



Entwicklerhandbuch

Deep-Learning-AMI



Deep-Learning-AMI: Entwicklerhandbuch

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

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist der AWS Deep Learning AMI?	1
Informationen zu diesem Handbuch	1
Voraussetzungen	1
Beispielanwendungsfälle	1
Features	2
Vorinstallierte Frameworks	2
Vorinstallierte GPU-Software	3
Model-Bereitstellung und Visualisierung	3
Erste Schritte	4
Erste Schritte mit dem DLAMI	4
DLAMI-Auswahl	4
CUDA-Installationen und Framework-Bindungen	5
Basis	6
Conda	7
Architektur	8
BS	8
Auswahl der Instance	9
Preisgestaltung	10
Verfügbarkeit in Regionen	11
GPU	11
CPU	12
Inferentia	13
Trainium	13
Richtlinie zur Support von Frameworks	15
Unterstützte Frameworks	15
Häufig gestellte Fragen	15
Welche Framework-Versionen erhalten Sicherheitspatches?	16
Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden?	16
Welche Bilder erhalten neue SageMaker AWS /Funktionen?	16
Wie ist die aktuelle Version in der Tabelle Unterstützte Frameworks definiert?	17
Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Frameworks aufgeführt ist?	17
Unterstützen DLAMIs frühere Versionen von? TensorFlow	17

Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version?	17
Wie oft werden neue Images veröffentlicht?	18
Wird meine Instance an Ort und Stelle gepatcht, während mein Workload läuft?	18
Was passiert, wenn eine neue gepatchte oder aktualisierte Framework-Version verfügbar ist?	18
Werden Abhängigkeiten aktualisiert, ohne die Framework-Version zu ändern?	18
Wann endet der aktive Support für meine Framework-Version?	18
Werden Images mit Framework-Versionen, die nicht mehr aktiv verwaltet werden, gepatcht?	20
Wie verwende ich eine ältere Framework-Version?	20
Wie bleibe ich up-to-date bei Support-Änderungen an Frameworks und deren Versionen?	21
Benötige ich eine kommerzielle Lizenz, um das Anaconda Repository nutzen zu können?	21
Ein DLAMI starten	22
Schritt 1: Starten Sie ein DLAMI	23
Rufen Sie die DLAMI-ID ab	23
Von der Amazon EC2 EC2-Konsole aus starten	24
Schritt 2: Connect zum DLAMI her	25
Schritt 3: Testen Sie Ihr DLAMI	26
Schritt 4: Ihre DLAMI-Instanz verwalten	26
Bereinigen	27
Jupyter-Einrichtung	27
Jupyter sichern	28
Starten des Servers	29
Konfigurieren des Clients	29
Am Jupyter-Notebook-Server anmelden	31
Verwendung eines DLAMI	34
Conda DLAMI	34
Einführung in das Deep Learning AMI mit Conda	34
Loggen Sie sich in Ihr DLAMI ein	35
Starten Sie die Umgebung TensorFlow	35
Wechseln Sie zur PyTorch Python-3-Umgebung	36
Entfernen von Umgebungen	37
Basis DLAMI	37
Verwenden des Deep Learning Base AMI	37
CUDA-Versionen konfigurieren	38
Jupyter Notebooks	38

Navigation der installierten Tutorials	39
Wechseln von Umgebungen mit Jupyter	40
Tutorials	40
10-Minuten-Tutorials	41
Aktivieren von Frameworks	41
Elastic Fabric Adapter	44
GPU-Überwachung und -Optimierung	59
AWS Inferentia	70
ARM64 DLAMI	92
Inferenz	95
Modellbereitstellung	96
Aktualisieren Sie Ihr DLAMI	102
DLAMI-Upgrade	102
Software-Updates	103
Benachrichtigungen veröffentlichen	104
Sicherheit	105
Datenschutz	106
Identitäts- und Zugriffsverwaltung	107
Authentifizierung mit Identitäten	107
Verwalten des Zugriffs mit Richtlinien	111
IAM mit Amazon EMR	113
Protokollieren und Überwachen	114
Nachverfolgung der Nutzung	114
Compliance-Validierung	114
Ausfallsicherheit	115
Sicherheit der Infrastruktur	115
Wichtige Änderungen an DLAMI	116
Häufig gestellte Fragen	116
Was ändert sich?	116
Warum ist diese Änderung erforderlich?	117
Welche DLAMIs sind von dieser Änderung betroffen?	118
Was bedeutet das für dich?	118
Wann sollten Sie mit der Verwendung der neuen DLAMIs beginnen?	119
Wird es bei den neuen DLAMIs zu Funktionseinbußen kommen?	119
Was ist mit DLCs?	119
Verwandte Informationen	120

Foren	120
Blogs	120
Häufig gestellte Fragen	120
Versionshinweise für DLAMI	124
.....	124
Basis DLAMI	124
DLAMI mit einem einzigen Framework	125
DLAMI mit mehreren Frameworks	126
DLAMI bis zur Veraltung bis zur Veraltung bis	127
Dokumentverlauf	130
AWS-Glossar	134
.....	CXXXV

Was ist der AWS Deep Learning AMI?

Willkommen zum Benutzerhandbuch für das AWS Deep Learning AMI.

Das AWS Deep Learning AMI (DLAMI) ist Ihre zentrale Anlaufstelle für Deep Learning in der Cloud. Dieses benutzerdefinierte Maschinen-Image ist in den meisten Amazon EC2 EC2-Regionen für eine Vielzahl von Instance-Typen verfügbar, von einer kleinen reinen CPU-Instance bis hin zu den neuesten leistungsstarken Multi-GPU-Instances. Es ist mit [NVIDIA CUDA](#) und [NVIDIA cuDNN](#) sowie den neuesten Versionen der beliebtesten Deep-Learning-Frameworks vorkonfiguriert.

Informationen zu diesem Handbuch

Diese Anleitung hilft Ihnen beim Starten und Verwenden des DLAMI. Er deckt mehrere häufige Anwendungsfälle für Deep-Learning ab – sowohl für Training als auch für Inferenz. Auch die Auswahl des richtigen AMIs für Ihren Zweck und des Instance-Typs werden besprochen. Das DLAMI enthält mehrere Tutorials für jedes der Frameworks. Es enthält auch Tutorials zu verteiltem Training, Debugging, Verwendung von AWS Inferentia und AWS Trainium sowie zu anderen Schlüsselkonzepten. Es enthält Anweisungen zur Konfiguration von Jupyter zum Ausführen der Tutorials in Ihrem Browser.

Voraussetzungen

Sie sollten mit Befehlszeilentools und grundlegendem Python vertraut sein, um das DLAMI erfolgreich ausführen zu können. Tutorials zur Verwendung der einzelnen Frameworks werden von den Frameworks selbst zur Verfügung gestellt. Dieser Leitfaden zeigt Ihnen jedoch, wie Sie die einzelnen Frameworks aktivieren und die entsprechenden Tutorials für den Einstieg finden.

Beispiele für DLAMI-Anwendungen

Lernen über Deep Learning: Das DLAMI ist eine hervorragende Wahl für das Lernen oder Unterrichten von Frameworks für maschinelles Lernen und Deep Learning. Es vereinfacht die Fehlersuche bei den Installationen der einzelnen Frameworks und sorgt für einen reibungslosen Betrieb auf einem einzelnen Computer. Das DLAMI wird mit einem Jupyter-Notebook geliefert und erleichtert die Ausführung der von den Frameworks bereitgestellten Tutorials für Personen, die mit maschinellem Lernen und Deep Learning noch nicht vertraut sind.

App-Entwicklung: Wenn Sie ein App-Entwickler sind und daran interessiert sind, Deep Learning zu nutzen, damit Ihre Apps die neuesten Fortschritte in der KI nutzen, ist das DLAMI das perfekte Testfeld 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 Datenanalytik: Wenn Sie ein Datenwissenschaftler sind oder daran interessiert sind, Ihre Daten mit Hilfe von Deep-Learning zu verarbeiten, stellen Ihnen viele der Frameworks eine Unterstützung für R und Spark bereit. Sie umfassen Tutorials von einfachen Regressionen bis hin zum Aufbau skalierbarer Datenverarbeitungssysteme für Personalisierungs- und Prognosesysteme.

Recherche: Wenn Sie ein Forscher sind und ein neues Framework ausprobieren, ein neues Modell testen oder neue Modelle trainieren möchten, können das DLAMI und die AWS Skalierungsfunktionen 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 [Verwandte Informationen](#).

Eigenschaften des DLAMI

Vorinstallierte Frameworks

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

- [Deep-Learning-AMI mit Conda](#) – Frameworks, die separat mit conda-Paketen und separaten Python-Umgebungen installiert werden.
- [Deep Learning Learning Learning Learning Learning Learning Learning](#) – Es sind 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.

Dies ist die vollständige Liste der unterstützten Frameworks von Deep Learning AMI with Conda:

- PyTorch
- TensorFlow 2

Note

Wir unterstützen MXNet, CNTK, Caffe, Caffe2, Theano, Chainer oder Keras in der nicht mehr. AWS Deep Learning AMI

Vorinstallierte GPU-Software

[Selbst wenn Sie eine reine CPU-Instanz verwenden, verfügt das DLAMI über NVIDIA CUDA und NVIDIA cuDNN.](#) Die installierte Software ist unabhängig vom Instance-Typ identisch. Beachten Sie, dass GPU-spezifische Tools nur in einer Instance funktionieren, die mindestens eine GPU umfasst. Weitere Informationen hierzu finden Sie in [Auswahl des Instanztyps für DLAMI.](#)

Weitere Informationen zur CUDA-Installation finden Sie im [CUDA-Installationen und Framework-Bindungen](#)

Model-Bereitstellung und Visualisierung

Deep Learning AMI mit Conda ist mit Modellservern sowohl für Modellvisualisierungen als auch für TensorFlow TensorBoard Modellvisualisierungen vorinstalliert.

- [TensorFlow Servieren](#)

Erste Schritte

Erste Schritte mit dem 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 [Verwandte Informationen](#) beschreibt benutzerdefinierte Setups, die von Interesse sein könnten.

Wenn Sie mit der Verwendung AWS oder Nutzung von Amazon EC2 noch nicht vertraut sind, beginnen Sie mit dem [Deep-Learning-AMI mit Conda](#). Wenn Sie mit Amazon EC2 und anderen AWS Services wie Amazon EMR, Amazon EFS oder Amazon S3 vertraut sind und daran interessiert sind, diese Services für Projekte zu integrieren, für die verteilte Schulungen oder Inferenzen erforderlich sind, sollten Sie herausfinden [Verwandte Informationen](#), ob einer für Ihren Anwendungsfall geeignet ist.

Wir empfehlen Ihnen, sich zuerst über [Wählen Sie Ihr DLAMI](#) zu informieren und so den am besten geeigneten Instance-Typ zu bestimmen.

Eine weitere Option ist dieses kurze Tutorial: [Launch a AWS Deep Learning AMI \(in 10 Minuten\)](#).

Nächster Schritt

[Wählen Sie Ihr DLAMI](#)

Wählen Sie Ihr DLAMI

Wir bieten eine Reihe von DLAMI-Optionen an. 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: CUDA versus Base versus Single-Framework versus Multi-Framework (Conda DLAMI)
- Rechenarchitektur: AWS [x86-basiert](#) versus ARM64-basiertes Graviton
- [Prozessortyp: GPU versus CPU versus Inferentia](#)
- SDK: CUDA gegen [Neuron AWS](#)
- Betriebssystem: Amazon Linux im Vergleich zu 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 Learning Learning Learning Learning Learning Learning Learning](#)
- [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 Learning Learning Learning Learning Learning Learning Learning](#) hat alle verfügbaren CUDA-Versionsserien

Die [Deep-Learning-AMI mit Conda](#) hat alle verfügbaren CUDA-Versionsserien

Note

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

Spezifische Framework-Versionennummern finden Sie in der [Versionshinweise für DLAMI](#)

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

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

Nächstes Thema

[Deep Learning Learning Learning Learning Learning Learning Learning Learning](#)

Verwandte Themen

- Anweisungen zum Wechseln zwischen CUDA-Versionen finden Sie im [Verwenden des Deep Learning Base AMI](#)-Tutorial.

Deep Learning Learning Learning Learning Learning Learning Learning Learning Learning

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 Ihnen die Wahl zwischen CUDA-Versionen.

Warum sollten Sie sich für den Basis-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 mit der Option Next Up mehr über die verschiedenen DLAMIs.

