Egyptian_cat": 0.06456123292446136,
  "lynx": 0.0012828150065615773,
  "plastic_bag": 0.00023522898778133094
}
```

Verwenden Sie die folgenden Befehle, um die Registrierung des densenet161-Modells aufzuheben und den Server anzuhalten:

```
curl -X DELETE http://localhost:8081/models/densenet161/1.0
torchserve --stop
```

Ihre Ausgabe sollte wie folgt aussehen:

```
{
  "status": "Model \"densenet161\" unregistered"
}
TorchServe has stopped.
```

Inferenz

Dieser Abschnitt enthält Tutorials zum Ausführen von Inferenzen mit den Frameworks und Tools des DLAMI.

Inferenz-Tools

- TensorFlow Servieren

Modellbereitstellung

Im Folgenden sind die Modell-Serving-Optionen aufgeführt, die auf dem Deep Learning-AMI mit Conda installiert sind. Klicken Sie auf eine der Optionen, um zu erfahren, wie Sie diese verwenden.

Themen

- TensorFlow Servieren
- TorchServe

TensorFlow Servieren

TensorFlow Serving ist ein flexibles, leistungsstarkes Serviersystem für Modelle des maschinellen Lernens.

Das `tensorflow-serving-api` ist mit einem einzigen Framework DLAMI vorinstalliert. Um Tensorflow Serving zu verwenden, aktivieren Sie zunächst die Umgebung. TensorFlow

```
$ source /opt/tensorflow/bin/activate
```

Verwenden Sie anschließend den von Ihnen bevorzugten Texteditor, um ein Skript mit folgendem Inhalt zu erstellen. Geben Sie ihr den Namen `test_train_mnist.py`. Auf dieses Skript wird im TensorFlow Tutorial verwiesen, in dem ein maschinelles Lernmodell für neuronale Netzwerke trainiert und evaluiert wird, das Bilder klassifiziert.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
```

```
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Führen Sie nun das Skript aus und geben Sie Server-Standort sowie Port und Dateinamen des Husky-Bilds als Parameter weiter.

```
$ /opt/tensorflow/bin/python3 test_train_mnist.py
```

Haben Sie Geduld, da dieses Skript kann eine Weile brauchen kann, bevor es etwas ausgibt. Wenn das Training abgeschlossen ist, sollten Sie Folgendes sehen:

```
I0000 00:00:1739482012.389276    4284 device_compiler.h:188] Compiled cluster using
 XLA!  This line is logged at most once for the lifetime of the process.
1875/1875 [==============================] - 24s 2ms/step - loss: 0.2973 - accuracy:
 0.9134
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1422 - accuracy:
 0.9582
Epoch 3/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.1076 - accuracy:
 0.9687
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0872 - accuracy:
 0.9731
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0731 - accuracy:
 0.9771
313/313 [==============================] - 0s 1ms/step - loss: 0.0749 - accuracy:
 0.9780
```

Weitere Funktionen und Beispiele

Wenn Sie mehr über TensorFlow Serving erfahren möchten, besuchen Sie die TensorFlow Website.

TorchServe

TorchServe ist ein flexibles Tool zur Bereitstellung von Deep-Learning-Modellen, aus denen exportiert wurden PyTorch. TorchServe ist mit dem Deep Learning AMI mit Conda vorinstalliert.

Weitere Informationen zur Verwendung TorchServe finden Sie in der Dokumentation zu Model Server. PyTorch

Topics

Stellen Sie ein Bildklassifizierungsmodell bereit auf TorchServe

Dieses Tutorial zeigt, wie Sie ein Bildklassifizierungsmodell mit bereitstellen TorchServe. Es verwendet ein DenseNet -161-Modell, das von bereitgestellt wird PyTorch. Sobald der Server läuft, wartet er auf Vorhersageanfragen. Wenn Sie ein Bild hochladen, in diesem Fall das Bild eines Kätzchens, gibt der Server eine Vorhersage der fünf am besten passenden Klassen aus den Klassen zurück, für die das Modell trainiert wurde.

Um ein Beispiel für ein Bildklassifizierungsmodell bereitzustellen TorchServe

1. Stellen Sie mit Conda v34 oder höher eine Connect zu einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance mit Deep Learning AMI her.
2. Aktivieren Sie die Umgebung`pytorch_p310`.

   ```
   source activate pytorch_p310
   ```

3. Klonen Sie das TorchServe Repository und erstellen Sie dann ein Verzeichnis zum Speichern Ihrer Modelle.

   ```
   git clone https://github.com/pytorch/serve.git
   mkdir model_store
   ```

4. Archivieren Sie das Modell mit dem Modellarchiver. Der `extra-files` Parameter verwendet eine Datei aus dem `TorchServe` Repo. Aktualisieren Sie daher den Pfad, falls erforderlich. Weitere Informationen zum Modellarchiver finden Sie unter Torch Model Archiver for. TorchServe

   ```
   wget https://download.pytorch.org/models/densenet161-8d451a50.pth
   ```

```
torch-model-archiver --model-name densenet161 --version 1.0 --model-file ./
serve/examples/image_classifier/densenet_161/model.py --serialized-file
 densenet161-8d451a50.pth --export-path model_store --extra-files ./serve/examples/
image_classifier/index_to_name.json --handler image_classifier
```

5. Ausführen TorchServe , um einen Endpunkt zu starten. Durch das `> /dev/null` Hinzufügen wird die Protokollausgabe gestoppt.

```
torchserve --start --ncs --model-store model_store --models densenet161.mar > /dev/
null
```

6. Laden Sie ein Bild eines Kätzchens herunter und senden Sie es an den TorchServe Predict-Endpunkt:

```
curl -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg
curl http://127.0.0.1:8080/predictions/densenet161 -T kitten.jpg
```

   Der Vorhersage-Endpunkt gibt eine Vorhersage in JSON zurück, die den folgenden fünf wichtigsten Vorhersagen ähnelt, wobei das Bild mit einer Wahrscheinlichkeit von 47% eine ägyptische Katze enthält, gefolgt von einer Wahrscheinlichkeit von 46%, dass es sich um eine Tabbykatze handelt.

```
{
 "tiger_cat": 0.46933576464653015,
 "tabby": 0.463387668132782,
 "Egyptian_cat": 0.0645613968372345,
 "lynx": 0.0012828196631744504,
 "plastic_bag": 0.00023323058849200606
}
```

7. Wenn Sie mit dem Testen fertig sind, beenden Sie den Server:

```
torchserve --stop
```

Andere Beispiele

TorchServe hat eine Vielzahl von Beispielen, die Sie auf Ihrer DLAMI-Instanz ausführen können. Sie können sie auf der Seite mit den Beispielen für das TorchServe Projekt-Repository einsehen.

Mehr Informationen

Weitere TorchServe Dokumentation, einschließlich der Einrichtung TorchServe mit Docker und der neuesten TorchServe Funktionen, finden Sie auf GitHubder TorchServe Projektseite unter.

