



Entwicklerhandbuch

# AWS Deep Learning AMIs



# AWS Deep Learning AMIs: 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 die DLAMI? .....	1
Informationen zu diesem Handbuch .....	1
Voraussetzungen .....	1
Beispielanwendungsfälle .....	1
Features .....	2
Vorinstallierte Frameworks .....	2
GPUVorinstallierte Software .....	3
Bereitstellung und Visualisierung von Modellen .....	3
Versionshinweise für DLAMIs .....	4
Basis-DLAMIs .....	4
Ein einziges Framework DLAMIs .....	5
Multi-Framework DLAMIs .....	6
Erste Schritte .....	7
Auswahl eines DLAMI .....	7
CUDA-Installationen und Framework-Bindungen .....	8
Basis .....	9
Conda .....	10
Architektur .....	11
BS .....	11
Eine Instanz auswählen .....	12
Preisgestaltung .....	13
Verfügbarkeit in Regionen .....	14
GPU .....	14
CPU .....	15
Inferentia .....	16
Trainium .....	16
Einrichtung .....	18
Eine DLAMI ID finden .....	18
Starten einer -Instance .....	20
Herstellen einer Verbindung mit einer Instance .....	22
Jupyter einrichten .....	22
Server sichern .....	23
Server wird gestartet .....	24
Client wird verbunden .....	25

---

Einloggen .....	26
Bereinigen .....	28
Verwenden eines DLAMI .....	30
Conda DLAMI .....	30
Einführung in das Deep Learning AMI mit Conda .....	30
Loggen Sie sich in Ihr DLAMI .....	31
Starten Sie die TensorFlow Umgebung .....	31
Wechseln Sie zur PyTorch Python-3-Umgebung .....	32
Entfernen von Umgebungen .....	33
Basis-DLAMI .....	33
Verwendung der Deep Learning Base AMI .....	33
CUDAVersionen konfigurieren .....	34
Jupyter Notebooks .....	34
Navigation der installierten Tutorials .....	35
Wechseln von Umgebungen mit Jupyter .....	36
Tutorials .....	36
Aktivieren von Frameworks .....	37
Elastic Fabric Adapter .....	40
GPUÜberwachung und Optimierung .....	55
AWS Inferentia .....	66
ARM64 DLAMI .....	88
Inferenz .....	91
Modellbereitstellung .....	92
Upgraden Ihres DLAMI .....	98
DLAMIEin Upgrade durchführen .....	98
Software-Updates .....	99
Benachrichtigungen veröffentlichen .....	100
Sicherheit .....	102
Datenschutz .....	103
Identity and Access Management .....	104
Authentifizierung mit Identitäten .....	104
Verwalten des Zugriffs mit Richtlinien .....	108
IAMmit Amazon EMR .....	111
Compliance-Validierung .....	111
Ausfallsicherheit .....	112
Sicherheit der Infrastruktur .....	112

Überwachen .....	112
Nachverfolgung der Nutzung .....	113
Rahmenunterstützungspolitik .....	114
DLAMIFramework-Unterstützung FAQs .....	114
Welche Framework-Versionen erhalten Sicherheitspatches? .....	115
Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden? .....	115
Welche Bilder erhalten neue SageMaker AWS KI/Funktionen? .....	115
Wie ist die aktuelle Version in der Tabelle Unterstützte Frameworks definiert? .....	115
Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Frameworks aufgeführt ist? .....	116
Werden frühere Versionen von DLAMIs unterstützt TensorFlow? .....	116
Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version? .....	116
Wie oft werden neue Images veröffentlicht? .....	116
Wird meine Instance gepatcht, während mein Workload läuft? .....	117
Was passiert, wenn eine neue gepatchte oder aktualisierte Framework-Version verfügbar ist? .....	117
Werden Abhängigkeiten aktualisiert, ohne die Framework-Version zu ändern? .....	117
Wann endet der aktive Support für meine Framework-Version? .....	117
Werden Images mit Framework-Versionen, die nicht mehr aktiv verwaltet werden, gepatcht? .....	119
Wie verwende ich eine ältere Framework-Version? .....	119
Wie bleibe ich auf up-to-date dem Laufenden, wenn Änderungen an Frameworks und ihren Versionen unterstützt werden? .....	119
Benötige ich eine kommerzielle Lizenz, um das Anaconda Repository nutzen zu können? ..	119
Wichtige Änderungen .....	120
DLAMINVIDIATreiberwechsel FAQs .....	120
Was hat sich geändert? .....	120
Warum war diese Änderung erforderlich? .....	121
DLAMIsWorauf hat sich diese Änderung ausgewirkt? .....	122
Was bedeutet das für dich? .....	122
Gibt es bei der neueren Version einen Verlust an Funktionalität? DLAMIs .....	122
Hatte diese Änderung Auswirkungen auf Deep Learning Containers? .....	123
Ähnliche Informationen .....	124
Veraltete Funktionen .....	125
Dokumentverlauf .....	127

---

..... CXXX

# Was ist AWS Deep Learning AMIs?

AWS Deep Learning AMIs (DLAMI) bietet maßgeschneiderte Maschinenbilder, die Sie für Deep Learning in der Cloud verwenden können. Sie DLAMIs sind in den meisten Versionen AWS-Regionen für eine Vielzahl von Amazon Elastic Compute Cloud (AmazonEC2) CPU -Instance-Typen verfügbar, von kleinen reinen Instances bis hin zu den neuesten Hochleistungs-Multi-Instances. GPU DLAMIs sind mit [NVIDIA CUDADNN](#) und [NVIDIA Cu](#) sowie den neuesten Versionen der beliebtesten Deep-Learning-Frameworks vorkonfiguriert.

## Informationen zu diesem Handbuch

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

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

## Voraussetzungen

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

## Beispielanwendungsfälle für DLAMI

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

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

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

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

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

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

#### Note

Ihre erste Wahl könnte darin bestehen, Ihren Instance-Typ auf eine größere Instance mit mehr GPUs (bis zu 8) aufzurüsten, Sie können aber auch horizontal skalieren, indem Sie einen Cluster von DLAMI Instances erstellen. Weitere Informationen zu Cluster-Builds finden Sie in den entsprechenden [Verwandte Informationen über DLAMI](#).

## Features von DLAMI

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

## Vorinstallierte Frameworks

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



- [Deep Learning AMI mit Conda](#)— Frameworks werden separat mithilfe von conda Paketen und separaten Python-Umgebungen installiert.
- [Deep Learning Learning Learning Learning Learning Learning Learning Learning](#)— Keine Frameworks installiert; nur [NVIDIA CUDA](#) und andere Abhängigkeiten.

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

- PyTorch
- TensorFlow 2

#### Note

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

## GPU Vorinstallierte Software

[Selbst wenn Sie CPU nur eine Instanz verwenden, DLAMIs wird sie haben NVIDIA CUDA und NVIDIA cu. DNN](#) Die installierte Software ist unabhängig vom Instance-Typ identisch. Beachten Sie, dass GPU -spezifische Tools nur auf einer Instanz funktionieren, die über mindestens eine Instanz verfügt. GPU Weitere Informationen zu Instanztypen finden Sie unter [Auswahl eines DLAMI Instanztyps](#).

Mehr über CUDA erfahren Sie unter [CUDA-Installationen und Framework-Bindungen](#).

## Bereitstellung und Visualisierung von Modellen

In Deep Learning AMI mit Conda sind Modellserver sowohl für Modellvisualisierungen als auch TensorBoard für TensorFlow Modellvisualisierungen vorinstalliert. Weitere Informationen finden Sie unter [TensorFlow Servieren](#).