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 mit mehreren Frameworks oder mit einem einzigen Framework als DLAMIs verfügbar. 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 AMI](#)“ bezeichnet. Diese DLAMIs unterstützen die Betriebssysteme Ubuntu20.04 und Amazon Linux 2. Die Unterstützung von Betriebssystemen hängt von der Unterstützung durch das Upstream-Betriebssystem ab.

Stable versus Release-Kandidaten

Die Conda AMIs verwenden optimierte Binärdateien der neuesten formellen Versionen jedes Frameworks. Release-Kandidaten und experimentelle Funktionen sind nicht zu erwarten. Die Optimierungen hängen davon ab, ob das Framework Beschleunigungstechnologien wie Intels MKL DNN unterstützt, wodurch Training und Inferenz für C5- und C4-CPU-Instance-Typen beschleunigt werden. Die Binärdateien wurden außerdem so kompiliert, dass sie erweiterte Intel-Befehlsätze unterstützen, einschließlich, aber nicht beschränkt auf AVX, AVX-2, SSE4.1 und SSE4.2. 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 Amazon EC2 EC2-Instance. Weitere Informationen finden Sie unter [Verwenden des Deep Learning-AMI mit Conda](#).

Wenn Sie die Installation aus dem Quellcode mit benutzerdefinierten oder optimierten Build-Optionen durchführen möchten, [Deep Learning Learning Learning Learning Learning Learning Learning](#) 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 AMIs [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 des Instanztyps für DLAMI](#)

Nächstes Thema

[DLAMI-Betriebssystemoptionen](#)

DLAMI-Betriebssystemoptionen

DLAMIs werden in den folgenden Betriebssystemen angeboten.

- Amazon Linux 2
- Ubuntu 20.04
- Ubuntu 22.04

Ältere Versionen von Betriebssystemen sind auf veralteten 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 des Instanztyps für DLAMI](#)

Auswahl des Instanztyps für DLAMI

Generell sollten Sie bei der Auswahl eines Instanztyps für ein DLAMI Folgendes berücksichtigen.

- 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 eine [Amazon Elastic Inference an Ihre Amazon EC2 EC2-Instance](#) anhängen. 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 GPUs verteilen. 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 .](#)

Die folgenden Themen enthalten Informationen zu Überlegungen zu Instance-Typen.

Important

Die Deep Learning AMIs beinhalten Treiber, Software oder Toolkits, die von der NVIDIA Corporation entwickelt wurden, ihr Eigentum sind oder von ihr bereitgestellt werden. Sie erklären, diese NVIDIA-Treiber, -Software oder -Toolkits nur auf Amazon EC2-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 Amazon EC2 EC2-Instance-Hardware bezahlen.

Einige Amazon EC2 EC2-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 GPUs benötigen, benötigen Sie eine Instanz, die nicht zur Instanzklasse des kostenlosen Tiers gehört.

Weitere Informationen zur Instance-Auswahl und Preisgestaltung finden Sie unter [Amazon EC2 EC2-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, DLAMIs in die Region Ihrer Wahl 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 EC2 Regions](#).

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 über mehrere GPU-Instances verfügen oder wenn Sie eine Schulung für viele Instances mit GPUs verwenden.

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

Bei der Auswahl einer Instance sollte die Größe Ihres Modells berücksichtigt werden. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instance überschreitet, wählen Sie für Ihre Anwendung einen anderen Instance-Typ mit ausreichend Speicher.

- [Amazon EC2 P3-Instances](#) verfügen über bis zu 8 NVIDIA Tesla V100-GPUs.

- [Amazon EC2 P4-Instances](#) verfügen über bis zu 8 NVIDIA Tesla A100-GPUs.
- [Amazon EC2 P5-Instances](#) verfügen über bis zu 8 NVIDIA Tesla H100-GPUs.
- [Amazon EC2 G3-Instances](#) verfügen über bis zu 4 NVIDIA Tesla M60-GPUs.
- [Amazon EC2 G4-Instances haben bis zu 4](#) NVIDIA T4-GPUs.
- [Amazon EC2 G5-Instances](#) verfügen über bis zu 8 NVIDIA A10G-GPUs.
- [Amazon EC2 G6-Instances](#) haben bis zu 8 NVIDIA L4-GPUs.
- [Amazon EC2 G5g-Instances verfügen über AWS ARM64-basierte 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 unter [Das 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 und wählen Sie Compute [Optimized aus](#).

Note

Bei der Auswahl einer Instance sollte die Größe Ihres Modells berücksichtigt werden. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instance überschreitet, wählen Sie für Ihre Anwendung einen anderen Instance-Typ mit ausreichend Speicher.

- [Amazon EC2 C5-Instances](#) haben bis zu 72 Intel vCPUs. 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

Bei der Auswahl einer Instance sollte die Größe Ihres Modells berücksichtigt werden. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instance überschreitet, wählen Sie für Ihre Anwendung einen anderen Instance-Typ mit ausreichend Speicher.

- [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 DLAMIs finden Sie unter. [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

Bei der Auswahl einer Instance sollte die Größe Ihres Modells berücksichtigt werden. Wenn Ihr Modell den verfügbaren Arbeitsspeicher einer Instance überschreitet, wählen Sie für Ihre Anwendung einen anderen Instance-Typ mit ausreichend Speicher.

- [Amazon EC2 Trn1-Instances](#) verfügen über bis zu 16 AWS Trainium-Chips und einen Netzwerkdurchsatz von 100 Gbit/s.

Richtlinie zur Support von Frameworks

[AWS Deep Learning AMIs](#) (DLAMIs) vereinfachen die Image-Konfiguration für Deep-Learning-Workloads und sind für die neuesten Frameworks, Hardware, Treiber, Bibliotheken und Betriebssysteme optimiert. Auf dieser Seite werden die Richtlinien zur Framework-Supportunterstützung für DLAMIs beschrieben. Eine Liste der verfügbaren DLAMIs finden Sie in den [Versionshinweisen für DLAMI](#).

Unterstützte Frameworks

In der folgenden [Tabelle mit den AWS Deep Learning AMI Framework-Supportrichtlinien](#) können Sie überprüfen, welche Frameworks und Versionen aktiv unterstützt werden.

Unter Ende des Patches erfahren Sie, wie lange aktuelle Versionen AWS unterstützt werden, die aktiv vom Wartungsteam des Origin-Frameworks unterstützt werden. Frameworks und Versionen sind als DLAMIs mit einem Framework oder als DLAMIs mit mehreren Frameworks verfügbar.

Note

In der Framework-Version x.y.z bezieht sich x auf die Hauptversion, y auf die Nebenversion und z auf die Patch-Version. Für TensorFlow 2.6.5 ist die Hauptversion beispielsweise 2, die Nebenversion ist 6 und die Patch-Version ist 5.

Weitere Informationen zu bestimmten Bildern finden Sie in den Versionshinweisen:

- Versionshinweise zu [DLAMI für ein einzelnes Framework](#)
- Versionshinweise zu [DLAMI für mehrere Frameworks](#)

Häufig gestellte Fragen

- [Welche Framework-Versionen erhalten Sicherheitspatches?](#)
- [Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden?](#)
- [Welche Bilder erhalten neue SageMaker AWS /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 Frameworks aufgeführt ist?](#)
- [Unterstützen DLAMIs frühere Versionen von? TensorFlow](#)
- [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 die Framework-Version in der [Tabelle der AWS Deep Learning AMI Framework-Support-Richtlinien](#) als Unterstützt gekennzeichnet ist, erhält sie Sicherheitspatches.

Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden?

Wir veröffentlichen neue DLAMIs kurz nach der Veröffentlichung neuer Versionen von TensorFlow und PyTorch. 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 /Funktionen?

Neue Funktionen werden normalerweise in der neuesten Version von DLAMIs für PyTorch und veröffentlicht. TensorFlow Einzelheiten zu neuen SageMaker Funktionen finden Sie in den Versionshinweisen für ein bestimmtes Bild. AWS Eine Liste der verfügbaren DLAMIs 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 AWS Deep Learning AMI Framework Support Policy](#) bezieht sich auf die neueste Framework-Version, die AWS 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 Frameworks aufgeführt ist?

Wenn Sie eine Version ausführen, die nicht in der [Tabelle der AWS Deep Learning AMI Framework-Supportrichtlinien 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 zu aktualisieren, das das neueste DLAMI Ihrer Wahl verwendet. Eine Liste der verfügbaren DLAMIs finden Sie in den [Versionshinweisen für DLAMI](#).

Unterstützen DLAMIs frühere Versionen von? TensorFlow

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 AMI Framework-Supportrichtlinien](#) angegeben. Weitere Informationen finden Sie unter [Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Frameworks aufgeführt ist?](#)

Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version?

[Um ein DLAMI mit der neuesten Framework-Version zu verwenden, rufen Sie die DLAMI-ID ab und verwenden Sie sie, um das DLAMI über die EC2-Konsole zu starten.](#) Beispiele für AWS CLI-Befehle zum Abrufen der AWS Deep Learning AMI ID finden Sie im Abschnitt Deep Learning Frameworks im [AWS Deep Learning AMIKatalog](#). AWS CLI-AMI-ID-Abfragen sind auch in den [DLAMI-Versionshinweisen für ein einzelnes Framework](#) enthalten. Die von Ihnen gewählte Framework-Version muss in der [Tabelle AWS Deep Learning AMI Framework-Support-Richtlinien als Unterstützt](#) gekennzeichnet 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-Instance einschalten, Ihre Workloads und Skripts migrieren und dann Ihre vorherige Instanz ausschalten.

Was passiert, wenn eine neue gepatchte oder aktualisierte Framework-Version verfügbar ist?

Suchen Sie regelmäßig auf der Seite mit den Versionshinweisen nach Ihrem Image. Wir empfehlen Ihnen, auf neue gepatchte oder aktualisierte Frameworks zu aktualisieren, sobald diese verfügbar sind. Eine Liste der verfügbaren DLAMIs finden Sie in den [Versionshinweisen für DLAMI](#).

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 Informationen zu Abhängigkeiten 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 \(Patch\)](#)

- [AWSSicherheitspatches](#)
- [Ende des Patch-Datums \(Alterung nicht mehr verfügbar\)](#)
- [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 Versionshinweisen für jeden kleineren Patch.

Neue DLAMIs, die aufgrund von Patch-Upgrades erstellt wurden, werden mit einer neuen [AMI-ID](#) gekennzeichnet.

AWSSicherheitspatches

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 DLAMIs, die aufgrund von Patch-Upgrades erstellt wurden, werden mit einer neuen [AMI-ID](#) gekennzeichnet.

Ende des Patch-Datums (Alterung nicht mehr verfügbar)

DLAMIs haben das Ende des Patches 365 Tage nach dem GitHub Veröffentlichungsdatum erreicht.

Für [Multi-Framework-DLAMIs](#) ist bei der Aktualisierung einer der Framework-Versionen 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 neuen Versionen 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 EC2-Konsole zu starten.](#) AWSCLI-Befehle zum Abrufen der AMI-ID finden Sie im Abschnitt Deep Learning Frameworks im [AWSDeep Learning AMI-Katalog](#). AWS CLI-AMI-ID-Abfragen sind auch in den [DLAMI-Versionshinweisen für ein einzelnes Framework](#) enthalten.

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 AMI Framework Support Policies](#) 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 gepflegte DLAMIs wurden vom Anaconda-Kanal auf die öffentlich verfügbare Open-Source-Version von Conda \(Conda-Forge\) migriert.](#)

Ein DLAMI starten und konfigurieren

Wenn Sie hier sind, sollten Sie bereits wissen, welches AMI Sie starten möchten. Wenn nicht, suchen Sie nach einem DLAMI und der zugehörigen Hardware, Frameworks und ID-Abruf in der [Versionshinweise für DLAMI](#)

Sie sollten auch wissen, welchen Instance-Typ und welche Region Sie wählen werden. Wenn dies nicht der Fall ist, durchsuchen Sie [Auswahl des Instanztyps für DLAMI](#).

Note

In den Beispielen verwenden wir p3.16xlarge als Standard-Instanztyp. Ersetzen Sie dies einfach durch den von Ihnen gewünschten Instance-Typ.

Important

Wenn Sie Elastic Inference verwenden möchten, müssen Sie die [Einrichtung von Elastic Inference abschließen](#), bevor Sie Ihr DLAMI starten können.