Ihr DLAMI aufrüsten

Hier finden Sie Informationen zum Upgrade Ihres DLAMI und Tipps zur Aktualisierung der Software auf Ihrem DLAMI.

Halten Sie Ihr Betriebssystem und sonstige installierte Software immer auf dem neuesten Stand, indem Sie Patches und Updates anwenden, sobald diese verfügbar werden.

Wenn Sie Amazon Linux oder Ubuntu verwenden, werden Sie bei der Anmeldung bei Ihrem DLAMI benachrichtigt, ob Updates verfügbar sind, und Sie erhalten Anweisungen zur Aktualisierung. Weitere Informationen zur Wartung von Amazon Linux finden Sie unter Instance-Software aktualisieren. Für Ubuntu-Instances lesen Sie in der offiziellen Ubuntu-Dokumentation nach.

Unter Windows überprüfen Sie Windows Update regelmäßig auf Software- und Sicherheits-Updates. Wenn Sie möchten, können neue Updates auch automatisch angewendet.

> **Important**
>
> Informationen zu den Sicherheitslücken Meltdown und Spectre und dazu, wie Sie Ihr Betriebssystem patchen können, um diese zu beheben, finden Sie im Security Bulletin -2018-013. AWS

Themen

- Upgrade auf eine neue DLAMI-Version
- Tipps für Software-Updates
- Erhalten Sie Benachrichtigungen über neue Updates

Upgrade auf eine neue DLAMI-Version

Die System-Images von DLAMI werden regelmäßig aktualisiert, um die Vorteile neuer Deep-Learning-Framework-Versionen, CUDA- und andere Softwareupdates sowie Leistungsoptimierungen zu nutzen. Wenn Sie ein DLAMI schon länger verwenden und die Vorteile eines Updates nutzen möchten, müssen Sie eine neue Instanz starten. Sie müssten auch manuell Datensätze, Kontrollpunkte oder andere wichtige Daten übertragen. Stattdessen können Sie Amazon EBS verwenden, um Ihre Daten aufzubewahren und sie an ein neues DLAMI anzuhängen. Auf diese

Weise können Sie häufig Aktualisierungen durchführen und dabei die Zeit, die zum Übertragen Ihrer Daten erforderlich ist, minimieren.

> **Note**
>
> Wenn Sie Amazon EBS-Volumes anhängen und zwischen ihnen verschieben DLAMIs, müssen Sie sowohl das Volume als auch das DLAMIs neue Volume in derselben Availability Zone haben.

1. Verwenden Sie Amazon EC2console , um ein neues Amazon EBS-Volume zu erstellen. Eine detaillierte Anleitung finden Sie unter Erstellen eines Amazon EBS-Volumes.
2. Hängen Sie Ihr neu erstelltes Amazon EBS-Volume an Ihr vorhandenes DLAMI an. Eine detaillierte Anleitung finden Sie unter Anhängen eines Amazon EBS-Volumes.
3. Übertragen Sie Ihre Daten, wie z. B. Datensätze, Kontrollpunkte und Konfigurationsdateien.
4. Starten Sie ein DLAMI. Detaillierte Anweisungen finden Sie unter Einrichtung einer DLAMI-Instanz.
5. Trennen Sie das Amazon EBS-Volume von Ihrem alten DLAMI. Eine ausführliche Anleitung finden Sie unter Trennen eines Amazon EBS-Volumes.
6. Hängen Sie das Amazon EBS-Volume an Ihr neues DLAMI an. Folgen Sie den Anweisungen aus Schritt 2 zum Verknüpfen des Volumes.
7. Nachdem Sie überprüft haben, ob Ihre Daten auf Ihrem neuen DLAMI verfügbar sind, beenden und beenden Sie Ihr altes DLAMI. Detaillierte Anweisungen zur Bereinigung finden Sie unter Bereinigen einer DLAMI-Instanz.

Tipps für Software-Updates

Von Zeit zu Zeit möchten Sie möglicherweise die Software auf Ihrem DLAMI manuell aktualisieren. Allgemein empfiehlt es sich, `pip` zum Aktualisieren von Python-Paketen zu verwenden. Sie sollten es auch verwendenpip, um Pakete innerhalb einer Conda-Umgebung auf dem Deep Learning AMI mit Conda zu aktualisieren. Weitere Anweisungen zum Upgrade und zur Installation finden Sie auf der jeweiligen Framework- oder Software-Website.

> **Note**
>
> Wir können nicht garantieren, dass ein Paket-Update erfolgreich sein wird. Der Versuch, ein Paket in einer Umgebung mit inkompatiblen Abhängigkeiten zu aktualisieren, kann zu einem Fehler führen. In einem solchen Fall sollten Sie sich an den Bibliotheksbetreuer wenden, um zu erfahren, ob es möglich ist, die Paketabhängigkeiten zu aktualisieren. Alternativ können Sie versuchen, die Umgebung so zu ändern, dass das Update möglich ist. Diese Änderung wird jedoch wahrscheinlich bedeuten, dass bestehende Pakete entfernt oder aktualisiert werden, was bedeutet, dass wir die Stabilität dieser Umgebung nicht mehr garantieren können.

Das AWS Deep Learning AMIs wird mit vielen Conda-Umgebungen und vielen vorinstallierten Paketen geliefert. Aufgrund der Anzahl der vorinstallierten Pakete ist es schwierig, eine Reihe von Paketen zu finden, die garantiert kompatibel sind. Möglicherweise wird die Warnung „Die Umgebung ist inkonsistent, bitte überprüfen Sie den Paketplan sorgfältig“ angezeigt. DLAMI stellt sicher, dass alle von DLAMI bereitgestellten Umgebungen korrekt sind, kann jedoch nicht garantieren, dass vom Benutzer installierte Pakete korrekt funktionieren.

Erhalten Sie Benachrichtigungen über neue Updates

> **Note**
>
> AWS Deep Learning AMIs hat einen wöchentlichen Veröffentlichungsrhythmus für Sicherheitspatches. Für diese inkrementellen Sicherheitspatches werden Versionsbenachrichtigungen gesendet, obwohl sie möglicherweise nicht in den offiziellen Versionshinweisen enthalten sind.

Sie können Benachrichtigungen erhalten, wenn ein neues DLAMI veröffentlicht wird. Benachrichtigungen werden mithilfe von Amazon SNS mit dem folgenden Thema veröffentlicht.

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

Nachrichten werden hier veröffentlicht, wenn ein neues DLAMI veröffentlicht wird. Die Version, die Metadaten und die regionalen AMI-IDs des AMI werden in der Nachricht enthalten sein.

Diese Nachrichten können mit verschiedenen Methoden empfangen werden. Wir empfehlen, dass Sie die folgende Methode verwenden.

1. Öffnen Sie die Amazon-SNS-Konsole.