# Versionshinweise für DLAMIs

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

Versionshinweise für DLAMI Frameworks, die wir nicht mehr Support, finden Sie im Abschnitt Archiv der Versionshinweise zu nicht unterstütztem Framework auf der Seite [DLAMIFramework-Supportrichtlinien](#).

## Note

AWS Deep Learning AMIs Sie haben einen nächtlichen Veröffentlichungsrhythmus für Sicherheitspatches. Wir nehmen diese inkrementellen Sicherheitspatches nicht in den offiziellen Versionshinweisen auf.

## Basis-DLAMIs

### GPU

- X86
  - [AWS Deep Learning-Basis AMI \(Amazon Linux 2023\)](#)
  - [AWS Deep Learning-Basis AMI \(Ubuntu 22.04\)](#)
  - [AWS Deep-Learning-Basis AMI \(Ubuntu 20.04\)](#)
  - [AWS Deep Learning-Basis AMI \(Amazon Linux 2\)](#)
- ARM64
  - [AWS Deep Learning-Basis ARM64 AMI \(Ubuntu 22.04\)](#)
  - [AWS Deep Learning-Basis ARM64 AMI \(Amazon Linux 2\)](#)

### AWS Neuron

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

## Qualcomm

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

## Ein einziges Framework DLAMIs

### PyTorch-spezifisch AMIs

#### GPU

- X86
  - [AWS Tiefes Lernen AMI GPU PyTorch 2.5 \(Ubuntu 22.04\)](#)
  - [AWS Deep Learning AMI GPU PyTorch 2.5 \(Amazon Linux 2023\)](#)
  - [AWS Tiefes Lernen AMI GPU PyTorch 2.4 \(Ubuntu 22.04\)](#)
  - [AWS Tiefes Lernen AMI GPU PyTorch 2.3 \(Ubuntu 20.04\)](#)
  - [AWS Deep Learning AMI GPU PyTorch 2.3 \(Amazon Linux 2\)](#)
  - [AWS Tiefes Lernen AMI GPU PyTorch 2.2 \(Ubuntu 20.04\)](#)
  - [AWS Deep Learning AMI GPU PyTorch 2.2 \(Amazon Linux 2\)](#)
- ARM64
  - [AWS Tiefes Lernen ARM64 AMI GPU PyTorch 2.5 \(Ubuntu 22.04\)](#)
  - [AWS Tiefes Lernen ARM64 AMI GPU PyTorch 2.4 \(Ubuntu 22.04\)](#)
  - [AWS Tiefes Lernen ARM64 AMI GPU PyTorch 2.3 \(Ubuntu 22.04\)](#)
  - [AWS Tiefes Lernen ARM64 AMI GPU PyTorch 2.2 \(Ubuntu 20.04\)](#)

### AWS Neuron

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

### TensorFlow-spezifisch AMIs

- X86
  - [AWS Deep Learning AMI GPU TensorFlow 2.18 \(Amazon Linux 2023\)](#)
  - [AWS Tiefes Lernen AMI GPU TensorFlow 2.18 \(Ubuntu 22.04\)](#)
  - [AWS Tiefes Lernen AMI GPU TensorFlow 2.17 \(Ubuntu 22.04\)](#)
  - [AWS Deep Learning AMI GPU TensorFlow 2.16 \(Amazon Linux 2\)](#)
  - [AWS Tiefes Lernen AMI GPU TensorFlow 2.16 \(Ubuntu 20.04\)](#)

## AWS Neuron

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

## Multi-Framework DLAMIs

### Tip

Wenn Sie nur ein Framework für maschinelles Lernen verwenden, empfehlen wir ein [einzelnes DLAMI](#) Framework.

## GPU

- X86
  - [AWS Tiefes Lernen AMI \(Amazon Linux 2\)](#)

## AWS Neuron

- X86
  - [AWS AMIDeep-Learning-Neuron \(Amazon Linux 2023\)](#)
  - [AWS AMIDeep-Learning-Neuron \(Ubuntu 22.04\)](#)

# Erste Schritte mit DLAMI

Dieser Leitfaden enthält Tipps zur Auswahl des DLAMI für Sie passenden Instance-Typs, zur Auswahl eines Instance-Typs, der zu Ihrem Anwendungsfall und Budget passt, und [Verwandte Informationen über DLAMI](#) beschreibt benutzerdefinierte Setups, die für Sie von Interesse sein könnten.

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

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

Nächster Schritt

[Wählen Sie ein DLAMI](#)

## Wählen Sie ein DLAMI

Wir bieten eine Reihe von DLAMI Optionen, wie in den [GPU DLAMI Versionshinweisen](#) erwähnt. Um Ihnen bei der Auswahl der richtigen Lösung 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:

- DLAMITyp: Base, Single-Framework, Multi-Framework (Conda) DLAMI
- Rechenarchitektur: [x86-basiertes, ARM64-basiertes Graviton AWS](#)
- Prozessortyp [:,, Inferentia, Trainium GPU CPU](#)
- SDK [CUDA AWS](#):, Neuron
- Betriebssystem: Amazon Linux, Ubuntu

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

## Themen

- [CUDA-Installationen und Framework-Bindungen](#)
- [Deep Learning Learning Learning Learning Learning Learning Learning Learning](#)
- [Deep Learning AMI mit Conda](#)
- [DLAMI-Architekturoptionen](#)
- [DLAMI-Betriebssystem-Optionen](#)

## 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 AMIs

Spezifische Framework-Versionsnummern finden Sie in der [Versionshinweise für DLAMIs](#)

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 der 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

Die Conda DLAMI verwendet conda virtuelle Umgebungen, sie sind entweder in mehreren Frameworks oder in einem einzigen Framework verfügbar. DLAMIs Diese Umgebungen sind so konfiguriert, dass die verschiedenen Framework-Installationen getrennt bleiben und der Wechsel zwischen Frameworks rationalisiert wird. Dies ist perfekt zum Lernen und Experimentieren mit den Frameworks des DLAMIs geeignet. Die meisten Benutzer sind der Meinung, dass das neue Deep Learning AMI mit Conda perfekt für sie ist.

Sie werden häufig mit den neuesten Versionen der Frameworks aktualisiert und verfügen über die neuesten GPU Treiber und Software. Sie werden in den meisten Dokumenten allgemein als „die AWS Deep Learning AMIs“ bezeichnet. Diese DLAMIs unterstützen die Betriebssysteme Ubuntu 20.04, Ubuntu 22.04, Amazon Linux 2 und Amazon Linux 2023. Die Unterstützung von Betriebssystemen hängt von der Unterstützung durch das vorgelagerte Betriebssystem ab.

### Stable versus Release-Kandidaten

Die Conda AMIs verwenden optimierte Binärdateien der neuesten formellen Versionen aus jedem Framework. Release-Kandidaten und experimentelle Funktionen sind nicht zu erwarten. Die Optimierungen hängen davon ab, dass das Framework Beschleunigungstechnologien wie die von Intel unterstützt MKLDNN, wodurch das Training und die Inferenz für C5- und C4-Instance-Typen beschleunigt werden. CPU Die Binärdateien wurden außerdem so kompiliert, dass sie erweiterte Intel-Befehlssätze unterstützen, einschließlich, aber nicht beschränkt auf AVX, AVX -2, SSE4 .1 und .2. SSE4 Diese beschleunigen Vektor- und Fließkommaoperationen auf CPU Intel-Architekturen. Darüber hinaus DNN werden beispielsweise die GPU Instance-Typen the CUDA und cu mit der Version aktualisiert, die von der neuesten offiziellen Version unterstützt wird.

Deep Learning AMI mit Conda installiert bei der ersten Aktivierung des Frameworks automatisch die optimierteste Version des Frameworks für Ihre EC2 Amazon-Instance. Weitere Informationen finden Sie unter [Deep Learning mit Conda verwenden AMI](#).