Themen

- [Schritt 1: Starten Sie ein DLAMI](#)
- [Schritt 2: Connect zum DLAMI her](#)
- [Schritt 3: Testen Sie Ihr DLAMI](#)
- [Schritt 4: Ihre DLAMI-Instanz verwalten](#)
- [Bereinigen](#)
- [Einrichten eines Jupyter-Notebook-Servers](#)

Schritt 1: Starten Sie ein DLAMI

Note

Für diese exemplarische Vorgehensweise verweisen wir möglicherweise speziell auf das Deep Learning AMI (Ubuntu 18.04). Auch wenn Sie ein anderes DLAMI auswählen, sollten Sie in der Lage sein, dieser Anleitung zu folgen.

1. [Finden Sie die ID Ihres DLAMI](#)
2. [Starten Sie eine Amazon EC2 EC2-Instance von Ihrem DLAMI](#)

Sie werden die Amazon EC2 EC2-Konsole verwenden. Folgen Sie den Anweisungen unter [Von der Amazon EC2 EC2-Konsole aus starten](#)

Tip

CLI-Option: Wenn Sie ein DLAMI mithilfe der AWS CLI einrichten möchten, benötigen Sie die AMI-ID, die Region und den Instance-Typ sowie Ihre Sicherheitstoken-Informationen. Stellen Sie sicher, dass Ihre AMI- und Instance-IDs bereit sind. Wenn Sie die AWS CLI noch nicht eingerichtet haben, tun Sie dies zunächst anhand der Anleitung zur [Installation der AWS Befehlszeilenschnittstelle](#).

3. Nachdem Sie die Schritte von einer dieser Optionen ausgeführt haben, warten Sie, bis die Instance bereit ist. Dieser Vorgang dauert einige Minuten. Sie können den Status der Instance in der [EC2-Konsole](#) überprüfen.

Rufen Sie die DLAMI-ID ab

Jedes AMI besitzt eine eindeutige Kennung (ID). Sie können die ID für das DLAMI Ihrer Wahl mit der AWS Befehlszeilenschnittstelle (AWS CLI) abfragen. Falls Sie das noch nicht AWS CLI installiert haben, finden Sie weitere Informationen unter [Erste Schritte mit der AWS CLI](#).

Note

Zur Erinnerung: [Sie finden alle DLAMIs und die zugehörigen Prozessoren/Beschleuniger, das Betriebssystem, die Rechenarchitektur, die empfohlenen Amazon EC2 EC2-Instance-](#)

[Familien, den Supportstatus und Anfragen zum ID-Abruf im Katalog.AWS Deep Learning AMI](#) Weitere Informationen (Treiber, Python-Versionen, Amazon [Versionshinweise für DLAMI](#) EBS-Typ) finden Sie auch in den DLAMI-Versionshinweisen unter.

1. Stellen Sie sicher, dass Ihre AWS Anmeldeinformationen konfiguriert sind.

```
aws configure
```

2. Verwenden Sie den folgenden Befehl, um die ID Ihres DLAMI abzurufen oder die im Katalog bereitgestellte Abfrage zu suchen. AWS Deep Learning AMI

```
aws ec2 describe-images --region us-east-1 --owners amazon \  
--filters 'Name=name,Values=Deep Learning AMI (Ubuntu 18.04) Version ??.' \  
'Name=state,Values=available' \  
--query 'reverse(sort_by(Images, &CreationDate))[:1].ImageId' --output text
```

Note

Sie können eine Release-Version für ein bestimmtes Framework angeben oder die neueste Version abrufen, indem Sie die Versionsnummer durch ein Fragezeichen ersetzen.

3. Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
ami-094c089c38ed069f2
```

Kopieren Sie diese DLAMI-ID und drücken Sie, um die q Eingabeaufforderung zu beenden.

Nächster Schritt

[Von der Amazon EC2 EC2-Konsole aus starten](#)

Von der Amazon EC2 EC2-Konsole aus starten

Note

Gehen Sie wie [folgt](#) vor, um eine Instance mit Elastic Fabric Adapter (EFA) zu starten.

1. Öffnen Sie die [EC2-Konsole](#).
2. Notieren Sie Ihre aktuelle Region in der obersten Navigation. Wenn Sie dies nicht wünschen AWS-Region, ändern Sie diese Option, bevor Sie fortfahren. Weitere Informationen finden Sie unter .
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 unter Meine AMIs nach einem vorhandenen DLAMI oder wählen Sie Quick Start.
 - 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 im AWS Deep Learning AMI Katalog. Allgemeine Empfehlungen zu DLAMI-Instanztypen finden Sie unter [Instanzauswahl](#).

Note

Wenn Sie [Elastic Inference](#) (EI) verwenden möchten, klicken Sie auf Configure Instance Details, wählen Sie Add an Amazon EI Accelerator aus und wählen Sie dann die Größe des Amazon EI Accelerators aus.

6. Wählen Sie Launch Instance aus.

Tip

Eine exemplarische Vorgehensweise [mit Screenshots finden Sie unter Erste Schritte mit AWS Deep Learning Using the Deep Learning AMI](#).

Nächster Schritt

[Schritt 2: Connect zum DLAMI her](#)

Schritt 2: Connect zum DLAMI her

Stellen Sie eine Connect zu dem DLAMI her, das Sie von einem Client (Windows, macOS oder Linux) aus gestartet haben. Weitere Informationen finden Sie unter [Connect to Your Linux Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Halten Sie eine Kopie des SSH-Login-Befehls griffbereit, wenn Sie das Jupyter-Setup nach dem Einloggen durchführen möchten. Sie werden eine Variante davon verwenden, um sich mit der Jupyter-Webseite zu verbinden.

Nächster Schritt

[Schritt 3: Testen Sie Ihr DLAMI](#)

Schritt 3: Testen Sie Ihr DLAMI

Abhängig von Ihrer DLAMI-Version haben Sie unterschiedliche Testmöglichkeiten:

- [Deep-Learning-AMI mit Conda](#)— gehe zu. [Verwenden des Deep Learning-AMI mit Conda](#)
- [Deep Learning Learning Learning Learning Learning Learning Learning](#)— Schlagen Sie in der Installationsdokumentation Ihres gewünschten Frameworks nach.

Sie können auch ein Jupyter-Notebook erstellen, Tutorials ausprobieren oder in Python programmieren. Weitere Informationen finden Sie unter [Einrichten eines Jupyter-Notebook-Servers](#).

Schritt 4: Ihre DLAMI-Instanz verwalten

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

Bereinigen

Wenn Sie das DLAMI nicht mehr benötigen, können Sie es beenden oder kündigen, um weitere Gebühren zu vermeiden. Durch das Stoppen einer Instance wird diese beibehalten, sodass Sie sie später weiter verwenden können. Ihre Konfigurationen, Dateien und andere nichtflüchtige Informationen werden in einem Volume auf Amazon S3 gespeichert. Sie werden mit der kleinen S3-Gebühr belastet, um das Volumen beizubehalten, während die Instance gestoppt ist, aber Sie werden nicht mehr für die Rechenressourcen belastet, während sie sich im gestoppten Zustand befindet. Wenn Sie die Instance erneut starten, wird das Volume geladen und Ihre Daten werden dort gespeichert. Wenn Sie eine Instance beenden, ist sie weg und Sie können sie nicht erneut starten. Ihre Daten befinden sich tatsächlich immer noch in S3. Um weitere Gebühren zu vermeiden, müssen Sie das Volume ebenfalls löschen. Weitere Anweisungen finden Sie unter [Terminate Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Einrichten eines Jupyter-Notebook-Servers

Mit einem Jupyter-Notebookserver können Sie Jupyter-Notebooks von Ihrer DLAMI-Instanz aus erstellen und ausführen. Mit Jupyter-Notebooks können Sie Experimente zum maschinellen Lernen (ML) für Training und Inferenz durchführen, während Sie die AWS Infrastruktur nutzen und auf im DLAMI integrierte Pakete zugreifen. Weitere Informationen zu Jupyter-Notebooks finden Sie in [der Jupyter Notebook-Dokumentation](#).

Um einen Jupyter-Notebook-Server einzurichten, müssen Sie:

- Konfigurieren Sie den Jupyter-Notebook-Server auf Ihrer Amazon EC2 EC2-DLAMI-Instance.
- Konfigurieren Sie Ihren Client so, dass Sie sich mit dem Jupyter-Notebook-Server verbinden können. Wir stellen Konfigurationsanweisungen für Windows-, MacOS- und Linux-Clients zur Verfügung.
- Testen Sie die Einrichtung, indem Sie sich am Jupyter-Notebook-Server anmelden.

Folgen Sie den Anweisungen in den folgenden Themen, um die Schritte zum Einrichten eines Jupyter abzuschließen. Nachdem Sie einen Jupyter-Notebookserver eingerichtet haben, finden Sie unter Informationen [Ausführen von Jupyter-Notebook-Tutorials](#) zum Ausführen der Beispiel-Notebooks, die im DLAMI geliefert werden.

Themen

- [Sichern Ihres Jupyter-Servers](#)

- [Starten des Jupyter-Notebook-Servers](#)
- [Konfigurieren des Clients für die Verbindung mit dem Jupyter-Server](#)
- [Testen des Jupyter-Notebook-Servers mit einer Anmeldung](#)

Sichern Ihres Jupyter-Servers

Hier richten wir Jupyter mit SSL und einem benutzerdefinierten Passwort ein.

Connect der Amazon EC2 EC2-Instance her, und führen Sie dann das folgende Verfahren aus.

Konfigurieren des Jupyter-Servers

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

Sie können ein reguläres SSL-Zertifikat erstellen, das Drittanbieter-signiert ist und nicht dazu führt, dass der Browser eine Sicherheitswarnung ausgibt. Dieser Prozess ist wesentlich aufwendiger. Besuchen Sie die [Jupyter-Dokumentation](#) für weitere Informationen.

Nächster Schritt

[Starten des Jupyter-Notebook-Servers](#)

Starten des Jupyter-Notebook-Servers

Nun können Sie den Jupyter-Server starten, indem Sie sich bei der Instance anmelden und den folgenden Befehl ausführen, der das SSL-Zertifikat verwendet, das Sie im vorherigen Schritt erstellt haben.

```
$ 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 Localhost-URL auf den Server zugreifen können. Dies funktioniert erst nach Erstellung des Tunnel.

Note

Jupyter übernimmt das Wechseln von Umgebungen, wenn Sie die Frameworks mithilfe der Jupyter-Web-Schnittstelle wechseln. Weitere Informationen zu diesem Thema finden Sie im [Wechseln von Umgebungen mit Jupyter](#).

Nächster Schritt

[Konfigurieren des Clients für die Verbindung mit dem Jupyter-Server](#)

Konfigurieren des Clients für die Verbindung mit dem Jupyter-Server

Nachdem Sie Ihren Client so konfiguriert haben, dass er eine Verbindung zum Jupyter-Notebook-Server herstellt, können Sie Notebooks auf dem Server in Ihrem Arbeitsbereich erstellen und darauf zugreifen und Ihren Deep-Learning-Code auf dem Server ausführen.

Informationen zur Konfiguration finden Sie unter den folgenden Links.

Themen

- [Konfigurieren eines Windows-Clients](#)
- [Konfigurieren eines Linux- oder macOS-Clients](#)

Konfigurieren eines Windows-Clients

Vorbereitung

Die folgenden Informationen benötigen Sie, um den SSH-Tunnel einzurichten:

- Der öffentliche DNS-Name Ihrer Amazon-EC2-Instance. Den öffentlichen DNS-Namen finden Sie in der EC2-Konsole.
- Das Schlüsselpaar für die private Schlüsseldatei. Weitere Informationen zum Zugriff auf Ihr Schlüsselpaar finden Sie unter [Amazon EC2-Schlüsselpaare](#) im Amazon EC2-Benutzerhandbuch für Linux-Instances.

Verwenden von Jupyter-Notebooks von einem Windows-Client

Informationen zu Ihrer Amazon-EC2-Instance von einem Windows-Client finden Sie in diesen Anleitungen.

1. [Problembehandlung beim Herstellen einer Verbindung zu Ihrer Instance](#)
2. [Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY](#)

Um einen Tunnel für einen laufenden Jupyter-Server zu erstellen, wird empfohlen, Git Bash auf Ihrem Windows-Client zu installieren und anschließend den Linux/macOS-Client-Anweisungen zu folgen. Sie können jedoch auch einen anderen Ansatz für das Öffnen eines SSH-Tunnels mit Port-Zuweisung wählen. Weitere Informationen finden Sie in der [Jupyter-Dokumentation](#).