2. Ändern Sie in der Navigationsleiste bei Bedarf die AWS Region auf USA West (Oregon). Sie müssen die Region auswählen, in der die SNS-Benachrichtigung, die Sie abonnieren, erstellt wurde.
3. Wählen Sie im Navigationsbereich Abonnements, Abonnement erstellen aus.
4. Führen Sie im Dialogfeld Create subscription die folgenden Schritte aus:
    a. Kopieren Sie für das Thema ARN den folgenden Amazon-Ressourcennamen (ARN) und fügen Sie ihn ein: **`arn:aws:sns:us-west-2:767397762724:dlami-updates`**
    b. Wählen Sie unter Protokoll eines aus [Amazon SQS, AWS Lamda, Email, Email-JSON]
    c. Geben Sie für Endpoint die E-Mail-Adresse oder den Amazon-Ressourcennamen (ARN) der Ressource ein, die Sie für den Empfang der Benachrichtigungen verwenden werden.
    d. Wählen Sie Create subscription (Abonnement erstellen) aus.
5. Sie erhalten eine Bestätigungs-E-Mail mit dem Betreff AWS Benachrichtigung — Abonnementbestätigung. Öffnen Sie die E-Mail und wählen Sie Confirm subscription aus, um Ihr Abonnement abzuschließen.

Sicherheit in AWS Deep Learning AMIs

Cloud-Sicherheit hat AWS höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS-Services in der läuft AWS Cloud. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS Deep Learning AMIs, finden Sie unter AWS Services im Umfang nach Compliance-Programmen AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS-Service , was Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, einschließlich der Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von DLAMI anwenden können. In den folgenden Themen erfahren Sie, wie Sie DLAMI konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere verwenden können AWS-Services , die Ihnen helfen, Ihre DLAMI-Ressourcen zu überwachen und zu sichern.

Weitere Informationen finden Sie unter Sicherheit bei Amazon EC2 im EC2 Amazon-Benutzerhandbuch.

Themen

- Datenschutz in AWS Deep Learning AMIs
- Identitäts- und Zugriffsmanagement für AWS Deep Learning AMIs
- Konformitätsvalidierung für AWS Deep Learning AMIs
- Resilienz in AWS Deep Learning AMIs
- Sicherheit der Infrastruktur in AWS Deep Learning AMIs

- AWS Deep Learning AMIs Instanzen überwachen

Datenschutz in AWS Deep Learning AMIs

Das Modell der AWS gemeinsamen Verantwortung und geteilter Verantwortung gilt für den Datenschutz in AWS Deep Learning AMIs. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit DLAMI oder anderen AWS-Services über die Konsole AWS CLI, API oder arbeiten. AWS SDKs Alle

Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Identitäts- und Zugriffsmanagement für AWS Deep Learning AMIs

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu kontrollieren. AWS IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um DLAMI-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Weitere Informationen zur Identitäts- und Zugriffsverwaltung finden Sie unter Identitäts- und Zugriffsmanagement für Amazon EC2.

Themen

- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- IAM mit Amazon EMR

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode für die Selbstsignierung von Anforderungen finden Sie unter AWS Signature Version 4 für API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung (MFA) in IAM im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb von Ihnen AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer

gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management Console, können Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Methoden für die Übernahme einer Rolle im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden

zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto.

Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter Auswählen zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der

identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.

- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos Entitäten. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Ressourcenkontrollrichtlinien (RCPs) — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter Resource Control Policies (RCPs) im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

IAM mit Amazon EMR

Sie können IAM mit Amazon EMR verwenden, um Benutzer, AWS Ressourcen, Gruppen, Rollen und Richtlinien zu definieren. Sie können auch steuern, auf welche AWS-Services Benutzer und Rollen zugreifen können.

Weitere Informationen zur Verwendung von IAM mit Amazon EMR finden Sie unter AWS Identity and Access Management für Amazon EMR.

Konformitätsvalidierung für AWS Deep Learning AMIs

Externe Prüfer bewerten die Sicherheit und Einhaltung von Vorschriften im AWS Deep Learning AMIs Rahmen mehrerer AWS Compliance-Programme. Informationen zu den unterstützten Compliance-Programmen finden Sie unter Konformitätsvalidierung für Amazon EC2.

Eine Liste der AWS-Services im Umfang der spezifischen Compliance-Programme enthaltenen AWS Services finden Sie unter Services im Bereich nach Compliance-Programm AWS . Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten in AWS Artifact .

Ihre Compliance-Verantwortung bei der Verwendung von DLAMI richtet sich nach der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften. AWS bietet die folgenden Ressourcen zur Unterstützung bei der Einhaltung von Vorschriften:

- Schnellstartanleitungen für Sicherheit und Compliance – In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Bereitstellung von sicherheits- und konformitätsorientierten Basisumgebungen auf AWS angegeben.
- AWS Ressourcen zur AWS von Vorschriften — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- Bewertung von Ressourcen anhand von AWS Config Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre

AWS Ressourcen zu bewerten und Ihre Einhaltung der Sicherheitsstandards und Best Practices der Sicherheitsbranche zu überprüfen.

Resilienz in AWS Deep Learning AMIs

Die AWS globale Infrastruktur basiert auf Availability AWS-Regionen Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Informationen zu EC2 Amazon-Funktionen zur Unterstützung Ihrer Datenausfallsicherheit und Ihrer Sicherungsanforderungen finden Sie unter Resilience EC2 in Amazon im EC2 Amazon-Benutzerhandbuch.

Sicherheit der Infrastruktur in AWS Deep Learning AMIs

Die Infrastruktursicherheit von AWS Deep Learning AMIs wird von Amazon unterstützt EC2. Weitere Informationen finden Sie unter Infrastruktursicherheit in Amazon EC2 im EC2 Amazon-Benutzerhandbuch.

AWS Deep Learning AMIs Instanzen überwachen

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer AWS Deep Learning AMIs Instanz und Ihrer anderen AWS Lösungen. Ihre DLAMI-Instance verfügt über mehrere GPU-Überwachungstools, darunter ein Hilfsprogramm, das Statistiken zur GPU-Nutzung an Amazon meldet. CloudWatch Weitere Informationen finden Sie unter GPU-Überwachung und -Optimierung und unter Überwachen von EC2 Amazon-Ressourcen im EC2 Amazon-Benutzerhandbuch.

Deaktivierung der Nutzungsverfolgung für DLAMI-Instanzen

Die folgenden AWS Deep Learning AMIs Betriebssystemverteilungen enthalten Code, mit dem Instanztyp, Instanz-ID, DLAMI-Typ und Betriebssysteminformationen erfasst werden können AWS .

> **Note**
>
> AWS sammelt oder speichert keine anderen Informationen über das DLAMI, wie z. B. die Befehle, die Sie innerhalb des DLAMI verwenden.