Wenn Sie aus dem Quellcode installieren und dabei benutzerdefinierte oder optimierte Build-Optionen verwenden möchten, [Deep Learning Learning 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 von DLAMI (v26, v25 usw.), die Python 2 Conda-Umgebungen enthalten, sind weiterhin verfügbar. Wir stellen jedoch Updates für die Python 2 Conda-Umgebungen für zuvor veröffentlichte DLAMI Versionen nur bereit, wenn von der Open-Source-Community Sicherheitsupdates für diese Versionen veröffentlicht wurden. DLAMIReleases mit den neuesten Versionen der PyTorch Frameworks TensorFlow und enthalten nicht die Python 2 Conda-Umgebungen.

## CUDA-Unterstützung

Spezifische CUDA Versionsnummern finden Sie in den [GPU DLAMI Versionshinweisen](#).

Nächstes Thema

[DLAMI-Architekturoptionen](#)

## Verwandte Themen

- Ein Tutorial zur Verwendung von Deep Learning AMI mit Conda finden Sie im [Deep Learning mit Conda verwenden AMI](#) 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 Tool ARM64 DLAMI](#) Weitere Informationen zu verfügbaren Instance-Typen finden Sie unter. [Auswahl eines DLAMI Instanztyps](#)

Nächstes Thema

[DLAMI-Betriebssystem-Optionen](#)

## DLAMI-Betriebssystem-Optionen

DLAMIs werden in den folgenden Betriebssystemen angeboten.

- Amazon Linux 2
- Amazon Linux 2023
- Ubuntu 20.04

- Ubuntu 22.04

Ältere Versionen von Betriebssystemen sind als veraltet DLAMIs verfügbar. [Weitere Informationen zu veralteten Versionen finden Sie unter DLAMI Veraltete Versionen für DLAMI](#)

Bevor Sie sich für eine entscheiden DLAMI, sollten Sie abwägen, welchen Instance-Typ Sie benötigen, und Ihre Region identifizieren. AWS

Nächstes Thema

[Auswahl eines DLAMI Instanztyps](#)

## Auswahl eines DLAMI Instanztyps

Generell sollten Sie bei der Auswahl eines Instance-Typs für eine Folgendes berücksichtigen DLAMI.

- Wenn Sie mit Deep Learning noch nicht vertraut sind, GPU könnte eine Instanz mit einer einzigen Instanz Ihren Anforderungen entsprechen.
- Wenn Sie sich Ihres Budgets bewusst sind, können Sie Instances verwenden, die CPU nur aus einer Version bestehen.
- Wenn Sie die hohe Leistung und Kosteneffizienz für die Deep-Learning-Modellinferenz optimieren möchten, können Sie Instances mit AWS Inferentia-Chips verwenden.
- Wenn Sie nach einer GPU Hochleistungsinstanz mit einer ARM64-basierten CPU Architektur suchen, können Sie den Instance-Typ G5g verwenden.
- Wenn Sie daran interessiert sind, ein vortrainiertes Modell für Inferenz und Vorhersagen auszuführen, können Sie Ihrer Amazon-Instance eine [Amazon Elastic Inference](#) hinzufügen. EC2 Amazon Elastic Inference bietet Ihnen Zugriff auf einen Accelerator mit einem Bruchteil von einem 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 verteilten GPUs. Möglicherweise stellt auch eine Instance mit weniger Speicher eine bessere Lösung für Sie dar, wenn Sie Ihre Batchgröße verringern. Dies kann sich auf die Genauigkeit und Schulungsgeschwindigkeit auswirken.
- Wenn Sie daran interessiert sind, Anwendungen für maschinelles Lernen mithilfe der NVIDIA Collective Communications Library (NCCL) auszuführen, die ein hohes Maß an Kommunikation

zwischen den Knoten in großem Umfang erfordern, sollten Sie [Elastic Fabric Adapter \(EFA\)](#) verwenden.

Weitere Informationen zu Instances finden Sie unter .

Die folgenden Themen enthalten Informationen zu Überlegungen zu Instanztypen.

#### Important

Deep Learning AMIs umfasst Treiber, Software oder Toolkits, die von der Corporation entwickelt wurden, im Eigentum von NVIDIA Corporation stehen oder von ihr bereitgestellt werden. Sie erklären sich damit einverstanden, diese NVIDIA Treiber, Software oder Toolkits nur auf EC2 Amazon-Instances zu verwenden, die NVIDIA Hardware enthalten.

## Themen

- [Die Preise für DLAMI](#)
- [DLAMI-Verfügbarkeit in Regionen](#)
- [Empfohlene Instanzen GPU](#)
- [Empfohlene Instanzen CPU](#)
- [Empfohlene Inferenzinstanzen](#)
- [Empfohlene Trainium-Instances](#)

## Die Preise für DLAMI

Die in der enthaltenen Deep-Learning-Frameworks DLAMI sind kostenlos und verfügen jeweils über eigene Open-Source-Lizenzen. Obwohl die in der enthaltene Software kostenlos DLAMI ist, müssen Sie dennoch für die zugrunde liegende EC2 Amazon-Instance-Hardware bezahlen.

Einige EC2 Amazon-Instance-Typen sind als kostenlos gekennzeichnet. Es ist möglich, das DLAMI auf einer dieser kostenlosen Instances auszuführen. Das bedeutet, dass die Nutzung von völlig kostenlos DLAMI ist, wenn Sie nur die Kapazität dieser Instanz nutzen. Wenn Sie eine leistungsfähigere Instanz mit mehr CPU Kernen, mehr Festplattenspeicher, mehr oder einer oder mehreren benötigenRAM, dann benötigen Sie eine InstanzGPUs, die nicht zur Instance-Klasse mit dem kostenlosen Nutzungskontingent gehört.

Weitere Informationen zur Instance-Auswahl und Preisgestaltung finden Sie unter [EC2Amazon-Preise](#).

## DLAMI-Verfügbarkeit in Regionen

Jede Region unterstützt eine andere Reihe von Instance-Typen, und oft fallen für einen Instance-Typ in verschiedenen Regionen leicht unterschiedliche Kosten an. DLAMIs sind nicht in jeder Region verfügbar, aber es ist möglich, in die Region Ihrer Wahl DLAMIs zu kopieren. Weitere Informationen finden Sie unter [Kopieren einer AMI](#). Beachten Sie die Auswahlliste für Regionen und achten Sie darauf, dass Sie eine Region auswählen, die Ihnen oder Ihren Kunden in der Nähe liegt. Wenn Sie planen, mehr als einen zu verwenden DLAMI und möglicherweise einen Cluster zu erstellen, stellen Sie sicher, dass Sie dieselbe Region für alle Knoten im Cluster verwenden.

Weitere Informationen zu Regionen finden Sie unter — [Amazon EC2 Service Endpoints](#).

Nächstes Thema

### [Empfohlene Instanzen GPU](#)

## Empfohlene Instanzen GPU

Für die meisten Deep-Learning-Zwecke empfehlen wir eine GPU Instanz. Das Trainieren neuer Modelle ist auf einer GPU Instanz schneller als auf einer CPU Instanz. Sie können sublinear skalieren, wenn Sie mehrere GPU Instanzen haben oder wenn Sie verteiltes Training auf viele Instanzen mit verwenden. GPUs

Die folgenden Instance-Typen unterstützen die. DLAMI Informationen zu GPU Instance-Typ-Optionen und deren Verwendung finden Sie unter und wählen Sie Accelerated Computing aus.

### Note

Bei der Auswahl einer Instance sollte die Größe Ihres Modells eine Rolle spielen. Wenn Ihr Modell die Verfügbarkeit einer Instanz überschreitet RAM, wählen Sie einen anderen Instanztyp mit ausreichend Speicher für Ihre Anwendung.