Nächster Schritt

[Konfigurieren eines Linux- oder macOS-Clients](#)

Konfigurieren eines Linux- oder macOS-Clients

1. Öffnen Sie ein Terminalfenster.
2. Führen Sie den folgenden Befehl aus, um alle Anfragen auf dem lokalen Port 8888 an Port 8888 auf Ihrer Remote-Amazon-EC2-Instance weiterzuleiten. Aktualisieren Sie den Befehl, indem Sie den Speicherort Ihres Schlüssels für den Zugriff auf die Amazon EC2 EC2-Instance und den öffentlichen DNS-Namen Ihrer Amazon EC2 EC2-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 Amazon-EC2-Remote-Instance, auf der der Jupyter-Notebook-Server ausgeführt wird.

Nächster Schritt

[Testen des Jupyter-Notebook-Servers mit einer Anmeldung](#)

Testen des Jupyter-Notebook-Servers mit einer Anmeldung

Sie können sich jetzt am Jupyter Notebook-Server anmelden.

Ihr nächster Schritt ist das Testen der Verbindung zum Server über Ihren Browser.

1. Geben Sie in der Adressleiste Ihres Browsers die folgende URL ein, oder klicken Sie auf diesen Link: <https://localhost:8888>.
2. Mit einem selbstsignierten SSL-Zertifikat warnt Sie Ihr Browser und fordert Sie auf, den Besuch der Website abzubrechen.

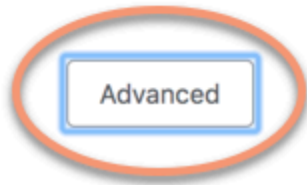


Your connection is not private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

Help improve Safe Browsing by sending some [system information and page content](#) to Google.
[Privacy policy](#)



Back to safety

Da Sie dies selbst eingerichtet haben, können Sie bedenkenlos fortfahren. Abhängig von Ihrem Browser wird die Schaltfläche "Erweitert", "Details anzeigen" oder eine ähnliche angezeigt.



Your connection is not private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

Help improve Safe Browsing by sending some [system information and page content](#) to Google.
[Privacy policy](#)

Hide advanced

Back to safety

This server could not prove that it is **localhost**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to localhost \(unsafe\)](#)

Klicken Sie darauf und anschließend auf den Link "Weiter zu localhost". Wenn die Verbindung erfolgreich ist, sehen Sie die Webseite des Jupyter-Notebook-Servers. Zu diesem Zeitpunkt werden Sie aufgefordert, das von Ihnen zuvor eingerichtete Passwort einzugeben.

Jetzt haben Sie Zugriff auf den Jupyter-Notebook-Server, der auf dem DLAMI läuft. Sie können nun neue Notebooks erstellen oder die bereitgestellten [Tutorials](#) ausführen.

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 [Einrichten eines Jupyter-Notebook-Servers](#)

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 einer 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`)

Sie können beispielsweise Environment (`conda_mxnet_p36`) sehen, was bedeutet, dass die Umgebung MXNet und Python 3 enthält. Die andere Variante davon wäre Environment (`conda_mxnet_p27`), also eine Umgebung mit MXNet und Python 2.

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 Startseite 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

- [10-Minuten-Tutorials](#)
- [Aktivieren von Frameworks](#)
- [Verteiltes Training mit Elastic Fabric Adapter](#)
- [GPU-Überwachung und -Optimierung](#)
- [Der AWS Inferentia-Chip mit DLAMI](#)
- [Das ARM64 DLAMI](#)
- [Inferenz](#)

- [Modellbereitstellung](#)

10-Minuten-Tutorials

- [Starten Sie a AWS Deep Learning AMI \(in 10 Minuten\)](#)
- [Trainieren Sie ein Deep-Learning-Modell mit DLC auf Amazon EC2 \(in 10 Minuten\)](#)

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.

- (Option für Python 3) — Aktiviere die PyTorch Python-3-Umgebung:

```
$ source activate pytorch_p310
```

- 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
```

- (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 Instanzen ohne GPUs:

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

- 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 Beispiel-Repository ausführen: PyTorch

```
(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 [PyTorch](#) Frameworks.

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 [TensorFlow](#) auf 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 AMI](#)
- [EFA auf dem DLAMI verwenden](#)

Starten einer Instance mit EFA AWS Deep Learning AMI

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 eingehenden und ausgehenden Datenverkehr zur und von der Sicherheitsgruppe selbst zulässt. [Weitere Informationen finden Sie in der EFA-Dokumentation.](#)

1. Öffnen Sie die Amazon EC2-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 Registerkarten Eingehend und Ausgehend 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 AMI wird derzeit von den folgenden Instance-Typen und Betriebssystemen unterstützt:

- P3dn.24xlarge: Amazon Linux 2, Ubuntu 20.04
- P4D.24xlarge: Amazon Linux 2, Ubuntu 20.04
- P5.48xlarge: Amazon Linux 2, Ubuntu 20.04

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 Amazon EC2-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 zudem optional eine primäre IPv4-Adresse und eine oder mehrere sekundäre IPv4-Adressen eingeben. Wenn Sie die Instance in einem Subnetz starten, das einen verknüpften IPv6-CIDR-Block hat, 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**.

- 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 Instance in der EC2-Konsole aus und sehen Sie sich die Registerkarte "Description (Beschreibung)" im unteren Bereich auf 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 der EC2-Instance und Starten einer neuen Instance mit den gleichen Schritten. Stellen Sie sicher, dass die EFA angeschlossen ist.
- Hängen Sie EFA an eine bestehende Instanz an.
 1. Wechseln Sie in der EC2-Konsole zu "Network Interfaces (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. Wechseln Sie zurück zur Registerkarte "EC2 Instances (EC-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 Regeln für eingehende/ausgehende Nachrichten 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 AMI

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 1.30.0 und dem OFI NCCL Plugin 1.7.4 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

Funktionstests

NCCL-Nachrichtenübertragung, Test mit mehreren Knoten

"nccl_message_transfer" ist ein einfacher Test, um sicherzustellen, dass das NCCL OFI-Plugin wie erwartet funktioniert. Der Test überprüft die Funktionalität der Verbindungseinrichtungs- und Datentransfer-APIs von NCCL. Stellen Sie sicher, dass Sie den vollständigen Pfad zu mpirun verwenden, wie im Beispiel gezeigt, wenn Sie NCCL-Anwendungen mit EFA ausführen. Ändern Sie die Parameter np und N basierend auf der Anzahl der Instances und GPUs in Ihrem Cluster. [Weitere Informationen finden Sie in der OFI NCCL-Dokumentation.AWS](#)

Der folgende nccl_message_transfer-Test gilt für eine generische CUDA xx.x-Version. Sie können die Befehle für jede verfügbare CUDA-Version in Ihrer Amazon EC2 EC2-Instance ausführen, indem Sie die CUDA-Version im Skript ersetzen.

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

Die Ausgabe sollte wie folgt aussehen. Sie können anhand der Ausgabe überprüfen, ob EFA als OFI-Anbieter verwendet wird.

```
INFO: Function: nccl_net_ofi_init Line: 1069: NET/OFI Selected Provider is efa (found 4  
 nics)  
INFO: Function: nccl_net_ofi_init Line: 1160: NET/OFI Using transport protocol SENDRECV  
INFO: Function: configure_ep_inorder Line: 261: NET/OFI Setting  
 FI_OPT_EFA_SENDRECV_IN_ORDER_ALIGNED_128_BYTES not supported.  
INFO: Function: configure_nccl_proto Line: 227: NET/OFI Setting NCCL_PROTO to "simple"  
INFO: Function: main Line: 86: NET/OFI Process rank 1 started. NCCLNet device used on  
 ip-172-31-13-179 is AWS Libfabric.  
INFO: Function: main Line: 91: NET/OFI Received 4 network devices  
INFO: Function: main Line: 111: NET/OFI Network supports communication using CUDA  
 buffers. Dev: 3  
INFO: Function: main Line: 118: NET/OFI Server: Listening on dev 3  
INFO: Function: main Line: 131: NET/OFI Send connection request to rank 1  
INFO: Function: main Line: 173: NET/OFI Send connection request to rank 0  
INFO: Function: main Line: 137: NET/OFI Server: Start accepting requests  
INFO: Function: main Line: 141: NET/OFI Successfully accepted connection from rank 1  
INFO: Function: main Line: 145: NET/OFI Send 8 requests to rank 1  
INFO: Function: main Line: 179: NET/OFI Server: Start accepting requests
```

```
INFO: Function: main Line: 183: NET/OFI Successfully accepted connection from rank 0
INFO: Function: main Line: 187: NET/OFI Rank 1 posting 8 receive buffers
INFO: Function: main Line: 161: NET/OFI Successfully sent 8 requests to rank 1
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 0
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 1
```

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 Instances und GPUs in Ihrem Cluster.
- 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 Mitglieds-knoten, 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 Amazon EC2 EC2-Instance ausführen, indem Sie die CUDA-Version im Skript ersetzen.

- Amazon Linux 2:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO -x --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 -x NCCL_PROTO=simple -b 8 -e
1G -f 2 -g 1 -c 1 -n 100 | tee ${TRAINING_LOG}
```

- Ubuntu 20.04:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO -x --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 -x NCCL_PROTO=simple-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 9591 on ip-172-31-4-37 device 0 [0x10] NVIDIA A100-SXM4-40GB
# Rank 1 Group 0 Pid 9592 on ip-172-31-4-37 device 1 [0x10] NVIDIA A100-SXM4-40GB
# Rank 2 Group 0 Pid 9593 on ip-172-31-4-37 device 2 [0x20] NVIDIA A100-SXM4-40GB
# Rank 3 Group 0 Pid 9594 on ip-172-31-4-37 device 3 [0x20] NVIDIA A100-SXM4-40GB
# Rank 4 Group 0 Pid 9595 on ip-172-31-4-37 device 4 [0x90] NVIDIA A100-SXM4-40GB
# Rank 5 Group 0 Pid 9596 on ip-172-31-4-37 device 5 [0x90] NVIDIA A100-SXM4-40GB
# Rank 6 Group 0 Pid 9597 on ip-172-31-4-37 device 6 [0xa0] NVIDIA A100-SXM4-40GB
# Rank 7 Group 0 Pid 9598 on ip-172-31-4-37 device 7 [0xa0] NVIDIA A100-SXM4-40GB
# Rank 8 Group 0 Pid 10216 on ip-172-31-13-179 device 0 [0x10] NVIDIA A100-
SXM4-40GB
# Rank 9 Group 0 Pid 10217 on ip-172-31-13-179 device 1 [0x10] NVIDIA A100-
SXM4-40GB
# Rank 10 Group 0 Pid 10218 on ip-172-31-13-179 device 2 [0x20] NVIDIA A100-
SXM4-40GB
# Rank 11 Group 0 Pid 10219 on ip-172-31-13-179 device 3 [0x20] NVIDIA A100-
SXM4-40GB
# Rank 12 Group 0 Pid 10220 on ip-172-31-13-179 device 4 [0x90] NVIDIA A100-
SXM4-40GB
```

```

# Rank 13 Group 0 Pid 10221 on ip-172-31-13-179 device 5 [0x90] NVIDIA A100-
SXM4-40GB
# Rank 14 Group 0 Pid 10222 on ip-172-31-13-179 device 6 [0xa0] NVIDIA A100-
SXM4-40GB
# Rank 15 Group 0 Pid 10223 on ip-172-31-13-179 device 7 [0xa0] NVIDIA A100-
SXM4-40GB
ip-172-31-4-37:9591:9591 [0] NCCL INFO Bootstrap : Using ens32:172.31.4.37
ip-172-31-4-37:9591:9591 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin_v6
symbol.
ip-172-31-4-37:9591:9591 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin
symbol (v4 or v5).
ip-172-31-4-37:9591:9591 [0] NCCL INFO cudaDriverVersion 12020
NCCL version 2.18.5+cuda12.2
...
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.7.4-aws
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Using CUDA runtime version 11070
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Using CUDA runtime version 11070
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
variable to 1
ip-172-31-4-37:9024:9062 [6] NCCL INFO NET/OFI Disabling NVLS support due to NCCL
version 21602
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
variable to 1
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Disabling NVLS support due to NCCL
version 21602
ip-172-31-4-37:9020:9063 [2] NCCL INFO NET/OFI Running on p4d.24xlarge platform,
Setting NCCL_TOPO_FILE environment variable to /opt/aws-ofi-nccl/share/aws-ofi-nccl/
xml/p4d-24x1-topo.xml
...
-----some output truncated-----
#
#                               out-of-place
#           in-place
#           size      count      type  redop  root  time  algbw  busbw #wrong
#           time     algbw  busbw #wrong
#           (us)     (B)    (elements)
#           (us)     (GB/s) (GB/s)
#           0         0      float  sum    -1    11.02  0.00  0.00  0
11.04  0.00  0.00  0
#           0         0      float  sum    -1    11.01  0.00  0.00  0
11.00  0.00  0.00  0