- Amazon Linux 2
- Amazon Linux 2023
- Ubuntu 20.04
- Ubuntu 22.04

Um die Nutzungsverfolgung zu deaktivieren

Wenn Sie möchten, können Sie die Nutzungsverfolgung für eine neue DLAMI-Instanz deaktivieren. Um sich abzumelden, müssen Sie Ihrer EC2 Amazon-Instance beim Start ein Tag hinzufügen. Das Tag sollte den Schlüssel verwenden, dessen `OPT_OUT_TRACKING` zugeordneter Wert auf gesetzt ist`true`. Weitere Informationen finden Sie unter Taggen Ihrer EC2 Amazon-Ressourcen im EC2 Amazon-Benutzerhandbuch.

DLAMI-Supportrichtlinie

Hier finden Sie Einzelheiten zur Support-Richtlinie für AWS Deep Learning AMIs (DLAMI).

Eine Liste der DLAMI-Frameworks und Betriebssysteme, die AWS derzeit unterstützt werden, finden Sie auf der Seite DLAMI-Supportrichtlinie. Die folgende Terminologie gilt für alle, die auf der Seite mit den Support-Richtlinien und dieser Seite DLAMIs erwähnt werden:

- Die aktuelle Version gibt die Framework-Version im Format x.y.z an. In diesem Format bezieht sich x auf die Hauptversion, y auf die Nebenversion und z auf die Patch-Version. Beispielsweise ist für TensorFlow 2.10.1 die Hauptversion 2, die Nebenversion 10 und die Patch-Version 1.
- Das Ende des Patches gibt an, wie lange eine bestimmte Framework- oder Betriebssystemversion AWS unterstützt wird.

Ausführliche Informationen zu bestimmten Themen DLAMIs finden Sie unter AMIs Versionshinweise zu Deep Learning.

DLAMI-Unterstützung FAQs

- Welche Framework-Versionen erhalten Sicherheitspatches?
- Welches Betriebssystem erhält Sicherheitspatches?
- Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden?
- Welche Bilder erhalten neue SageMaker AWS KI/Funktionen?
- Wie ist die aktuelle Version in der Tabelle Unterstützte Frameworks definiert?
- Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Version aufgeführt ist?
- Werden frühere Patch-Versionen einer Framework-Version DLAMIs unterstützt?
- Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version?
- Wie oft werden neue Images veröffentlicht?
- Wird meine Instance an Ort und Stelle gepatcht, während mein Workload läuft?
- Was passiert, wenn eine neue gepatchte oder aktualisierte Framework-Version verfügbar ist?
- Werden Abhängigkeiten aktualisiert, ohne die Framework-Version zu ändern?
- Wann endet der aktive Support für meine Framework-Version?

- Werden Images mit Framework-Versionen, die nicht mehr aktiv verwaltet werden, gepatcht?
- Wie verwende ich eine ältere Framework-Version?
- Wie bleibe ich up-to-date bei Support-Änderungen an Frameworks und deren Versionen?
- Benötige ich eine kommerzielle Lizenz, um das Anaconda Repository nutzen zu können?

Welche Framework-Versionen erhalten Sicherheitspatches?

Wenn sich die Framework-Version in der Tabelle AWS Deep Learning AMIs Support-Richtlinien unter Unterstützte Framework-Versionen befindet, erhält sie Sicherheitspatches.

Welches Betriebssystem erhält Sicherheitspatches?

Wenn das Betriebssystem in der Tabelle mit den AWS Deep Learning AMIs Support-Richtlinien unter Unterstützte Betriebssystemversionen aufgeführt ist, erhält es Sicherheitspatches.

Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden?

Wir veröffentlichen DLAMIs bald nach der Veröffentlichung neuer Versionen von TensorFlow und PyTorch neue. Dazu gehören Hauptversionen, Haupt-Nebenversionen und major-minor-patch Versionen von Frameworks. Wir aktualisieren auch Images, wenn neue Versionen von Treibern und Bibliotheken verfügbar werden. Weitere Informationen zur Image-Wartung finden Sie unter Wann endet der aktive Support für meine Framework-Version?

Welche Bilder erhalten neue SageMaker AWS KI/Funktionen?

Neue Funktionen werden normalerweise in der neuesten Version von DLAMIs for PyTorch und TensorFlow veröffentlicht. Einzelheiten zu neuen SageMaker KI oder AWS Funktionen finden Sie in den Versionshinweisen für ein bestimmtes Bild. Eine Liste der verfügbaren DLAMIs Versionen finden Sie in den Versionshinweisen für DLAMI. Weitere Informationen zur Image-Wartung finden Sie unter Wann endet der aktive Support für meine Framework-Version?

Wie ist die aktuelle Version in der Tabelle Unterstützte Frameworks definiert?

Die aktuelle Version in der Tabelle mit den AWS Deep Learning AMIs Support-Richtlinien bezieht sich auf die neueste Framework-Version, AWS die am verfügbar ist GitHub. Jede neueste Version enthält

Updates für die Treiber, Bibliotheken und relevanten Pakete im DLAMI. Informationen zur Image-Wartung finden Sie unter Wann endet der aktive Support für meine Framework-Version?

Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Version aufgeführt ist?

Wenn Sie eine Version ausführen, die nicht in der Tabelle mit den AWS Deep Learning AMIs Support-Richtlinien aufgeführt ist, verfügen Sie möglicherweise nicht über die aktuellsten Treiber, Bibliotheken und relevanten Pakete. Für eine weitere up-to-date Version empfehlen wir Ihnen, auf eines der unterstützten Frameworks oder Betriebssysteme zu aktualisieren, die mit dem neuesten DLAMI Ihrer Wahl verfügbar sind. Eine Liste der verfügbaren DLAMIs Versionen finden Sie in den Versionshinweisen für DLAMI.

Werden frühere Patch-Versionen einer Framework-Version DLAMIs unterstützt?

Nein. Wir Support die neueste Patch-Version der neuesten Hauptversion jedes Frameworks, die 365 Tage nach der ersten GitHub Veröffentlichung veröffentlicht wurde, wie in der Tabelle mit den AWS Deep Learning AMIs Support-Richtlinien angegeben. Weitere Informationen finden Sie unter Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Version aufgeführt ist?.

Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version?

Um ein DLAMI mit der neuesten Framework-Version zu verwenden, können Sie AWS CLI- oder SSM-Parameter verwenden, um die DLAMI-ID abzurufen und damit das DLAMI über die Konsole zu starten. EC2 Beispiele für AWS CLI- oder SSM-Parameterbefehle zum Abrufen der AWS Deep Learning AMIs ID finden Sie auf der Seite mit den DLAMI-Versionshinweisen für einzelne Framework-DLAMI-Versionshinweise. Die von Ihnen gewählte Framework-Version muss in der Tabelle AWS Deep Learning AMIs Support-Richtlinien unter Unterstützte Framework-Versionen aufgeführt sein.

Wie oft werden neue Images veröffentlicht?