- [Amazon EC2 P5e-Instances](#) haben bis zu 8 NVIDIA Tesla H200. GPUs
- [Amazon EC2 P5-Instances](#) haben bis zu 8 NVIDIA Tesla GPUs H100.
- [Amazon EC2 P4-Instances](#) haben bis zu 8 NVIDIA Tesla GPUs A100.

- [Amazon EC2 P3-Instances](#) haben bis zu 8 NVIDIA Tesla GPUs V100.
- [Amazon EC2 G3-Instances](#) haben bis zu 4 NVIDIA Tesla GPUs M60.
- [Amazon EC2 G4-Instances](#) haben bis zu 4 NVIDIA GPUs T4.
- [Amazon EC2 G5-Instances](#) haben bis zu 8 NVIDIA GPUs A10G.
- [Amazon EC2 G6-Instances](#) haben bis zu 8 NVIDIA GPUs L4.
- [Amazon EC2 G6e-Instances](#) haben bis zu 8 NVIDIA L40S Tensor Core. GPUs
- [Amazon EC2 G5G-Instances verfügen über ARM64-basierte AWS Graviton2-Prozessoren.](#)

DLAMIInstances bieten Tools zur Überwachung und Optimierung Ihrer Prozesse. GPU Weitere Informationen zur Überwachung Ihrer GPU Prozesse finden Sie unter [GPUÜberwachung und Optimierung](#).

Spezifische Tutorials zur Arbeit mit G5g-Instances finden Sie unter [Das Tool ARM64 DLAMI](#).

Nächstes Thema

## [Empfohlene Instanzen CPU](#)

### Empfohlene Instanzen CPU

Ganz gleich, ob Sie ein begrenztes Budget haben, etwas über Deep Learning lernen oder einfach nur einen Prognosedienst betreiben möchten, in CPU dieser Kategorie stehen Ihnen viele erschwingliche Optionen zur Verfügung. Einige Frameworks nutzen die Vorteile von Intel MKLDNN, wodurch das Training und die Inferenz für CPU C5-Instance-Typen (nicht in allen Regionen verfügbar) beschleunigt werden. Informationen zu Instance-Typen finden Sie unter CPU Instance-Typen und EC2. Wählen Sie [dort](#) Compute Optimized aus.

#### Note

Die Größe Ihres Modells sollte ein Faktor bei der Auswahl einer Instanz sein. Wenn Ihr Modell die Verfügbarkeit einer Instanz überschreitetRAM, wählen Sie einen anderen Instanztyp mit ausreichend Speicher für Ihre Anwendung.

- [Amazon EC2 C5-Instances](#) verfügen über bis zu 72 IntelvCPUs. C5-Instances zeichnen sich durch wissenschaftliche Modellierung, Stapelverarbeitung, verteilte Analytik, Hochleistungsrechnen (HPC) sowie Inferenz für maschinelles Lernen und Deep Learning aus.

## Nächstes Thema

### [Empfohlene Inferenzinstanzen](#)

## Empfohlene Inferenzinstanzen

AWS Inferentia-Instances sind so konzipiert, dass sie eine hohe Leistung und Kosteneffizienz für Inferenz-Workloads mit Deep-Learning-Modellen bieten. Insbesondere Inf2-Instance-Typen verwenden AWS Inferentia-Chips und das [AWS Neuron SDK](#), das in beliebte Frameworks für maschinelles Lernen wie und integriert ist. TensorFlow PyTorch

Kunden können Inf2-Instances verwenden, um umfangreiche Inferenzanwendungen für maschinelles Lernen wie Suche, Empfehlungsmaschinen, Computer Vision, Spracherkennung, Verarbeitung natürlicher Sprache, Personalisierung und Betrugserkennung zu den niedrigsten Kosten in der Cloud auszuführen.

### Note

Die Größe Ihres Modells sollte ein Faktor bei der Auswahl einer Instanz sein. Wenn Ihr Modell die Verfügbarkeit einer Instanz überschreitet RAM, wählen Sie einen anderen Instanztyp mit ausreichend Speicher für Ihre Anwendung.

- [Amazon EC2 Inf2-Instances](#) verfügen über bis zu 16 AWS Inferentia-Chips und einen Netzwerkdurchsatz von 100 Gbit/s.

Weitere Informationen zu den ersten Schritten mit AWS Inferentia finden Sie unter DLAMIs [Der AWS Inferentia-Chip mit DLAMI](#)


## Nächstes Thema

### [Empfohlene Trainium-Instances](#)

## Empfohlene Trainium-Instances

AWS Trainium-Instances sind darauf ausgelegt, hohe Leistung und Kosteneffizienz für Inferenz-Workloads mit Deep-Learning-Modellen zu bieten. Insbesondere Trn1-Instance-Typen verwenden AWS Trainium-Chips und [AWS Neuron SDK](#), das in beliebte Frameworks für maschinelles Lernen wie und integriert ist. TensorFlow PyTorch

Kunden können Trn1-Instances verwenden, um umfangreiche Inferenzanwendungen für maschinelles Lernen wie Suche, Empfehlungsmaschinen, Computer Vision, Spracherkennung, Verarbeitung natürlicher Sprache, Personalisierung und Betrugserkennung zu den niedrigsten Kosten in der Cloud auszuführen.

 Note

Die Größe Ihres Modells sollte ein Faktor bei der Auswahl einer Instanz sein. Wenn Ihr Modell die Verfügbarkeit einer Instanz überschreitet RAM, wählen Sie einen anderen Instanztyp mit ausreichend Speicher für Ihre Anwendung.

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

# Eine DLAMI Instanz einrichten

Nachdem Sie [einen DLAMI und einen Amazon Elastic Compute Cloud \(AmazonEC2\) -Instanztyp ausgewählt](#) haben, den Sie verwenden möchten, können Sie Ihre neue DLAMI Instance einrichten.

Wenn Sie noch keinen EC2 Instance-Typ ausgewählt haben, finden Sie weitere Informationen unter [Erste Schritte mit DLAMI](#). DLAMI

## Themen

- [Finden Sie die ID eines DLAMI](#)
- [Eine DLAMI Instanz starten](#)
- [Verbindung zu einer DLAMI Instance herstellen](#)
- [Einen Jupyter Notebook-Server auf einer Instanz einrichten DLAMI](#)
- [Eine Instanz bereinigen DLAMI](#)

## Finden Sie die ID eines DLAMI

Jede DLAMI hat eine eindeutige Kennung (ID). Wenn Sie eine DLAMI Instance über die EC2 Amazon-Konsole starten, können Sie optional die DLAMI ID verwenden, um nach der Instance zu suchen DLAMI, die Sie verwenden möchten. Wenn Sie eine DLAMI Instance mit der AWS Command Line Interface (AWS CLI) starten, ist diese ID erforderlich.

Sie können die ID für das Objekt DLAMI Ihrer Wahl finden, indem Sie einen AWS CLI Befehl für Amazon EC2 oder Parameter Store verwenden, eine Funktion von AWS Systems Manager. Anweisungen zur Installation und Konfiguration von finden [Sie unter Erste Schritte mit dem AWS CLI](#) im AWS Command Line Interface Benutzerhandbuch. AWS CLI

### Using Parameter Store

Um eine DLAMI ID zu finden, verwenden Sie `ssm get-parameter`

Im folgenden [ssm get-parameter](#) Befehl lautet das Format des Parameternamens für die `--name` Option `/aws/service/deeplearning/ami/$architecture/$ami_type/latest/ami-id`. In diesem Namensformat *architecture* kann entweder `x86_64` oder sein `arm64`. Geben Sie das an, *ami\_type* indem Sie den DLAMI Namen verwenden und die Schlüsselwörter „deep“, „learning“ und „ami“ entfernen. AMI Der Name kann in gefunden werden [Versionshinweise für DLAMIs](#).