```

0	0	float	sum	-1	11.02	0.00	0.00	0
11.02	0.00	0.00	0					
0	0	float	sum	-1	11.01	0.00	0.00	0
11.00	0.00	0.00	0					
0	0	float	sum	-1	11.02	0.00	0.00	0
11.02	0.00	0.00	0					
256	4	float	sum	-1	632.7	0.00	0.00	0
628.2	0.00	0.00	0					
512	8	float	sum	-1	627.4	0.00	0.00	0
629.6	0.00	0.00	0					
1024	16	float	sum	-1	632.2	0.00	0.00	0
631.7	0.00	0.00	0					
2048	32	float	sum	-1	631.0	0.00	0.00	0
634.2	0.00	0.00	0					
4096	64	float	sum	-1	623.3	0.01	0.01	0
633.6	0.01	0.01	0					
8192	128	float	sum	-1	635.1	0.01	0.01	0
633.5	0.01	0.01	0					
16384	256	float	sum	-1	634.8	0.03	0.02	0
637.0	0.03	0.02	0					
32768	512	float	sum	-1	647.9	0.05	0.05	0
636.8	0.05	0.05	0					
65536	1024	float	sum	-1	658.9	0.10	0.09	0
667.0	0.10	0.09	0					
131072	2048	float	sum	-1	671.9	0.20	0.18	0
662.9	0.20	0.19	0					
262144	4096	float	sum	-1	692.1	0.38	0.36	0
685.1	0.38	0.36	0					
524288	8192	float	sum	-1	715.3	0.73	0.69	0
696.6	0.75	0.71	0					
1048576	16384	float	sum	-1	734.6	1.43	1.34	0
729.2	1.44	1.35	0					
2097152	32768	float	sum	-1	785.9	2.67	2.50	0
794.5	2.64	2.47	0					
4194304	65536	float	sum	-1	837.2	5.01	4.70	0
837.6	5.01	4.69	0					
8388608	131072	float	sum	-1	929.2	9.03	8.46	0
931.4	9.01	8.44	0					
16777216	262144	float	sum	-1	1773.6	9.46	8.87	0
1772.8	9.46	8.87	0					
33554432	524288	float	sum	-1	2110.2	15.90	14.91	0
2116.1	15.86	14.87	0					
67108864	1048576	float	sum	-1	2650.9	25.32	23.73	0
2658.1	25.25	23.67	0					

```

134217728      2097152      float      sum      -1      3943.1      34.04      31.91      0
3945.9  34.01  31.89      0
268435456      4194304      float      sum      -1      7216.5      37.20      34.87      0
7178.6  37.39  35.06      0
536870912      8388608      float      sum      -1      13680      39.24      36.79      0
13676  39.26  36.80      0
[ 1073741824      16777216      float      sum      -1      25645      41.87      39.25      0
25497  42.11  39.48      0 ] <- Used For Benchmark
...
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 7.46044

```

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 [Leistungstests](#)
- 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

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

```

```
# ip-172-31-27-205:6427:6820 [0] NCCL INFO NET/OFI Using CUDA runtime version
11060

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

# Validation of logs
grep "NET/OFI Using CUDA runtime version ${CUDA_RUNTIME_VERSION}" ${TRAINING_LOG}
|| { echo "Runtime cuda text not found"; exit 1; }
grep "NET/OFI Initializing aws-ofi-nccl" ${TRAINING_LOG} || { echo "aws-ofi-nccl
is not working, please check if it is installed correctly"; exit 1; }
grep "NET/OFI Configuring AWS-specific options" ${TRAINING_LOG} || { echo "AWS-
specific options text not found"; exit 1; }
grep "Using network AWS Libfabric" ${TRAINING_LOG} || { echo "AWS Libfabric text
not found"; exit 1; }
grep "busbw" ${TRAINING_LOG} || { echo "busbw text not found"; exit 1; }
grep "Avg bus bandwidth " ${TRAINING_LOG} || { echo "Avg bus bandwidth text not
found"; exit 1; }
grep "NCCL version $NCCL_VERSION" ${TRAINING_LOG} || { echo "Text not found: NCCL
version $NCCL_VERSION"; exit 1; }

if [[ ${INSTANCE_TYPE} == "p4d.24xlarge" ]]; then
    grep "NET/AWS Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Text not found:
NET/AWS Libfabric/0/GDRDMA"; exit 1; }
    grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Selected Provider is efa text not found"; exit 1; }
    grep "aws-ofi-nccl/xml/p4d-24x1-topo.xml" ${TRAINING_LOG} || { echo "Topology
file not found"; exit 1; }
elif [[ ${INSTANCE_TYPE} == "p4de.24xlarge" ]]; then
    grep "NET/AWS Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus
bandwidth text not found"; exit 1; }
    grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
    grep "aws-ofi-nccl/xml/p4de-24x1-topo.xml" ${TRAINING_LOG} || { echo
"Topology file not found"; exit 1; }
elif [[ ${INSTANCE_TYPE} == "p5.48xlarge" ]]; then
    grep "NET/AWS Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus
bandwidth text not found"; exit 1; }
    grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
```



```

    grep "aws-ofi-nccl/xml/p5.48x1-topo.xml" ${TRAINING_LOG} || { echo "Topology
file not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p3dn.24xlarge" ]]; then
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Selected Provider is efa text not found"; exit 1; }
    fi
    echo "***** check_efa_nccl_all_reduce passed for cuda
version ${CUDA_VERSION} *****"
else
    echo "***** check_efa_nccl_all_reduce failed for cuda
version ${CUDA_VERSION} *****"
fi

```

- 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 GPUs mit 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 GPUs mit 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 AMI (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 pynvml-Paket aus [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 GPUs mit 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 Amazon EC2 EC2-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 Amazon EC2 EC2-Instances konfigurieren, an die Ihre Instance Metriken weiterleiten kann. CloudWatch Ausführliche Schritte finden Sie unter [IAM-Rollen und -Benutzer für die Verwendung mit dem Agenten erstellen](#). CloudWatch

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 DLAMIs sind mit den neuesten verfügbaren AWS CloudWatch Agenten-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, 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 Daten, an die gesendet werden, 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: PutMetric Data“. 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 `mitstore_reso`.
- 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
```

- Führen Sie das `gpumon`-Dienstprogramm im Hintergrund aus.

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

- Öffnen Sie in Ihrem Browser <https://console.aws.amazon.com/cloudwatch/> und wählen Sie Metrik aus. Es wird einen Namespace 'DeepLearningTrain' haben.

Tip

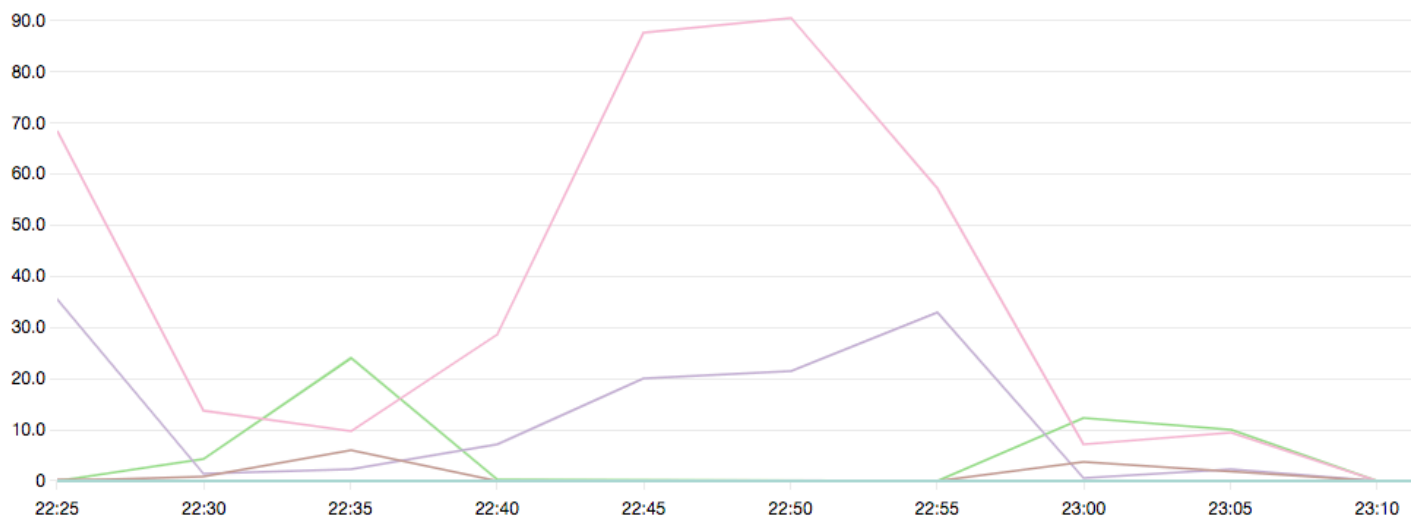
Sie können den Namespace durch Modifizierung von `gpumon.py` ändern. Sie können auch das Berichtsintervall durch Anpassung von `store_reso` ändern.

Im Folgenden finden Sie ein CloudWatch Beispieldiagramm, das über einen Lauf von `gpumon.py` berichtet, der einen Trainingsjob auf der `p2.8xlarge`-Instance überwacht.

GPU usage, Memory usage 

1h 3h 12h 1d 3d 1w custom

Various units



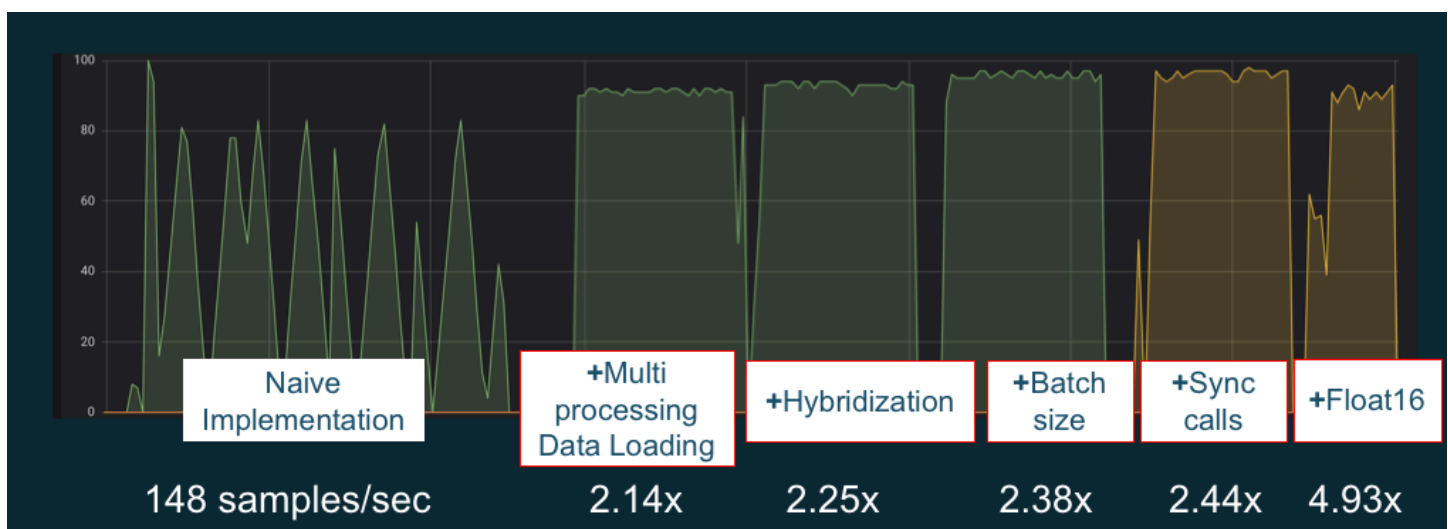
Möglicherweise sind diese weiteren Themen zur GPU-Überwachung und -Optimierung für Sie interessant:

- [Überwachen](#)
 - [Überwachen Sie GPUs mit CloudWatch](#)
- [Optimierung](#)
 - [Vorverarbeitung](#)
 - [Training](#)

Optimierung

Um Ihre GPUs vollständig nutzen zu können, können Sie Ihre Datenpipeline optimieren und Ihr Deep Learning-Netzwerk feineinstellen. 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: [Performance Tricks with 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 GPUs mit 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, die die Mixed-Precision-Implementierung für MXNet beschreiben, und 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.
- [Mixed-Precision-Schulung unter Verwendung von float16 mit MXNet](#): Artikel mit häufig gestellten Fragen auf der MXNet-Website.
- [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 GPUs mit 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 AMI (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 AMI (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 MXNet-Neuron und dem Neuron Compiler AWS](#)
- [Verwenden der MXNet-Neuron-Modellbereitstellung](#)
- [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
tf.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 Beispieldownload 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 MXNet-Neuron und dem Neuron Compiler AWS

Die MXNet-Neuron-Kompilierungs-API bietet eine Methode zum Kompilieren eines Modelldiagramms, das Sie auf einem AWS Inferentia-Gerät ausführen können.