Die Bereitstellung aktualisierter Patch-Versionen hat für uns höchste Priorität. Wir erstellen routinemäßig zum frühestmöglichen Zeitpunkt gepatchte Images. Wir suchen nach neu gepatchten Framework-Versionen (z. B. TensorFlow 2.9 bis TensorFlow 2.9.1) und neue Nebenversionen (z. B. TensorFlow 2.9 bis TensorFlow 2.10) und stellen sie so bald wie möglich zur Verfügung. Wenn

eine vorhandene Version von mit einer neuen Version von CUDA veröffentlicht TensorFlow wird, veröffentlichen wir ein neues DLAMI für diese Version von TensorFlow mit Unterstützung für die neue CUDA-Version.

Wird meine Instance an Ort und Stelle gepatcht, während mein Workload läuft?

Nein. Patch-Updates für DLAMI sind keine „direkten“ Updates.

Sie müssen eine neue EC2 Instanz einschalten, Ihre Workloads und Skripts migrieren und dann Ihre vorherige Instanz ausschalten.

Was passiert, wenn eine neue gepatchte oder aktualisierte Framework-Version verfügbar ist?

Um über Änderungen in DLAMI informiert zu werden, abonnieren Sie bitte die Benachrichtigungen für das entsprechende DLAMI, siehe Benachrichtigungen über neue Updates erhalten.

Werden Abhängigkeiten aktualisiert, ohne die Framework-Version zu ändern?

Wir aktualisieren Abhängigkeiten, ohne die Framework-Version zu ändern. Wenn ein Abhängigkeitsupdate jedoch zu einer Inkompatibilität führt, erstellen wir ein Image mit einer anderen Version. Aktuelle Abhängigkeitsinformationen finden Sie in den Versionshinweisen für DLAMI.

Wann endet der aktive Support für meine Framework-Version?

DLAMI-Bilder sind unveränderlich. Sobald sie erstellt wurden, ändern sie sich nicht. Es gibt vier Hauptgründe, warum der aktive Support für eine Framework-Version endet:

- Upgrades der Framework-Version (Patch)
- AWS Sicherheitspatches
- Ende des Patch-Datums (Alterung abgelaufen)
- Abhängigkeit end-of-support

> **Note**
>
> Aufgrund der Häufigkeit von Versionspatch-Upgrades und Sicherheitspatches empfehlen wir, die Seite mit den Versionshinweisen für Ihr DLAMI häufig zu überprüfen und ein Upgrade durchzuführen, wenn Änderungen vorgenommen werden.

Upgrades der Framework-Version (Patch)

Wenn Sie einen DLAMI-Workload haben, der auf TensorFlow 2.7.0 basiert und Version 2.7.1 TensorFlow veröffentlicht GitHub, dann AWS veröffentlicht Sie ein neues DLAMI mit 2.7.1. TensorFlow Die vorherigen Images mit 2.7.0 werden nicht mehr aktiv gepflegt, sobald das neue Image mit 2.7.1 veröffentlicht wird. TensorFlow Das DLAMI mit TensorFlow 2.7.0 erhält keine weiteren Patches. Die Seite mit den DLAMI-Versionshinweisen für TensorFlow 2.7 wird dann mit den neuesten Informationen aktualisiert. Es gibt keine eigene Seite mit den Versionshinweisen für jeden kleineren Patch.

Neue, die aufgrund DLAMIs von Patch-Upgrades erstellt wurden, werden mit einer neuen AMI-ID gekennzeichnet.

AWS Sicherheitspatches

Wenn Sie einen Workload haben, der auf einem Image mit TensorFlow 2.7.0 basiert und AWS Sie einen Sicherheitspatch erstellen, wird eine neue Version des DLAMI für 2.7.0 veröffentlicht. TensorFlow Die vorherige Version der Images mit TensorFlow 2.7.0 wird nicht mehr aktiv gepflegt. Weitere Informationen finden Sie unter Schritte Wird meine Instance an Ort und Stelle gepatcht, während mein Workload läuft? zur Suche nach der neuesten DLAMI-Version finden Sie unter Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version?

Neue, die aufgrund DLAMIs von Patch-Upgrades erstellt wurden, werden mit einer neuen AMI-ID gekennzeichnet.

Ende des Patch-Datums (Alterung abgelaufen)

DLAMIs das Ende des Patches wurde 365 Tage nach dem GitHub Veröffentlichungsdatum erreicht.

Wenn für Multi-Frameworks DLAMIs eine der Framework-Versionen aktualisiert wird, ist ein neues DLAMI mit der aktualisierten Version erforderlich. Das DLAMI mit der alten Framework-Version wird nicht mehr aktiv gepflegt.

> **Important**
>
> Wir machen eine Ausnahme, wenn es ein größeres Framework-Update gibt. Wenn beispielsweise TensorFlow 1.15 auf TensorFlow 2.0 aktualisiert wird, unterstützen wir weiterhin die neueste Version von TensorFlow 1.15 für einen Zeitraum von zwei Jahren ab dem Datum der GitHub Veröffentlichung oder sechs Monate, nachdem das Origin-Framework-Wartungsteam den Support eingestellt hat, je nachdem, welches Datum früher liegt.

Abhängigkeit end-of-support

Wenn Sie einen Workload auf einem TensorFlow 2.7.0 DLAMI-Image mit Python 3.6 ausführen und diese Version von Python für markiert ist end-of-support, werden alle auf Python 3.6 basierenden DLAMI-Images nicht mehr aktiv verwaltet. Ebenso werden alle DLAMI-Images, die von Ubuntu 16.04 abhängig sind end-of-support, nicht mehr aktiv verwaltet, wenn eine Betriebssystemversion wie Ubuntu 16.04 markiert ist.

Werden Images mit Framework-Versionen, die nicht mehr aktiv verwaltet werden, gepatcht?

Nein. Für Bilder, die nicht mehr aktiv gepflegt werden, wird es keine Neuveröffentlichungen geben.

Wie verwende ich eine ältere Framework-Version?

Um ein DLAMI mit einer älteren Framework-Version zu verwenden, rufen Sie die DLAMI-ID ab und verwenden Sie sie, um das DLAMI über die Konsole zu starten. EC2 AWS CLI-Befehle zum Abrufen der AMI-ID finden Sie auf der Seite mit den Versionshinweisen in den DLAMI-Versionshinweisen für einzelne Frameworks.

Wie bleibe ich up-to-date bei Support-Änderungen an Frameworks und deren Versionen?

Bleiben Sie up-to-date bei den DLAMI-Frameworks und -Versionen, indem Sie die Tabelle mit den AWS Deep Learning AMIs Framework Support Policys und die DLAMI-Versionshinweise verwenden.

Benötige ich eine kommerzielle Lizenz, um das Anaconda Repository nutzen zu können?

Anaconda hat für bestimmte Benutzer auf ein kommerzielles Lizenzmodell umgestellt. Aktiv gepflegt DLAMIs wurden vom Anaconda-Kanal auf die öffentlich verfügbare Open-Source-Version von Conda (Conda-Forge) migriert.