**⚠ Important**

Um diesen Befehl verwenden zu können, muss der Prinzipal AWS Identity and Access Management (IAM), den Sie verwenden, über die `ssm:GetParameter` entsprechende Berechtigung verfügen. Weitere Informationen zu IAM Prinzipalen finden Sie im Abschnitt [Zusätzliche Ressourcen](#) unter IAMRollen im IAMBenutzerhandbuch.

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

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
ami-09ee1a996ac214ce7
```

**💡 Tip**

Spezifischere `ssm get-parameter` Beispielbefehle für einige derzeit unterstützte DLAMI Frameworks finden Sie unter [Versionshinweise für DLAMIs](#). Wählen Sie den Link zu den Versionshinweisen Ihrer Wahl DLAMI und suchen Sie dann in den Versionshinweisen nach der entsprechenden ID-Abfrage.

## Using Amazon EC2 CLI

Um eine DLAMI ID zu finden, verwenden Sie `ec2 describe-images`

Geben Sie im folgenden [ec2 describe-images](#) Befehl für den Wert des Filters `Name=name` den DLAMI Namen ein. Sie können eine Release-Version für ein bestimmtes Framework angeben oder die neueste Version abrufen, indem Sie die Versionsnummer durch ein Fragezeichen (?) ersetzen.

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

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
ami-09ee1a996ac214ce7
```

#### Tip

Einen `ec2 describe-images` Beispielfehl, der spezifisch für den Befehl DLAMI Ihrer Wahl ist, finden Sie unter [Versionshinweise für DLAMIs](#). Wählen Sie den Link zu den Versionshinweisen Ihrer Wahl aus DLAMI, und suchen Sie dann in den Versionshinweisen nach der zugehörigen ID-Abfrage.

Nächster Schritt

### [Eine DLAMI Instanz starten](#)

## Eine DLAMI Instanz starten

Nachdem Sie [die ID gefunden](#) haben DLAMI, die Sie zum Starten einer DLAMI Instance verwenden möchten, können Sie die Instance starten. Um es zu starten, können Sie entweder die EC2 Amazon-Konsole oder die AWS Command Line Interface (AWS CLI) verwenden.

#### Note

Für diese exemplarische Vorgehensweise verweisen wir möglicherweise speziell auf den Deep Learning Base OSS Nvidia-Treiber GPU AMI (Ubuntu 22.04). Selbst wenn Sie ein anderes DLAMI auswählen, sollten Sie in der Lage sein, diesem Handbuch zu folgen.

EC2 console

#### Note

Um Anwendungen für Hochleistungsrechnen (HPC) und maschinelles Lernen zu beschleunigen, können Sie Ihre DLAMI Instance mit einem Elastic Fabric Adapter (EFA) starten. Spezifische Anweisungen finden Sie unter [Eine AWS Deep Learning AMIs Instance starten mit EFA](#).

1. Öffnen Sie die [EC2Konsole](#).
2. Notieren Sie sich Ihre aktuelle AWS-Region Position in der obersten Navigationsleiste. Wenn dies nicht Ihre gewünschte Region ist, ändern Sie diese Option, bevor Sie fortfahren. Weitere Informationen finden Sie unter der [EC2Amazon-Servicendpunkte](#) in der Allgemeine Amazon Web Services-Referenz.
3. Wählen Sie Launch Instance aus.
4. Geben Sie einen Namen für Ihre Instance ein und wählen Sie DLAMI den für Sie passenden aus.
  - a. Suchen Sie DLAMI in My nach einer vorhandenen AMIs Datei oder wählen Sie Schnellstart.
  - b. Suchen Sie nach DLAMI ID. Durchsuchen Sie die Optionen und wählen Sie dann Ihre Wahl aus.
5. Wählen Sie einen Instance-Type. Sie können die empfohlenen Instance-Familien für Ihr DLAMI In finden [Versionshinweise für DLAMIs](#). Allgemeine Empfehlungen zu DLAMI Instance-Typen finden Sie unter [Auswahl eines DLAMI Instanztyps](#).
6. Wählen Sie Launch Instance aus.

## AWS CLI

- Um die verwenden zu können AWS CLI, benötigen Sie die ID der InstanzDLAMI, die Sie verwenden möchten, den AWS-Region EC2 Instance-Typ und Ihre Sicherheitstoken-Informationen. Anschließend können Sie die Instance mit dem [ec2 run-instances](#) AWS CLI Befehl starten.

Anweisungen zur Installation und Konfiguration von finden [Sie unter Erste Schritte mit dem AWS CLI](#) im AWS Command Line Interface Benutzerhandbuch. AWS CLI Weitere Informationen, einschließlich Befehlsbeispielen, finden Sie unter [Starten, Auflisten und Schließen von EC2 Amazon-Instances für die AWS CLI](#).

Nachdem Sie Ihre Instance entweder über die EC2 Amazon-Konsole oder gestartet haben AWS CLI, warten Sie, bis die Instance bereit ist. Dieser Vorgang dauert einige Minuten. Sie können den Status der Instance in der [EC2Amazon-Konsole](#) überprüfen. Weitere Informationen finden Sie unter [Statuschecks für EC2 Amazon-Instances](#) im EC2Amazon-Benutzerhandbuch.

## Nächster Schritt

### [Verbindung zu einer DLAMI Instance herstellen](#)

## Verbindung zu einer DLAMI Instance herstellen

Nachdem Sie [eine DLAMI Instance gestartet](#) haben und die Instance läuft, können Sie über einen Client (Windows, macOS oder Linux) eine Verbindung zu ihr herstellen SSH. Anweisungen finden Sie SSH im EC2 Amazon-Benutzerhandbuch unter [Connect zu Ihrer Linux-Instance](#) herstellen.

Halten Sie eine Kopie des SSH Login-Befehls bereit, falls Sie nach der Anmeldung einen Jupyter Notebook-Server einrichten möchten. Um eine Verbindung zur Jupyter-Webseite herzustellen, verwenden Sie eine Variante dieses Befehls.

## Nächster Schritt

### [Einen Jupyter Notebook-Server auf einer Instanz einrichten DLAMI](#)

## Einen Jupyter Notebook-Server auf einer Instanz einrichten DLAMI

Mit einem Jupyter Notebook-Server können Sie Jupyter-Notebooks von Ihrer Instanz aus erstellen und ausführen. DLAMI Mit Jupyter-Notebooks können Sie Machine-Learning-Experimente (ML) für Training und Inferenz durchführen und dabei die Infrastruktur nutzen und auf die integrierten Pakete zugreifen. AWS DLAMI Weitere Informationen zu Jupyter-Notebooks finden Sie unter [The Jupyter Notebook auf der Website Jupyter](#) User Documentation.

Um einen Jupyter Notebook-Server einzurichten, müssen Sie:

- Konfigurieren Sie den Jupyter Notebook-Server auf Ihrer Instanz. DLAMI
- Konfigurieren Sie Ihren Client so, dass er eine Verbindung zum Jupyter Notebook-Server herstellt. Wir stellen Konfigurationsanweisungen für Windows-, MacOS- und Linux-Clients zur Verfügung.
- Testen Sie das Setup, indem Sie sich beim Jupyter Notebook-Server anmelden.

Folgen Sie den Anweisungen in den folgenden Themen, um diese Schritte abzuschließen. Nachdem Sie einen Jupyter Notebook-Server eingerichtet haben, können Sie die Beispiel-Notebook-Tutorials ausführen, die im Lieferumfang von enthalten sind. DLAMIs Weitere Informationen finden Sie unter [Ausführen von Jupyter-Notebook-Tutorials](#).