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 folgendem 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 für die MXNet-Neuron-Kompilierung, 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/aws-labs/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 der MXNet-Neuron-Modellbereitstellung](#)

Verwenden der MXNet-Neuron-Modellbereitstellung

In diesem Tutorial lernen Sie, ein vorgeschultes MXNet-Modell für die Echtzeit-Bildklassifizierung mit MMS (Multi Model Server) zu verwenden. 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 werden. 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 folgendem 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/aws-labs/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-Geräteziel.

```
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 Eingabennamen 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/
squeeze_net_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 außerdem die nicht benötigte Datenkopie für `model_input` auskommentieren, da MXNet-Neuron die `NDArray`- und die `Gluon-API` nicht unterstützt.

```
...
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 Multimodell-Server und laden Sie das Modell, das die RESTful-API verwendet, mit den folgenden Befehlen. 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"  
}  
]
```

Geben Sie zur Nachbereitung des Tests einen Löschbefehl über die RESTful-API aus und beenden Sie den Modell-Server 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-GPU-DLAMIs wurden entwickelt, um eine hohe Leistung und Kosteneffizienz für Deep-Learning-Workloads zu 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-GPU-DLAMIs sind mit Docker, NVIDIA Docker, NVIDIA-Treiber, 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-step Anweisungen zum Starten eines DLAMI finden Sie unter [Starten und Konfigurieren eines DLAMI](#).

Eine Liste der neuesten ARM64-DLAMIs 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 DLAMI PyTorch](#)

Verwenden der ARM64-GPU DLAMI PyTorch

Der AWS Deep Learning AMI ist sofort einsatzbereit für GPUs mit Arm64-Prozessoren und ist optimiert für PyTorch. Das PyTorch ARM64-GPU-DLAMI umfasst eine Python-Umgebung, die mit [PyTorch](#), [TorchVision](#) und [TorchServe](#) für 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 MNIST-Beispiel-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": 0.00023322898778133094
}
```

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 Serversystem für Modelle des maschinellen Lernens.

Das `tensorflow-serving-api` ist mit Deep Learning AMI mit Conda vorinstalliert! Ein Beispielskript für das Trainieren, Exportieren und Bereitstellen eines MNIST-Modells finden Sie in `~/examples/tensorflow-serving/`.

Um eines dieser Beispiele auszuführen, stellen Sie zunächst mit Conda eine Verbindung zu Ihrem Deep Learning-AMI her und aktivieren Sie die TensorFlow Umgebung.

```
$ source activate tensorflow2_p310
```

Jetzt setzen Sie die Verzeichnisse auf den Ordner mit den Beispiel-Skripts.

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

Bereitstellen eines vorgeschulten Inception-Modells

Im Folgenden finden Sie ein Beispiel, mit dem Sie die Bereitstellung verschiedener Modelle wie Inception testen können. In der Regel benötigen Sie ein servables Modell und Client-Skripte, die bereits auf Ihr DLAMI heruntergeladen wurden.

Bereitstellen und Testen von Inference mit einem Inception-Modell

1. Laden Sie das Modell herunter.

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

2. Entpacken Sie das Modell.

```
$ unzip INCEPTION.zip
```

3. Laden Sie das Bild eines Huskys herunter.

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

4. Starten Sie den Server. Beachten Sie, dass Sie in Amazon Linux das Verzeichnis, das für `model_base_path` verwendet wird, von `/home/ubuntu` in `/home/ec2-user` ändern müssen.