Tabelle mit den Richtlinien Support den DLAMI-Support

Weitere Informationen finden Sie in der Support-Richtlinie.

Unterstützte Framework-Versionen

Framework	Aktuelle Version	CUDA-Version	GitHub GA	Ende des Patches
PyTorch	2.7.0	12.8	2025-04-23	2026-04-23
PyTorch	2.6.0	12.6	2025-01-29	2026-01-29
PyTorch	2.5.1	12.4	2024-11-24	24.11.2025
PyTorch	2.4.1	12.4	24-07-24	24.07.2025
TensorFlow	2.18.0	12,5	2024-10-24	2025-10-24
TensorFlow	2.17.0	12.3	2024-11-07	2025-11-07

Unterstützte Betriebssystemversionen

Betriebssystem	Ende des Patches
Amazon Linux 2023	2029-06-30
Amazon Linux 2	2026-06-30
Ubuntu 24.04	2029-04-30
Ubuntu 22.04	30.04.2027

Nicht unterstützte Framework-Versionen

Die in dieser Tabelle aufgeführten Versionen werden 2 Jahre nach ihrem Support-Datum angezeigt.

Framework	Aktuelle Version	CUDA-Version	GitHub GA	Ende des Patches
PyTorch	2.3.0	12.1	24.04.2024	24.04.2025
PyTorch	2.2.0	12,1	2024-01-30	2025-01-30
PyTorch	1.13.1	11.7	28.10.2022	28.10.2024
PyTorch	2.1.0	12,1	2023-10-04	2024-10-04
PyTorch	2.0.0	12,1	15.03.2023	2024-03-15
PyTorch	1.12.1	11.6	01.07.2022	01.07.2023
PyTorch	1.11.0	11.5	10.03.2022	2023-03-10
TensorFlow	2.16.0	12.3	2024-03-07	2025-03-07
TensorFlow	2.15.0	12.2	2023-11-14	2024-11-14
TensorFlow	2.13.0	11.8	2023-07-19	2024-07-19
TensorFlow	2.12.0	11.8	2023-03-23	2024-03-23
TensorFlow	2.11.0	11.2	18.11.2022	18.11.2023
TensorFlow	2.10.1	11.2	06.09.2022	2023-09-06
TensorFlow	2.9.3	11.2	17.05.2022	17.05.2023

Nicht unterstützte Betriebssystemversionen

Betriebssystem	Ende des Patches
Ubuntu 20.04	2025-05-31
Ubuntu 18.04	2023-05-31

Archiv der DLAMI-Versionshinweise nicht unterstützt

Base

GPU

- AWS Deep-Learning-Basis-AMI (Ubuntu 20.04)
- AWS Deep-Learning-Basis-AMI (Ubuntu 18.04)

Ein einziges Framework

PyTorch Spezifisches AMI

GPU

- AWS Deep-Learning-AMI-GPU PyTorch 2.3 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 2.3 (Amazon Linux 2)
- AWS ARM64 Deep-Learning-AMI-GPU PyTorch 2.3 (Ubuntu 22.04)
- AWS Deep-Learning-AMI-GPU PyTorch 2.2 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU PyTorch 2.2 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 2.1 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 2.0 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU PyTorch 2.0 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 1.13 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU PyTorch 1.13 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 1.12 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU PyTorch 1.12 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 1.11 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU PyTorch 1.11 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 1.10 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU PyTorch 1.10 (Ubuntu 20.04)
- AWS Deep Learning AMI Graviton GPU PyTorch 1.10 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 1.10 (Ubuntu 18.04)

- AWS Deep-Learning-AMI-GPU PyTorch 1.9 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU PyTorch 1.9 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU PyTorch 1.9 (Ubuntu 18.04)

TensorFlow Spezifisches AMI

GPU

- AWS Deep-Learning-AMI-GPU TensorFlow 2.16 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.16 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.15 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.15 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.13 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.13 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.12 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.12 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.11 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.11 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.10 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.10 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.9 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.9 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.8 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.8 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.7 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.7 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.6 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.6 (Ubuntu 20.04)
- AWS Deep Learning-AMI Graviton GPU TensorFlow 2.6 (Ubuntu 20.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.6 (Ubuntu 18.04)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.5 (Amazon Linux 2)
- AWS Deep-Learning-AMI-GPU TensorFlow 2.5 (Ubuntu 20.04)

Multifunktionales Framework

GPU