## Themen

- [Den Jupyter Notebook-Server auf einer Instanz sichern DLAMI](#)
- [Den Jupyter Notebook-Server auf einer Instanz starten DLAMI](#)
- [Einen Client mit dem Jupyter Notebook-Server auf einer Instanz verbinden DLAMI](#)
- [Anmeldung beim Jupyter Notebook-Server auf einer Instance DLAMI](#)

## Den Jupyter Notebook-Server auf einer Instanz sichern DLAMI

Um Ihren Jupyter Notebook-Server zu schützen, empfehlen wir, ein Passwort einzurichten und ein Zertifikat für den Server zu erstellen. SSL Um ein Passwort zu konfigurieren und SSL stellen Sie zunächst eine [Verbindung zu Ihrer DLAMI Instanz](#) her und folgen Sie dann diesen Anweisungen.

Um den Jupyter Notebook-Server zu sichern

1. Jupyter bietet ein Passwort-Dienstprogramm. Führen Sie den folgenden Befehl aus und geben Sie Ihr bevorzugtes Passwort ein, wenn Sie dazu aufgefordert werden.

```
$ jupyter notebook password
```

Das Ergebnis sieht etwa folgendermaßen aus:

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

2. Erstellen Sie ein selbstsigniertes Zertifikat. SSL 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 könnten daran interessiert sein, ein reguläres SSL Zertifikat zu erstellen, das von einem Drittanbieter signiert ist und nicht dazu führt, dass der Browser Ihnen eine Sicherheitswarnung ausgibt. Dieser Prozess ist wesentlich aufwendiger. Weitere Informationen finden Sie unter [Sichern eines Notebook-Servers](#) in der Jupyter Notebook-Benutzerdokumentation.

Nächster Schritt

[Den Jupyter Notebook-Server auf einer Instanz starten DLAMI](#)

## Den Jupyter Notebook-Server auf einer Instanz starten DLAMI

Nachdem Sie [Ihren Jupyter Notebook-Server mit einem Passwort gesichert haben](#)SSL, können Sie [den Server starten](#). Melden Sie sich bei Ihrer DLAMI Instanz an und führen Sie den folgenden Befehl aus, der das zuvor SSL erstellte Zertifikat verwendet.

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

Wenn der Server gestartet ist, können Sie jetzt über einen SSH Tunnel von Ihrem Client-Computer aus eine Verbindung zu ihm herstellen. 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 einen lokalen Host auf den Server zugreifen könnenURL, da dies erst funktioniert, wenn Sie den Tunnel erstellt haben.

**Note**

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

Nächster Schritt

[Einen Client mit dem Jupyter Notebook-Server auf einer Instanz verbinden DLAMI](#)

# Einen Client mit dem Jupyter Notebook-Server auf einer Instanz verbinden

## DLAMI

Nachdem Sie [den Jupyter Notebook-Server auf Ihrer DLAMI Instanz gestartet haben, konfigurieren Sie Ihren](#) Windows-, macOS- oder Linux-Client so, dass er eine Verbindung zum Server herstellt. Wenn Sie eine Verbindung herstellen, können Sie Jupyter-Notebooks auf dem Server in Ihrem Workspace erstellen und darauf zugreifen und Ihren Deep-Learning-Code auf dem Server ausführen.

### Voraussetzungen

Stellen Sie sicher, dass Sie über Folgendes verfügen, das Sie für die Einrichtung eines Tunnels benötigen: SSH

- Der öffentliche DNS Name Ihrer EC2 Amazon-Instance. Weitere Informationen finden Sie unter [Hostnamentypen für EC2 Amazon-Instances](#) im EC2Amazon-Benutzerhandbuch.
- Das Schlüsselpaar für die private Schlüsseldatei. Weitere Informationen zum Zugriff auf Ihr key pair finden Sie unter [EC2Amazon-Schlüsselpaare und EC2 Amazon-Instances](#) im EC2Amazon-Benutzerhandbuch.

### Von einem Windows-, macOS- oder Linux-Client aus eine Connect

Um von einem Windows-, macOS- oder Linux-Client aus eine Verbindung zu Ihrer DLAMI Instance herzustellen, folgen Sie den Anweisungen für das Betriebssystem Ihres Clients.

#### Windows

Um von einem Windows-Client aus eine Verbindung zu Ihrer DLAMI Instance herzustellen, verwenden Sie SSH

1. Verwenden Sie einen SSH Client für Windows, z. B. PuTTY. Anweisungen finden Sie TTY im EC2Amazon-Benutzerhandbuch unter Herstellen einer [Connect zu Ihrer Linux-Instance mithilfe von Pu](#). Weitere SSH Verbindungsoptionen finden Sie unter [Stellen Sie eine Verbindung zu Ihrer Linux-Instance her mit SSH](#).
2. (Optional) Erstellen Sie einen SSH Tunnel zu einem laufenden Jupyter-Server. Installieren Sie Git Bash auf Ihrem Windows-Client und folgen Sie dann den Verbindungsanweisungen für macOS- und Linux-Clients.

## macOS or Linux

Um von einem macOS- oder Linux-Client aus eine Verbindung zu Ihrer DLAMI Instance herzustellen, verwenden Sie SSH

1. Öffnen Sie ein -Terminalfenster.
2. Führen Sie den folgenden Befehl aus, um alle Anfragen auf dem lokalen Port 8888 an den Port 8888 auf Ihrer EC2 Amazon-Remoteinstanz weiterzuleiten. Aktualisieren Sie den Befehl, indem Sie den Speicherort Ihres Schlüssels für den Zugriff auf die EC2 Amazon-Instance und den öffentlichen DNS Namen Ihrer EC2 Amazon-Instance ersetzen. Beachten Sie, dass der Benutzername für ein Amazon Linux AMI `ec2-user` anstelle von `ubuntu` lautet.

```
$ ssh -i ~/mykeypair.pem -N -f -L 8888:localhost:8888 ubuntu@ec2-###-##-##-###.compute-1.amazonaws.com
```

Dieser Befehl öffnet einen Tunnel zwischen Ihrem Client und der entfernten EC2 Amazon-Instance, auf der der Jupyter Notebook-Server ausgeführt wird.

## Nächster Schritt

### [Anmeldung beim Jupyter Notebook-Server auf einer Instance DLAMI](#)

## Anmeldung beim Jupyter Notebook-Server auf einer Instance DLAMI

Nachdem Sie [Ihren Client mit dem Jupyter Notebook-Server auf Ihrer DLAMI Instanz verbunden](#) haben, können Sie sich beim Server anmelden.

Um sich in Ihrem Browser beim Server anzumelden

1. Geben Sie in der Adressleiste Ihres Browsers Folgendes URL ein oder klicken Sie auf diesen Link: <https://localhost:8888>
2. Bei einem selbstsignierten SSL Zertifikat warnt Sie Ihr Browser und fordert Sie auf, den weiteren Besuch der Website zu vermeiden.





## 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, wird die Webseite des Jupyter Notebook-Servers angezeigt. An dieser Stelle werden Sie nach dem Passwort gefragt, das Sie zuvor eingerichtet haben.

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

## Eine Instanz bereinigen DLAMI

Wenn Sie Ihre DLAMI Instance nicht mehr benötigen, können Sie sie bei Amazon beenden oder kündigen EC2, um unerwartete Gebühren zu vermeiden.

Wenn Sie eine Instance beenden, können Sie sie behalten und sie später wieder starten, wenn Sie sie erneut verwenden möchten. Ihre Konfigurationen, Dateien und andere nichtflüchtige Informationen werden in einem Volume auf Amazon Simple Storage Service (Amazon S3) gespeichert. Solange Ihre Instance gestoppt ist, fallen für die Aufbewahrung des Volumes S3-Gebühren an, für Rechenressourcen fallen jedoch keine Gebühren an. Wenn Sie die Instance erneut starten, wird das Speichervolume mit Ihren Daten bereitgestellt.