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

5. Wenn der Server im Vordergrund läuft, müssen Sie eine weitere Terminalsitzung starten, um fortzufahren. Öffnen Sie ein neues Terminal und aktivieren Sie es TensorFlow mit `source activate tensorflow2_p310`. Verwenden Sie anschließend den von Ihnen bevorzugten Texteditor, um ein Skript mit folgendem Inhalt zu erstellen. Geben Sie ihr den Namen `inception_client.py`. Dieses Skript verwendet einen Image-Dateinamen als Parameter und ruft ein Voraussageergebnis von dem vorgeschulten Modell ab.

```
from __future__ import print_function

import grpc
import tensorflow as tf
import argparse

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

parser = argparse.ArgumentParser(
    description='TF Serving Test',
    formatter_class=argparse.ArgumentDefaultsHelpFormatter
)
```

```
parser.add_argument('--server_address', default='localhost:9000',
                    help='Tenforflow Model Server Address')
parser.add_argument('--image', default='Siberian_Husky_bi-eyed_Flickr.jpg',
                    help='Path to the image')
args = parser.parse_args()

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

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

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

    print("Inception Client Passed")

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

6. Führen Sie nun das Skript aus und geben Sie Server-Standort sowie Port und Dateinamen des Husky-Bilds als Parameter weiter.

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


Schulen und Bereitstellen eines MNIST-Modells

Für dieses Tutorial exportieren wir ein Modell und stellen es mit der Anwendung `tensorflow_model_server` bereit. Schließlich können Sie den Modell-Server mit einem Beispiel-Client-Skript testen.

Führen Sie das Skript aus, das ein MNIST Modell trainiert und exportiert. Als einziges Argument für das Skript müssen Sie einen Ordnerspeicherort angeben, wo es das Modell speichern kann. Hier können wir es einfach in `mnist_model` ablegen. Das Skript erstellt den Ordner für Sie.

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

Haben Sie Geduld, da dieses Skript kann eine Weile brauchen kann, bevor es etwas ausgibt. Wenn das Training abgeschlossen ist und das Modell schließlich exportiert wurde, sollten Sie Folgendes sehen:

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

Ihr nächster Schritt besteht darin, den `tensorflow_model_server` auszuführen, um das exportierte Modell bereitzustellen.

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

Ein Client-Skript wird bereitgestellt, damit Sie den Server testen können.

Wenn Sie einen Test durchführen, müssen Sie ein neues Terminal-Fenster öffnen.

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

Weitere Funktionen und Beispiele

Wenn Sie mehr über TensorFlow Serving erfahren möchten, besuchen Sie die [TensorFlow Website](#).

Sie können TensorFlow Serving auch mit [Amazon Elastic Inference](#) verwenden. Weitere Informationen finden Sie in der Anleitung zur [Verwendung von Elastic Inference with TensorFlow Serving](#).

TorchServe

TorchServe ist ein flexibles Tool zur Bereitstellung von Deep-Learning-Modellen, aus PyTorch denen exportiert wurden. 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 eine Connect zu einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance mit Deep Learning AMI mit Conda v34 oder höher 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.

Aktualisieren Sie Ihr DLAMI

Hier finden Sie Informationen zum Upgrade Ihres DLAMI und Tipps zur Aktualisierung der Software auf Ihrem DLAMI.

Themen

- [Upgrade auf eine neue DLAMI-Version](#)
- [Tipps für Software-Updates](#)
- [Erhalten Sie Benachrichtigungen über neue Updates](#)

Upgrade auf eine neue DLAMI-Version

Die Systemabbilder 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 DLAMIs verschieben, müssen Sie sowohl die DLAMIs als auch das neue Volume in derselben Availability Zone haben.

1. Verwenden Sie die Amazon EC2-Konsole, 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 [Ein DLAMI starten und konfigurieren](#).

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](#).

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 verwenden `pip`, 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 AMI 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 haben 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 Lambda, 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 AMI

Cloud-Sicherheit hat AWS höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS 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 Dienste in der AWS Cloud ausführt. 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 DLAMI gelten, finden Sie unter [AWS Services in Scope by Compliance Program AWS](#) Program.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die 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 AWS Dienste nutzen können, die Ihnen helfen, Ihre DLAMI-Ressourcen zu überwachen und zu sichern.

Weitere Informationen finden Sie unter [Sicherheit in Amazon EC2](#).

Themen

- [Datenschutz in AWS Deep Learning AMI](#)
- [Identity and Access Management in AWS Deep Learning AMI](#)
- [Anmeldung und Überwachung AWS Deep Learning AMI](#)
- [Konformitätsvalidierung für AWS Deep Learning AMI](#)
- [Resilienz in AWS Deep Learning AMI](#)
- [Sicherheit der Infrastruktur in AWS Deep Learning AMI](#)

Datenschutz in AWS Deep Learning AMI

Das AWS [Modell](#) der mit gilt für den Datenschutz in AWS Deep Learning AMI. Wie in diesem Modell beschrieben, AWS ist es für den Schutz der globalen Infrastruktur verantwortlich, 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).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail
- 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 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-2](#).

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 AWS SDKs arbeiten. 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.

Identity and Access Management in AWS Deep Learning AMI

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 zum Identity and Access Management finden Sie unter [Identity and Access Management 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 Zugangportal 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,

um Anfragen selbst zu [signieren](#), finden Sie im IAM-Benutzerhandbuch unter [AWS API-Anfragen signieren](#).

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. 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 [Verwenden der Multi-Faktor-Authentifizierung \(MFA\) in AWS](#) im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie ein AWS-Konto erstellen, 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 und verwenden Sie diese, 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 Ihrem 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 einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten 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 [Erstellen eines IAM-Benutzers \(anstatt einer Rolle\)](#) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine [IAM-Rolle](#) ist eine Identität innerhalb Ihres Unternehmens AWS-Konto, die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen [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 [Verwenden von IAM-Rollen](#) 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](#) 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 zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie unter [Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch](#).

- **Serviceübergreifender Zugriff** — Einige verwenden Funktionen in anderen. AWS-Services AWS-Services Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon-EC2 aus oder speichert Objekte in Amazon-S3. 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 EC2 ausgeführte Anwendungen** — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und API-Anfragen stellen AWS CLI . AWS Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter [Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden](#) im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter [Erstellen einer IAM-Rolle \(anstatt eines Benutzers\)](#) im IAM-Benutzerhandbuch.

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. Berechtigungen in den Richtlinien bestimmen, ob die 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 bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

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 [Erstellen von IAM-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 eingebundenen Richtlinie wählen, finden Sie unter [Auswahl 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.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter [Zugriffskontrollliste \(ACL\) – Übersicht](#) (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

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.

- **Service Control Policies (SCPs)** — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. **AWS Organizations** ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten, die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Entitäten. Root-Benutzer des AWS-Kontos Weitere Informationen zu Organizations und SCPs finden Sie unter [Funktionsweise von SCPs](#) 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 Amazon EMR verwenden **AWS Identity and Access Management**, um Benutzer, AWS Ressourcen, Gruppen, Rollen und Richtlinien zu definieren. Sie können auch steuern, auf welche AWS Dienste diese 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](#).

Anmeldung und Überwachung AWS Deep Learning AMI

Ihre AWS Deep Learning AMI 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 and -Optimierung](#) und [Überwachen von Amazon EC2](#).

Nachverfolgung der Nutzung

Die folgenden AWS Deep Learning AMI Betriebssystemverteilungen enthalten Code, mit dem Instanztyp, Instanz-ID, DLAMI-Typ und Betriebssysteminformationen erfasst werden können AWS . 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.

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

Um die Nutzungsverfolgung für Ihr DLAMI zu deaktivieren, fügen Sie Ihrer Amazon EC2 EC2-Instance beim Start ein Tag hinzu. Das Tag sollte den Schlüssel verwenden, dessen OPT_OUT_TRACKING zugeordneter Wert auf gesetzt ist. `true` Weitere Informationen finden Sie unter [Taggen Ihrer Amazon EC2 EC2-Ressourcen](#).

Konformitätsvalidierung für AWS Deep Learning AMI

Externe Prüfer bewerten die Sicherheit und Einhaltung von Vorschriften im AWS Deep Learning AMI Rahmen mehrerer AWS Compliance-Programme. Weitere Informationen zu den unterstützten Compliance-Programmen finden Sie unter [Compliance-Validierung für Amazon EC2](#).

Eine Liste der AWS Services im Rahmen bestimmter Compliance-Programme finden Sie unter [AWS Services im Umfang nach Compliance-Programmen 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, um Sie bei der Einhaltung von Vorschriften zu unterstützen:

- [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 Arbeitsmapen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- [Bewertung von Ressourcen anhand von Regeln](#) im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- [AWS Security Hub](#)— Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus, sodass Sie überprüfen können AWS, ob Sie die Sicherheitsstandards und Best Practices der Branche einhalten.

Resilienz in AWS Deep Learning AMI

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability 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 AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Informationen zu Funktionen zur Unterstützung Ihrer Datenresilienz- und Sicherungsanforderungen finden Sie unter [Resilience in Amazon EC2](#).

Sicherheit der Infrastruktur in AWS Deep Learning AMI

Die Infrastruktursicherheit von AWS Deep Learning AMI wird von Amazon EC2 unterstützt. Weitere Informationen finden Sie unter [Infrastruktursicherheit in Amazon EC2](#).

Wichtige Änderungen an DLAMI

Häufig gestellte Fragen

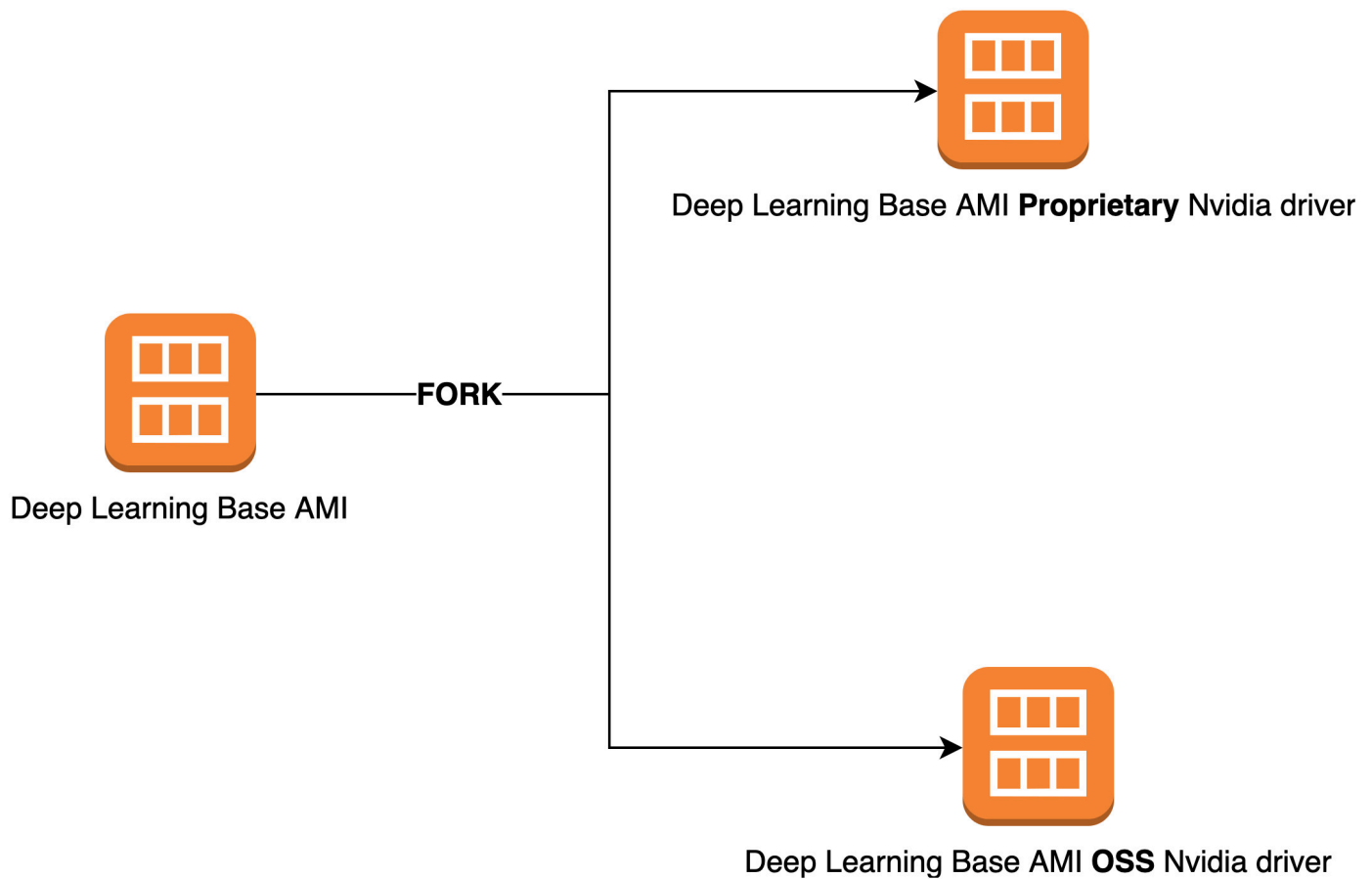
- [Was ändert sich?](#)
- [Warum ist diese Änderung erforderlich?](#)
- [Welche DLAMIs sind von dieser Änderung betroffen?](#)
- [Was bedeutet das für dich?](#)
- [Wann sollten Sie mit der Verwendung der neuen DLAMIs beginnen?](#)
- [Wird es bei den neuen DLAMIs zu Funktionseinbußen kommen?](#)
- [Was ist mit DLCs?](#)

Was ändert sich?

Am 15.11.2023 wird AWS Deep Learning AMI (DLAMIs) in zwei separate Gruppen aufgeteilt:

- DLAMIs, die einen proprietären Treiber von Nvidia verwenden (zur Unterstützung von P3, P3dn, G3).
- DLAMs, die den Nvidia OSS-Treiber verwenden (zur Unterstützung von G4dn, G5, P4, P5).

Infolgedessen werden für jede der beiden Kategorien neue DLAMIs mit neuen Namen und neuen AMI-IDs erstellt. Diese DLAMIs werden nicht austauschbar sein — d. h. DLAMIs aus einer Gruppe unterstützen keine Instances, die von der anderen Gruppe unterstützt werden, z. B. unterstützt das DLAMI, das p5 unterstützt, G3 nicht und umgekehrt.



Warum ist diese Änderung erforderlich?

Derzeit enthalten DLAMIs für NVIDIA-GPUs einen proprietären Kernel-Treiber von NVIDIA. Vor Kurzem hat die Upstream-Linux-Kernel-Community jedoch eine Änderung akzeptiert, die proprietäre Kerneltreiber wie den NVIDIA-GPU-Treiber von der Kommunikation mit anderen Kerneltreibern isoliert. Diese Änderung deaktiviert GPUDirect RDMA auf Instances der P4/P5-Serie. Dies ist der Mechanismus, der es GPUs ermöglicht, EFA effizient für verteilte Schulungen zu nutzen. Aus diesem Grund wird DLAMIS den OpenRM-Treiber (NVIDIA-Open-Source-Treiber) verwenden, 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 (P3, G3 usw.) Um sicherzustellen, dass wir weiterhin aktuelle, leistungsstarke und sichere DLAMIs bereitstellen, die beide Instanztypen unterstützen, werden wir DLAMIs in zwei Gruppen aufteilen — eine mit dem OpenRM-Treiber (unterstützt G4dn, G5, P4 und P5) und eine mit dem älteren proprietären Treiber (unterstützt ältere Instanzen P3, P3dn, G3).

Welche DLAMIs sind von dieser Änderung betroffen?

Alle DLAMIs sind von dieser Änderung betroffen.

Was bedeutet das für dich?

Die neuen DLAMIs bieten weiterhin die Funktionalität, Leistung und Sicherheit der aktuellen DLAMIs, solange sie auf einem kompatiblen Instance-Typ ausgeführt werden. Wenn Sie DLAMIs verwenden, müssen Sie sicherstellen, dass ein DLAMI auf einer der kompatiblen Instanzen gestartet wird, die in den Versionshinweisen der einzelnen DLAMIs aufgeführt sind (siehe hier). Beispiel: Sie müssen diese Änderung berücksichtigen, um:

- Rufen Sie DLAMIs mit den richtigen CLI-Abfragen auf (siehe unten)
- Starten Sie DLAMIs über die Konsole und die CLI auf einem kompatiblen Instanztyp

Wenn Sie DLAMIs von der EC2-Konsole aus starten Schnellstart: In jeder DLAMI-Beschreibung sind die Typen von Instances aufgeführt, die in der EC2-Konsole unterstützt werden. Sie sollten die DLAMIs auf kompatiblen Instances starten.

ubuntu Verified provider	Deep Learning Base GPU AMI (Ubuntu 20.04) 20231018 ami-05f9aedeafddcf112 (64-bit (x86)) Supported EC2 instances: P5*, P4*, P3*, G3*, G5*, G4dn. Release notes: https://aws.amazon.com/releases/notes/aws-deep-learning-base-gpu-ami-ubuntu-20-04/	Select	64-bit (x86)
ubuntu Verified provider	Deep Learning AMI GPU PyTorch 2.0.1 (Ubuntu 20.04) 20231003 ami-005656037407fcf99 (64-bit (x86)) Supported EC2 instances: P5, P4d, P4de, P3, P3dn, G5, G4dn, G3. Release notes: https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html	Select	64-bit (x86)
ubuntu Verified provider	Deep Learning AMI Neuron PyTorch 1.13 (Ubuntu 20.04) 20231003 ami-0f337e1c69255b2b6 (64-bit (x86)) Supported EC2 instances: Inf1, Trn1, Trn1n, Inf2. Release notes: https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html	Select	64-bit (x86)

Wenn Sie DLAMIs mit CLI starten, müssen Sie Ihre Abfragen ändern. Beispielsweise:

Derzeit wird die folgende CLI-Abfrage für Basis-DLAMIs verwendet, die alle Instanzen unterstützen [P3, P3dn, G3, G4dn, G5, P4, P5]:

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

Die neuen CLI-Abfragen lauten:

Für Basis-DLAMI, das P3, P3dn und G3 unterstützt:

```
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
```

Für Basis-DLAMI, das G4dn, G5, P4 und P5 unterstützt:

```
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
```

[Die aktualisierten Versionshinweise für neue AMIs finden Sie hier.](#) Informationen zum Starten von AMIs auf EC2-Instances finden Sie [hier](#).

Wann sollten Sie mit der Verwendung der neuen DLAMIs beginnen?

Sie sollten so schnell wie möglich damit beginnen, die neuen DLAMIs für die neuesten Frameworks, Abhängigkeiten, Patches und Funktionen zu verwenden. [Wenn Sie Amazon Linux 2-DLAMIs verwenden, die vor dem 08.11.2023 veröffentlicht wurden, können Sie optional wählen, ihre DLAMIs bis zum 30.11.2023 weiterhin live zu patchen \(siehe Anweisungen hier\).](#)

Wird es bei den neuen DLAMIs zu Funktionseinbußen kommen?

Nein, bei den neuen DLAMIs gibt es keinen Funktionsverlust. Die neuen DLAMIs bieten nach der Aufteilung weiterhin die gesamte Funktionalität, Leistung und Sicherheit der alten DLAMIs vor der Aufteilung, sofern sie auf einer kompatiblen Instanz ausgeführt werden. Wir teilen die DLAMIs in zwei Gruppen auf, sodass wir Ihnen weiterhin aktuelle, performante und sichere DLAMIs für den Einsatz auf einer Vielzahl von Instanzen anbieten können.

Was ist mit DLCs?

DLCs enthalten den NVIDIA-Treiber nicht, sodass sie von dieser Änderung nicht betroffen sind. Sie sollten jedoch sicherstellen, dass die DLCs auf AMIs ausgeführt werden, die mit den zugrunde liegenden Instances kompatibel sind.

Verwandte Informationen

Themen

- [Foren](#)
- [Verwandte -Blog-Beiträge](#)
- [Häufig gestellte Fragen](#)

Foren

- [Forum:AWS Deep-Learning-AMIs](#)

Verwandte -Blog-Beiträge

- [Aktualisierte Liste von Artikeln zu Deep Learning-AMIs](#)
- [Starte einenAWS Deep Learning AMI \(in 10 Minuten\)](#)
- [Schnelleres Training mit Optimized TensorFlow 1.6 auf Amazon EC2 C5- und P3-Instances](#)
- [NeueAWS Deep-Learning-AMIs für Praktiker des Machine Learning](#)
- [Neue Schulungen verfügbar: Einführung in Machine Learning und Deep Learning amAWS](#)
- [Reise ins Deep Learning mitAWS](#)

Häufig gestellte Fragen

- F: Wie behalte ich den Überblick über Produktankündigungen im Zusammenhang mit DLAMI?

Hier sind zwei Vorschläge dazu:

- Setzen Sie ein Lesezeichen für diese Blog-Kategorie „AWSDeep Learning AMIs“, die Sie hier finden: [Aktualisierte Liste von Artikeln zu Deep Learning-AMIs](#).
- Schauen Sie sich das [Forum an:AWS Deep Learning-AMIs](#)
- F: Sind die NVIDIA-Treiber und CUDA installiert?

Ja. Einige DLAMIs haben unterschiedliche Versionen. Der [Deep-Learning-AMI mit Conda](#) hat die neuesten Versionen aller DLAMI. Dies wird detaillierter in [CUDA-Installationen und Framework-](#)

[Bindungen](#) beschrieben. Sie können auch in den Versionshinweisen des jeweiligen AMI nachlesen, um zu überprüfen, was installiert ist.

- F: Ist cuDNN installiert?

Ja.

- F: Wie stelle ich fest, dass die GPUs erkannt werden und sehe ihren aktuellen Status?

Führen Sie `nvidia-smi`. Dies zeigt, abhängig vom Instance-Typ, eine oder mehrere GPUs und ihren aktuellen Speicherverbrauch an.

- F: Sind virtuelle Umgebungen für mich eingerichtet?

Ja, aber nur im [Deep-Learning-AMI mit Conda](#).

- F: Welche Version von Python ist installiert?

Jedes DLAMI hat sowohl Python 2 als auch 3. Jedes [Deep-Learning-AMI mit Conda](#) umfasst Umgebungen für beide Versionen für alle Frameworks.

- F: Ist Keras installiert?

Das hängt vom AMI ab. Das [Deep-Learning-AMI mit Conda](#) stellt Keras als Frontend für jedes Framework zur Verfügung. Die Version von Keras hängt von der Unterstützung des Frameworks ab.

- F: Ist es kostenlos?

Alle DLAMIs sind kostenlos. Je nachdem, welchen Instance-Typ Sie wählen, kann die Instance jedoch kostenpflichtig sein. Weitere Informationen finden Sie unter [Preise für das DLAMI](#).

- F: Ich erhalte CUDA-Fehler oder GPU-bezogene Meldungen von meinem Framework. Was ist los?

Überprüfen Sie, welchen Instance-Typ Sie verwendet haben. Für viele Beispiele und Tutorials ist eine GPU erforderlich. Wenn beim Ausführen keine GPU `nvidia-smi` angezeigt wird, müssen Sie einen weiteren DLAMI starten, indem Sie eine Instanz mit einer oder mehreren GPUs verwenden. Weitere Informationen finden Sie unter [Auswahl des Instanztyps für DLAMI](#).

- F: Kann ich Docker verwenden?

Docker ist seit Version 14 des Deep Learning AMI mit Conda vorinstalliert. Beachten Sie, dass Sie [Nvidia-Docker](#) auf GPU-Instanzen verwenden sollten, um die GPU nutzen zu können.

- F: In welchen Regionen sind Linux-DLAMIs verfügbar?

Region	Code
USA Ost (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
GovCloud	us-gov-west-1
USA West (Nordkalifornien)	us-west-1
US West (Oregon)	us-west-2
Peking (China)	cn-north-1
Ningxia (China)	cn-northwest-1
Asien-Pazifik (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
EU (Frankfurt)	eu-central-1
EU (Irland)	eu-west-1
EU (London)	eu-west-2
EU (Paris)	eu-west-3
SA (São Paulo)	sa-east-1

- F: In welchen Regionen sind Windows-DLAMIs verfügbar?

Region	Code
USA Ost (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
GovCloud	us-gov-west-1
USA West (Nordkalifornien)	us-west-1
US West (Oregon)	us-west-2
Peking (China)	cn-north-1
Asien-Pazifik (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
EU (Frankfurt)	eu-central-1
EU (Irland)	eu-west-1
EU (London)	eu-west-2
EU (Paris)	eu-west-3
SA (São Paulo)	sa-east-1

Versionshinweise für DLAMI

Note

AWS Deep Learning AMI Wir haben einen nächtlichen Veröffentlichungsrhythmus für Sicherheitspatches. Diese inkrementellen Sicherheitspatches sind nicht in den offiziellen Versionshinweisen enthalten.

Versionshinweise zu nicht unterstützten Frameworks finden Sie auf der [Seite mit den DLAMI-Supportrichtlinien](#).

Basis DLAMI

GPU

- X86
 - [AWS Deep Learning-Basis-AMI \(Amazon Linux 2\)](#)
 - [AWS Deep-Learning-Basis-AMI \(Ubuntu 22.04\)](#)
 - [AWS Deep-Learning-Basis-AMI \(Ubuntu 20.04\)](#)
- ARM64
 - [AWS Deep Learning Base ARM64-AMI \(Ubuntu 22.04\)](#)
 - [AWS Deep Learning Base ARM64-AMI \(Amazon Linux 2\)](#)

AWS Neuron

- X86
 - [AWS Deep-Learning-Basis AMI Neuron \(Amazon Linux 2\)](#)
 - [AWS Deep-Learning-Basis AMI Neuron \(Ubuntu 20.04\)](#)

Qualcomm

- X86
 - [AWS Deep-Learning-Basis Qualcomm AMI \(Amazon Linux 2\)](#)

DLAMI mit einem einzigen Framework

PyTorch-Spezifisches AMI

GPU

- X86
 - [AWS Deep-Learning-AMI-GPU PyTorch 2.3 \(Ubuntu 20.04\)](#)
 - [AWS Deep-Learning-AMI-GPU PyTorch 2.3 \(Amazon Linux 2\)](#)
 - [AWS Deep-Learning-AMI-GPU PyTorch 2.2 \(Ubuntu 20.04\)](#)
 - [AWS Deep-Learning-AMI-GPU PyTorch 2.2 \(Amazon Linux 2\)](#)
 - [AWS Deep-Learning-AMI-GPU PyTorch 1.13 \(Amazon Linux 2\)](#)
 - [AWS Deep-Learning-AMI-GPU PyTorch 1.13 \(Ubuntu 20.04\)](#)
- ARM64
 - [AWS ARM64 AMI GPU PyTorch 2.3 für Deep Learning \(Ubuntu 22.04\)](#)
 - [AWS ARM64 AMI GPU PyTorch 2.2 für Deep Learning \(Ubuntu 20.04\)](#)

AWS Neuron

- X86
 - [AWS Deep-Learning-AMI Neuron PyTorch 1.13 \(Amazon Linux 2\)](#)
 - [AWS Deep-Learning-AMI Neuron PyTorch 1.13 \(Ubuntu 20.04\)](#)

TensorFlow-Spezifisches AMI

GPU

- X86
 - [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 Neuron

- X86
 - [AWS Deep-Learning-AMI Neuron TensorFlow 2.10 \(Amazon Linux 2\)](#)
 - [AWS Deep-Learning-AMI Neuron TensorFlow 2.10 \(Ubuntu 20.04\)](#)

DLAMI mit mehreren Frameworks

Note

Wenn Sie nur ein Framework für maschinelles Lernen verwenden, empfehlen wir ein [DLAMI mit einem einzigen Framework](#)

GPU

- X86
 - [AWS Deep-Learning-AMI \(Amazon Linux 2\)](#)

AWS Neuron

- X86
 - [AWS Deep-Learning-AMI-Neuron \(Ubuntu 22.04\)](#)

DLAMI bis zur Veraltung bis zur Veraltung bis

In der folgenden Tabelle werden Informationen zu veralteten Funktionen im AWS Deep Learning AMI aufgeführt.

Veraltete Funktion	Datum der Veraltung	Hinweis zur Veraltung
Ubuntu 16.04	10.07.2021	Ubuntu Linux 16.04 LTS erreichte am 30. April 2021 das Ende seines fünfjährigen LTS-Fensters 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-life ab Dezember 2020 verfügbar. Seit Oktober 2021 gibt es in neuen Versionen keine Updates mehr für das Deep Learning AMI (Amazon Linux). Frühere ab Version (Amazon Linux) ab Version 1 (Amazon Linux) bis zur Veraltung bis zur Veraltung bis zur Veraltung bis zur Veraltung
Chainer	01.07.2020	Chainer hat das Ende der Hauptversionen ab

Veraltete Funktion	Datum der Veraltung	Hinweis zur Veraltung
		<p>Dezember 2019 angekündigt. Folglich werden wir ab Juli 2020 keine Chainer Conda-Umgebungen mehr in den DLAMI aufnehmen . Frühere Versionen des DLAMI, die diese Umgebungen enthalten, werden weiterhin verfügbar sein. Wir werden Updates für diese Umgebungen nur bereitstellen, wenn von der Open-Source-Community Sicherheitskorrekturen für diese Frameworks veröffentlicht werden.</p>
Python 3.6	15.06.2020	Aufgrund von Kundenwünschen wechseln wir für neue TF/MX/PT-Versionen zu Python 3.7.

Veraltete Funktion	Datum der Veraltung	Hinweis zur Veraltung
Python 2	01.01.2020	<p>Die Python-Open-Source-Community hat die Unterstützung für Python 2 offiziell beendet.</p> <p>Die TensorFlow PyTorch, und MXNet-Communities haben außerdem angekündigt, dass die Versionen TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4 und MXNet 1.6.0 die letzten Versionen sein werden, die Python 2 unterstützen.</p>

Leitfaden zur Dokumentenhistorie für AWS Deep Learning AMI Entwickler

Änderung	Beschreibung	Datum
ARM64 DLAMI	Das unterstützt AWS Deep Learning AMI jetzt Bilder auf GPUs mit Arm64-Prozessoren.	29. November 2021
TensorFlow 2	Das Deep Learning AMI mit Conda enthält jetzt TensorFlow 2 mit CUDA 10.	3. Dezember 2019
AWS Inferenz	Das Deep Learning AMI unterstützt jetzt AWS Inferentia-Hardware und das AWS Neuron SDK.	3. Dezember 2019
TensorFlow Serving mit einem Inception-Modell verwenden	Ein Beispiel für die Verwendung von Inferenz mit einem Inception-Modell wurde für TensorFlow Serving hinzugefügt, sowohl für mit als auch ohne Elastic Inference.	28. November 2018
Elastic Inference	Die Voraussetzungen für Elastic Inference sowie zugehörige Informationen wurden zur Einrichtungsanleitung hinzugefügt.	28. November 2018
Installation PyTorch über einen Nightly-Build	Es wurde ein Tutorial hinzugefügt, das erklärt PyTorch, wie Sie mit Conda einen nächtlichen Build von PyTorch auf Ihrem Deep Learning-AMI deinstallieren	25. September 2018

und anschließend installieren können.

[Docker ist jetzt auf Ihrem DLAMI vorinstalliert](#)

Seit Version 14 des Deep Learning AMI mit Conda, Docker und NVIDIAS Version von Docker für GPUs ist vorinstalliert.

25. September 2018

[Conda-Tutorial](#)

Die Beispiel-MOTD wurde auf eine neuere Version aktualisiert.

23. Juli 2018

Frühere Aktualisierungen:

In der folgenden Tabelle werden wichtige Änderungen in den einzelnen Versionen AWS Deep Learning AMI vor Juli 2018 beschrieben.

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

Änderung	Beschreibung	Datum
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
Linux-AMIs v3.0 sowie Einführung von MXNet Model Server, TensorFlow Serving und TensorBoard	Es wurden Tutorials für Conda-AMIs mit neuen Modell- und Visualisierungsfunktionen 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/releases/notes/ verschoben	25. Januar 2018
Linux-AMIs v2.0	Basis-, Quell- 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

Änderung	Beschreibung	Datum
Zwei Windows-AMI-Optionen hinzugefügt	Windows 2012 R2- und Windows 2016-AMIs veröffentlicht: AMI-Auswahlhilfe ergänzt und Release-Informationen hinzugefügt.	30. November 2017
Erste Dokumentationsversion	Detaillierte Beschreibung der Änderung mit Link zum geänderten Thema/Abschnitt.	15. November 2017

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im [AWS-Glossar](#) in der AWS-Glossar-Referenz.

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.