- AWS Deep-Learning-AMI (Ubuntu 18.04)

Wichtige NVIDIA-Treiberänderungen für DLAMIs

Am 15. November 2023 AWS wurden wichtige Änderungen an AWS Deep Learning AMIs (DLAMI) im Zusammenhang mit dem verwendeten NIVIDA-Treiber vorgenommen. DLAMIs Informationen darüber, was sich geändert hat und ob sich dies auf Ihre Verwendung von auswirkt, finden Sie unter. DLAMIs Änderung des DLAMI NVIDIA-Treibers FAQs

Änderung des DLAMI NVIDIA-Treibers FAQs

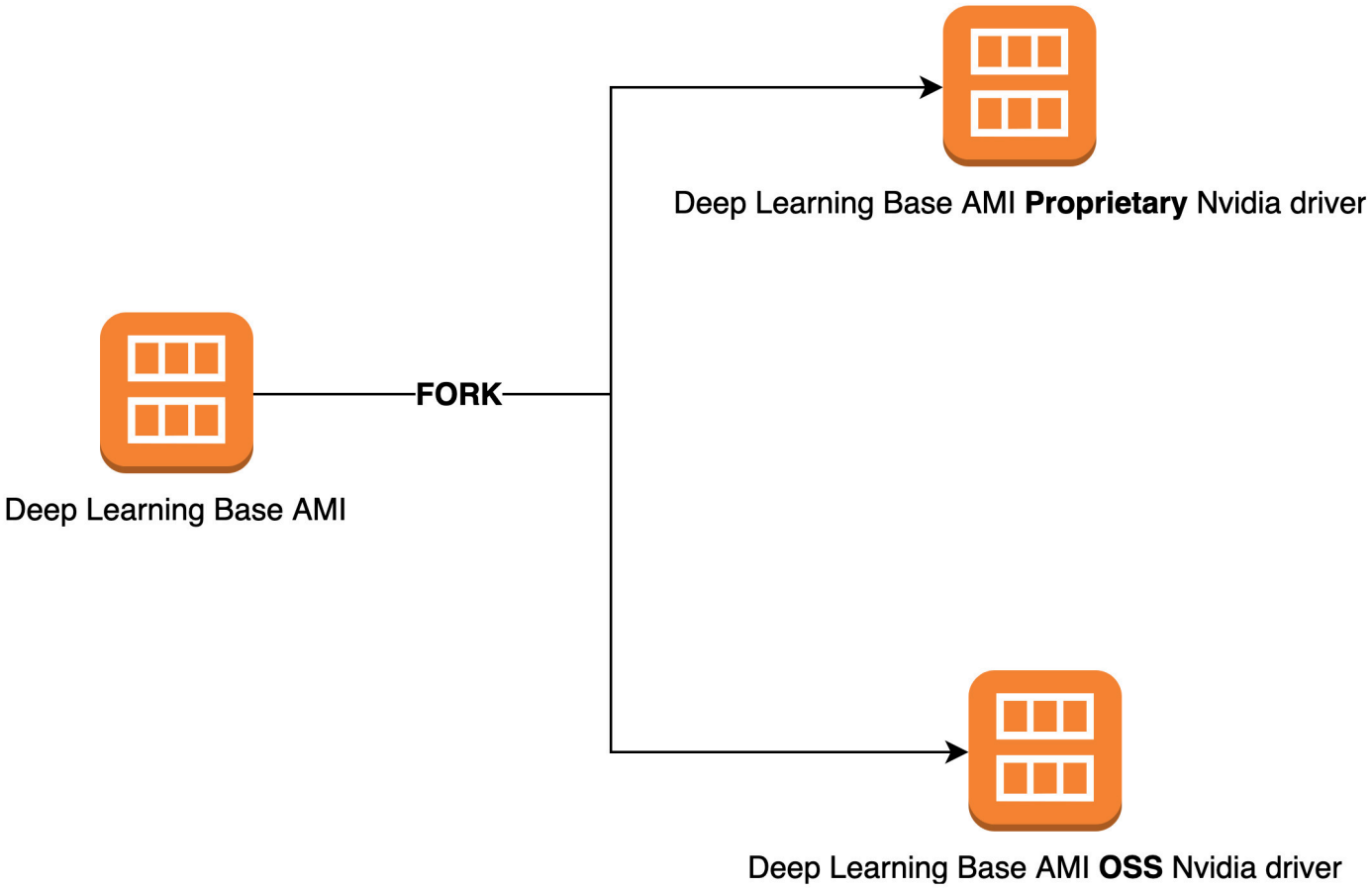
- Was hat sich geändert?
- Warum war diese Änderung erforderlich?
- DLAMIs Worauf hat sich diese Änderung ausgewirkt?
- Was bedeutet das für dich?
- Gibt es bei der neueren Version einen Verlust an Funktionalität? DLAMIs
- Hatte diese Änderung Auswirkungen auf Deep Learning Containers?

Was hat sich geändert?

Wir haben uns DLAMIs in zwei separate Gruppen aufgeteilt:

- DLAMIs die einen proprietären NVIDIA-Treiber verwenden (zur Unterstützung von P3, P3dn, G3)
- DLAMIs die den NVIDIA OSS-Treiber verwenden (zur Unterstützung von G4dn, G5, P4, P5)

Aus diesem Grund haben wir DLAMIs für jede der beiden Kategorien neue mit neuen Namen und neuem AMI erstellt IDs. Diese DLAMIs sind nicht austauschbar. Das heißt, DLAMIs von einer Gruppe werden keine Instances unterstützt, die von der anderen Gruppe unterstützt werden. Beispielsweise unterstützt das DLAMI, das P5 unterstützt, G3 nicht, und das DLAMI, das G3 unterstützt, unterstützt P5 nicht.



Warum war diese Änderung erforderlich?

Bisher war DLAMIs für NVIDIA ein proprietärer Kernel-Treiber von NVIDIA GPUs enthalten. Die Upstream-Linux-Kernel-Community akzeptierte jedoch eine Änderung, die proprietäre Kerneltreiber wie den NVIDIA-GPU-Treiber von der Kommunikation mit anderen Kerneltreibern isoliert. Durch diese Änderung wird GPUDirect RDMA auf Instances der Serien P4 und P5 deaktiviert. Dies ist der Mechanismus, der die effiziente Nutzung von EFA für GPUs verteilte Schulungen ermöglicht. Verwenden Sie daher DLAMIs jetzt den OpenRM-Treiber (NVIDIA-Open-Source-Treiber), der mit den Open-Source-EFA-Treibern verknüpft ist, um G4dn, G5, P4 und P5 zu unterstützen. Dieser OpenRM-Treiber unterstützt jedoch keine älteren Instanzen (wie P3 und G3). Um sicherzustellen, dass wir weiterhin aktuelle, performante und sichere Produkte anbieten, DLAMIs die beide Instanztypen unterstützen, haben wir uns DLAMIs in zwei Gruppen aufgeteilt: eine mit dem OpenRM-Treiber (der G4dn, G5, P4 und P5 unterstützt) und eine mit dem älteren proprietären Treiber (der P3, P3dn und G3 unterstützt).

DLAMIs Worauf hat sich diese Änderung ausgewirkt?

Diese Änderung betraf alle DLAMIs.

Was bedeutet das für dich?

Alle bieten DLAMIs weiterhin Funktionalität, Leistung und Sicherheit, solange Sie sie auf einem unterstützten Amazon Elastic Compute Cloud (Amazon EC2) Instance-Typ ausführen. Um zu ermitteln, welche EC2 Instance-Typen ein DLAMI unterstützt, lesen Sie die Versionshinweise für dieses DLAMI und suchen Sie dann nach Supported Instances. EC2 Eine Liste der derzeit unterstützten DLAMI-Optionen und Links zu ihren Versionshinweisen finden Sie unter. AMIs Versionshinweise zu Deep Learning

Darüber hinaus müssen Sie die richtigen Befehle AWS Command Line Interface (AWS CLI) verwenden, um die aktuelle Version aufzurufen. DLAMIs

Verwenden Sie für Base DLAMIs , die P3, P3dn und G3 unterstützen, diesen Befehl:

```
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
```

Verwenden Sie für Basen DLAMIs , die G4dn, G5, P4 und P5 unterstützen, diesen Befehl:

```
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
```

Gibt es bei der neueren Version einen Verlust an Funktionalität? DLAMIs

Nein, es gibt keinen Funktionsverlust. Die aktuellen DLAMIs Versionen bieten die gesamte Funktionalität, Leistung und Sicherheit der Vorgängerversionen DLAMIs, sofern Sie sie auf einem unterstützten EC2 Instance-Typ ausführen.

Hatte diese Änderung Auswirkungen auf Deep Learning Containers?

Nein, diese Änderung hatte keine Auswirkungen auf AWS Deep Learning Containers, da sie den NVIDIA-Treiber nicht enthalten. Stellen Sie jedoch sicher, dass Sie Deep Learning Containers auf Geräten ausführen AMIs , die mit den zugrunde liegenden Instances kompatibel sind.

Weitere Informationen über DLAMI

Weitere Ressourcen mit verwandten Informationen zu DLAMI finden Sie außerhalb des AWS Deep Learning AMIs Developer Guide. Schauen Sie AWS re:Post sich unter Fragen anderer Kunden zu DLAMI an oder stellen Sie Ihre eigenen Fragen. Lesen Sie auf dem AWS Machine Learning Blog und anderen AWS Blogs offizielle Beiträge über DLAMI.