Wenn Sie eine Instance beenden, ist sie weg und Sie können sie nicht erneut starten. Natürlich fallen bei einer beendeten Instance keine weiteren Gebühren für die Rechenressourcen an. Ihre Daten befinden sich jedoch weiterhin auf Amazon S3, und es können weiterhin S3-Gebühren anfallen. Um alle weiteren Gebühren im Zusammenhang mit Ihrer beendeten Instance zu vermeiden, müssen Sie auch das Speichervolumen auf Amazon S3 löschen. Anweisungen finden Sie unter [EC2Amazon-Instances beenden](#) im EC2Amazon-Benutzerhandbuch.

Weitere Informationen zu EC2 Amazon-Instanzstatus, wie z. B. `stopped` und `terminated`, finden Sie unter [Änderungen des EC2 Amazon-Instanzstatus](#) im EC2Amazon-Benutzerhandbuch.

# Verwenden eines DLAMI

## Themen

- [Deep Learning mit Conda verwenden AMI](#)
- [Verwenden der Deep Learning Base AMI](#)
- [Ausführen von Jupyter-Notebook-Tutorials](#)
- [Tutorials](#)

In den folgenden Abschnitten wird beschrieben, wie Deep Learning AMI mit Conda verwendet werden kann, um zwischen Umgebungen zu wechseln, Beispielcode aus jedem der Frameworks auszuführen und Jupyter auszuführen, sodass Sie verschiedene Notebook-Tutorials ausprobieren können.

## Deep Learning mit Conda verwenden AMI

### Themen

- [Einführung in das Deep Learning AMI mit Conda](#)
- [Loggen Sie sich in Ihr DLAMI](#)
- [Starten Sie die TensorFlow Umgebung](#)
- [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.

Deep Learning AMI mit Conda wurde so konfiguriert, dass Sie problemlos 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. Danach können Sie zu den ausführlicheren Tutorials von DLAMI wechseln oder die Beispiele zu den Frameworks auf der Projektseite jedes Frameworks nutzen.

## Loggen Sie sich in Ihr DLAMI

Nachdem Sie sich bei Ihrem Server angemeldet haben, wird Ihnen eine Servernachricht des Tages (MOTD) angezeigt, in der verschiedene Conda-Befehle beschrieben werden, mit denen Sie zwischen den verschiedenen Deep-Learning-Frameworks wechseln können. Im Folgenden finden Sie ein BeispielMOTD. Ihre MOTD Spezifik kann variieren, wenn neue Versionen von veröffentlicht DLAMI 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 TensorFlow Umgebung

### Note

Wenn Sie Ihre erste Conda-Umgebung starten, haben Sie bitte etwas Geduld, während diese geladen wird. Deep Learning AMI mit Conda installiert bei der ersten Aktivierung

des Frameworks automatisch die für Ihre EC2 Instanz am besten optimierte Version des Frameworks. 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 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 Platz in der DLAMI ausgeht, deinstallieren Sie Conda-Pakete, die Sie nicht verwenden:

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

## Verwenden der Deep Learning Base AMI

### Verwendung der Deep Learning Base AMI

The Base AMI verfügt über eine grundlegende Plattform mit GPU Treibern und Beschleunigungsbibliotheken, mit denen Sie Ihre eigene maßgeschneiderte Deep-Learning-

Umgebung bereitstellen können. Standardmäßig AMI ist die Umgebung mit einer beliebigen NVIDIA CUDA Version konfiguriert. Sie können auch zwischen verschiedenen Versionen von wechselnCUDA. Weitere Informationen zur Vorgehensweise finden Sie in den folgenden Anweisungen.

## CUDAVersionen konfigurieren

Sie können die CUDA Version überprüfen, indem Sie NVIDIA das nvcc Programm ausführen.

```
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 [DLAMIBase-Versionshinweisen](#).

## Ausführen von Jupyter-Notebook-Tutorials

Mit der jeweiligen Quelle der Deep-Learning-Projekte werden Tutorials und Beispiele geliefert, die in den meisten Fällen in einem beliebigen DLAMI ausgeführt werden können. 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

Zum Ausführen des Jupyter-Notebook-Tutorials, das auf dem DLAMI installiert ist, müssen Sie zuerst Folgendes ausführen: [Einen Jupyter Notebook-Server auf einer Instanz einrichten DLAMI](#).

Sobald der Jupyter-Server ausgeführt wird, können Sie die Tutorials über Ihren Webbrowser aufrufen. Wenn Sie Deep Learning AMI mit Conda ausführen oder Python-Umgebungen eingerichtet haben, können Sie die 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 stehen Benutzern von Deep Learning mit Conda zur Verfügung. AMI



**Note**

Viele Tutorials erfordern zusätzliche Python-Module, die in Ihrem DLAMI möglicherweise nicht eingerichtet sind. Wenn Sie eine Fehlermeldung wie "xyz module not found" erhalten, melden Sie sich bei dem DLAMI an, aktivieren die Umgebung wie oben beschrieben und installieren die erforderlichen Module.

**Tip**

Deep-Learning-Tutorials und -Beispiele basieren oft auf einem oder mehreren GPUs. Wenn Ihr Instance-Typ keinen hatGPU, müssen Sie möglicherweise einen Teil des Beispielcodes ändern, damit er ausgeführt werden kann.

## Navigation der installierten Tutorials

Sobald Sie beim Jupyter-Server angemeldet sind und das Verzeichnis mit den Tutorials sehen können (nur bei Deep Learning AMI mit Conda), werden Ihnen Ordner mit Tutorials für jeden Framework-Namen angezeigt. Wenn ein Framework in der Liste fehlt, sind die Tutorials für dieses Framework in Ihrem aktuellen DLAMI nicht 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 zum ersten Mal ein Notebook auf dem Deep Learning AMI mit Conda ausführen, wird es wissen wollen, welche Umgebung Sie verwenden möchten. Es wird eine Liste zur Auswahl bereitgestellt. Jede Umgebung hat einen Namen, der diesem Muster entspricht:

Environment (conda\_framework\_python-version)

Zum Beispiel könnten Sie sehen Environment (conda\_mxnet\_p36), was bedeutet, dass die Umgebung Python 3 hatMXNet. Die andere Variante davon wäre Environment (conda\_mxnet\_p27), was bedeutet, dass die Umgebung Python 2 hatMXNet.

**Tip**

Wenn Sie sich Gedanken darüber machen, welche Version von aktiv CUDA ist, können Sie dies unter anderem sehen, MOTD wenn Sie sich zum ersten Mal bei der 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 eine Fehlerprotokollierung, und möglicherweise wird ein out-of-memory Fehler angezeigt. 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 Anleitungen zur Verwendung von Deep Learning AMI mit der Software von Conda.

### Themen

- [Aktivieren von Frameworks](#)
- [Verteiltes Training mit Elastic Fabric Adapter](#)
- [GPUÜberwachung und Optimierung](#)
- [Der AWS Inferentia-Chip mit DLAMI](#)
- [Das Tool ARM64 DLAMI](#)
- [Inferenz](#)
- [Modellbereitstellung](#)

## Aktivieren von Frameworks

Im Folgenden sind die Deep-Learning-Frameworks aufgeführt, die auf Deep Learning AMI with 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 robustes Conda-Paket eines Frameworks veröffentlicht wird, wird es in der DLAMI getestet und 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
```