AWS re:Post

Schlagwort: AWS Deep Learning AMIs

AWS Blog

- AWS Blog Machine Learning | Kategorie: AWS Deep Learning AMIs
- AWS Blog Machine Learning | Schnelleres Training mit optimiertem TensorFlow 1.6 auf Amazon EC2 C5- und P3-Instances
- AWS Blog Machine Learning | Neu AWS Deep Learning AMIs für Praktiker des Machine Learning
- AWS Partner Network (APN) Blog | Neue Schulungskurse verfügbar: Einführung in Machine Learning und Deep Learning auf AWS
- AWS Nachrichtenblog | Eine Reise in die Welt des Deep Learning mit AWS

Veraltete Funktionen von DLAMI

In der folgenden Tabelle sind die veralteten Funktionen von AWS Deep Learning AMIs (DLAMI), das Datum, an dem wir sie eingestellt haben, und Einzelheiten darüber aufgeführt, warum wir sie als veraltet eingestuft haben.

Funktion	Datum	Details
Ubuntu 16.04	10.07.2021	Ubuntu Linux 16.04 LTS hat am 30. April 2021 das Ende seines fünfjährigen LTS-Fensters erreicht und wird von seinem Anbieter nicht mehr unterstützt. Ab Oktober 2021 gibt es in neuen Versionen keine Updates mehr für das Deep Learning Base AMI (Ubuntu 16.04). Frühere Versionen werden weiterhin verfügbar sein.
Amazon Linux	10.07.2021	Amazon Linux ist end-of-lifeab Dezember 2020 verfügbar. Ab Oktober 2021 gibt es in neuen Versionen keine Updates mehr für das Deep Learning AMI (Amazon Linux). Frühere Versionen des Deep Learning AMI (Amazon Linux) werden weiterhin verfügbar sein.
Chainer	01.07.2020	Chainer hat das Ende der Hauptversionen ab Dezember 2019 angekündi

Funktion	Datum	Details
		gt. Aus diesem Grund werden wir ab Juli 2020 keine Chainer Conda-Umgebungen mehr in das DLAMI aufnehmen. Frühere Versionen des DLAMI, die diese Umgebunge n enthalten, werden weiterhin verfügbar sein. Wir werden Updates für diese Umgebungen nur bereitstellen, wenn von der Open-Source-Community Sicherheitskorrekturen für diese Frameworks veröffent licht werden.
Python 3.6	15.06.2020	Aufgrund von Kundenanf ragen wechseln wir für neue TF/MX/PT Versionen zu Python 3.7.

Funktion	Datum	Details
Python 2	01.01.2020	Die Python-Open-Source -Community hat die Unterstützung für Python 2 offiziell beendet. Die TensorFlow PyTorch ,- und MXNet Communitys haben außerdem angekündigt, dass die Versionen TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4 und MXNet 1.6.0 die letzten sein werden, die Python 2 unterstützen.

Dokumentenhistorie für DLAMI

Die folgende Tabelle enthält eine Historie der letzten DLAMI-Versionen und der damit verbundenen Änderungen am AWS Deep Learning AMIs Developer Guide.

Letzte Änderungen

Änderung	Beschreibung	Datum
Verwenden von TensorFlow Serving zum Trainieren eines MNIST-Modells	Ein Beispiel für die Verwendung von Tensorflow Serving zum Trainieren des MNIST-Modells.	14. Februar 2025
ARM64 DLAMI	Das unterstützt AWS Deep Learning AMIs jetzt Bilder auf Basis eines Arm64-Prozessors. GPUs	29. November 2021
TensorFlow 2	Das Deep Learning-AMI mit Conda enthält jetzt TensorFlow 2 mit CUDA 10.	3. Dezember 2019
AWS Inferentia	Das Deep Learning AMI unterstützt jetzt AWS Inferentia-Hardware und das AWS Neuron SDK.	3. Dezember 2019
Installation von einem Nightly PyTorch Build aus	Es wurde ein Tutorial hinzugefügt, das erklärt PyTorch, wie Sie mit Conda einen nächtlichen Build von PyTorch auf Ihrem Deep Learning-AMI deinstallieren und anschließend installieren können.	25. September 2018

Conda-Tutorial	Die Beispiel-MOTD wurde auf eine neuere Version aktualisiert.	23. Juli 2018

Frühere Änderungen

Die folgende Tabelle enthält eine Historie früherer DLAMI-Versionen und verwandter Änderungen vor Juli 2018.

Änderung	Beschreibung	Datum
TensorFlow mit Horovod	Es wurde ein Tutorial für das Training ImageNet mit TensorFlow und Horovod hinzugefügt.	6. Juni 2018
Aktualisierungsanleitung	Aktualisierungsanleitung wurde hinzugefügt.	15. Mai 2018
Neue Regionen und neues 10-Minuten-Tutorial	Neue Regionen hinzugefügt: US West (Nordkalifornien), Südamerika, Kanada (Zentral), EU (London) und EU (Paris). Außerdem die erste Version eines 10-Minuten-Tutorials mit dem Titel: „Erste Schritte mit dem Deep Learning AMI“.	26. April 2018
Chainer-Tutorial	Ein Tutorial für die Nutzung von Chainer in den Modi mit mehreren GPUs, einer GPU und CPU wurde hinzugefügt. Die CUDA-Integration wurde für mehrere Frameworks von CUDA 8 auf CUDA 9 aktualisiert.	28. Februar 2018

Änderung	Beschreibung	Datum
Linux AMIs v3.0 sowie Einführung von MXNet Model Server, Serving und TensorFlow TensorBoard	Es wurden Tutorials für Conda AMIs mit neuen Funktionen zur Bereitstellung von Modellen und Visualisierungen unter Verwendung von MXNet Model Server v0.1.5, TensorFlow Serving v1.4.0 und v0.4.0 hinzugefügt. TensorBoard AMI und Framework CUDA-Funktionen, wie in den Conda- und CUDA-Übersichten beschrieben. Neueste Versionshinweise nach https://aws.amazon.com/releasenotes/ verschoben	25. Januar 2018
Linux v2.0 AMIs	Base, Source und Conda AMIs wurden mit NCCL 2.1 aktualisiert. Source und Conda AMIs wurden mit MXNet v1.0, PyTorch 0.3.0 und Keras 2.0.9 aktualisiert.	11. Dezember 2017
Zwei Windows-AMI-Optionen hinzugefügt	Windows 2012 R2 und 2016 AMIs veröffentlicht: zur AMI-Auswahlhilfe hinzugefügt und zu den Versionshinweisen hinzugefügt.	30. November 2017
Erste Dokumentationsversion	Detaillierte Beschreibung der Änderung mit Link zum geänderten Thema/Abschnitt.	15. November 2017

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.