Starte das iPython Terminal.

```
(pytorch_p310)$ ipython
```

Führen Sie ein schnelles PyTorch Programm aus.

```
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

Mit Conda können Sie den neuesten PyTorch Build in einer oder beiden PyTorch Conda-Umgebungen auf Ihrem Deep Learning AMI 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`

- Um zu überprüfen, ob der PyTorch Nightly-Build gut mit dem MNIST Beispiel funktioniert, können Sie ein Testskript aus dem PyTorch Beispiel-Repository ausführen:

```
(pytorch_p310)$ cd ~  
(pytorch_p310)$ git clone https://github.com/pytorch/examples.git pytorch_examples  
(pytorch_p310)$ cd pytorch_examples/mnist  
(pytorch_p310)$ python main.py || exit 1
```

## Weitere Tutorials

Weitere Tutorials und Beispiele finden Sie in den offiziellen Dokumenten, der [PyTorch Dokumentation](#) und auf der [PyTorch](#) Website des Frameworks.

## TensorFlow 2

Dieses Tutorial zeigt, wie Sie TensorFlow 2 auf einer Instanz aktivieren, auf der Deep Learning AMI with Conda (DLAMI auf Conda) ausgeführt wird, und ein TensorFlow 2-Programm ausführen.

Wenn ein robustes Conda-Paket eines Frameworks veröffentlicht wird, wird es in der DLAMI getestet und vorinstalliert.

### 2 aktivieren TensorFlow

Um DLAMI mit TensorFlow Conda auf dem zu laufen

1. Um TensorFlow 2 zu aktivieren, öffnen Sie eine Amazon Elastic Compute Cloud (AmazonEC2) - Instanz von DLAMI with 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 für [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 Instance 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 AWS Cloud zu erreichen.

In den folgenden Themen werden erste Schritte bei Verwendung von EFA mit dem DLAMI beschrieben.

### Note

Wählen Sie Ihre DLAMI aus dieser [Basisliste GPU DLAMI](#)

## Themen

- [Eine AWS Deep Learning AMIs Instance starten mit EFA](#)
- [Verwenden von EFA auf dem DLAMI](#)

## Eine AWS Deep Learning AMIs Instance starten mit EFA

Die neueste Version von Base DLAMI ist sofort einsatzbereit EFA und wird mit den erforderlichen Treibern, Kernelmodulen, libfabric, openmpi und dem [NCCLOFIPlugin](#) für Instances geliefert. GPU

[Die unterstützten CUDA Versionen einer Base finden Sie DLAMI in den Versionshinweisen.](#)

## Hinweis:

- Wenn Sie eine NCCL Anwendung mit `mpirun` ausführen, müssen Sie den vollständigen Pfad zur 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 [Verwenden von EFA auf dem DLAMI](#) gezeigt.

## Themen

- [Vorbereiten einer EFA-aktivierten Sicherheitsgruppe](#)
- [Starten Ihrer Instance](#)
- [Überprüfen des EFA-Anhangs](#)

## Vorbereiten einer EFA-aktivierten Sicherheitsgruppe

EFA erfordert eine Sicherheitsgruppe, die allen eingehenden und ausgehenden Datenverkehr von und zu der Sicherheitsgruppe selbst zulässt. Weitere Informationen finden Sie in der [EFA Dokumentation](#).

1. Öffnen Sie die EC2 Amazon-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie im Navigationsbereich Security Groups (Sicherheitsgruppen) und anschließend Create Security Group (Sicherheitsgruppe erstellen) aus.
3. Führen Sie im Fenster Create Security Group Folgendes aus:
  - Geben Sie für Security group name (Name der Sicherheitsgruppe) einen beschreibenden Namen für die Sicherheitsgruppe ein, wie etwa `EFA-enabled security group`.
  - (Optional:) Geben Sie unter Description (Beschreibung) eine kurze Beschreibung der Sicherheitsgruppe ein.
  - Wählen Sie für VPC die Instances aus, VPC in denen 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 Inbound und Outbound wie folgt vor:
  - Wählen Sie Edit aus.

- Wählen Sie für Type (Typ) die Option All traffic (Gesamter Datenverkehr) aus.
  - Wählen Sie für Source (Quelle) die Option Custom (Benutzerdefiniert) aus.
  - Fügen Sie die Sicherheitsgruppen-ID, die Sie kopiert haben, in das Feld ein.
  - Wählen Sie Save (Speichern) aus.
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-Instance kommunizieren.

## Starten Ihrer Instance

EFAon the AWS Deep Learning AMIs wird derzeit von den folgenden Instance-Typen und Betriebssystemen unterstützt:

- P3dn: Amazon Linux 2, Ubuntu 20.04
- P4d, P4de: Amazon Linux 2, Amazon Linux 2023, Ubuntu 20.04, Ubuntu 22.04
- P5, P5e, P5en: Amazon Linux 2, Amazon Linux 2023, Ubuntu 20.04, Ubuntu 22.04

Der folgende Abschnitt zeigt, wie Sie eine aktivierte Instance starten. EFA DLAMI Weitere Informationen zum Starten einer EFA aktivierten Instance finden Sie unter [Launch EFA -Enabled Instances in einer Cluster Placement-Gruppe](#).

1. Öffnen Sie die EC2 Amazon-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie Launch Instance aus.
3. Wählen Sie auf der AMI Seite Wählen Sie ein unterstütztes Produkt aus, das auf der [Seite mit den DLAMI Versionshinweisen DLAMI](#) gefunden wurde
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 Netzwerk und Subnetz das VPC Subnetz aus, in dem die Instances gestartet werden sollen.



- [Optional] Wählen Sie für Placement-Gruppe die Option Instance zur Placement-Gruppe hinzufügen aus. Um eine optimale Leistung zu erzielen, starten Sie die Instances innerhalb einer Platzierungsgruppe.
  - [Optional] Wählen Sie für Placement-Gruppenname die Option Zu einer neuen Placement-Gruppe hinzufügen aus, geben Sie einen aussagekräftigen Namen für die Placement-Gruppe ein und wählen Sie dann für Placement-Gruppenstrategie die Option Cluster aus.
  - Achten Sie darauf, den „Elastic Fabric Adapter“ auf dieser Seite zu aktivieren. Wenn diese Option deaktiviert ist, ändern Sie das Subnetz in ein Subnetz, das den ausgewählten Instance-Typ unterstützt.
  - Wählen Sie im Abschnitt Network Interfaces (Netzwerkschnittstellen für das Gerät eth0 die Option New network interface (Neue Netzwerkschnittstelle) aus. Sie können optional eine primäre IPv4 Adresse und eine oder mehrere sekundäre IPv4 Adressen angeben. Wenn Sie die Instance in einem Subnetz starten, dem ein IPv6 CIDR Block zugeordnet ist, können Sie optional eine primäre IPv6 Adresse und eine oder mehrere sekundäre IPv6 Adressen angeben.
  - Wählen Sie Next: Add Storage aus.
6. Auf der Seite Add Storage (Speicher hinzufügen) können Sie neben den durch das AMI festgelegten Datenträger (wie z. B. dem Root-Gerät-Datenträger) auch Datenträger angeben, die an die Instance angefügt werden sollen. Wählen Sie anschließend Next: Add Tags (Weiter: Tags hinzufügen) aus.
  7. Geben Sie auf der Seite Add Tags (Tags (Markierungen) hinzufügen) Tags (Markierungen) für die Instances an, z. B. einen benutzerfreundlichen Namen, und wählen Sie anschließend Next: Configure Security Group (Weiter: Sicherheitsgruppe konfigurieren).
  8. Wählen Sie auf der Seite Sicherheitsgruppe konfigurieren für Sicherheitsgruppe zuweisen die Option Eine bestehende Sicherheitsgruppe auswählen und wählen Sie dann die Sicherheitsgruppe aus, die Sie zuvor erstellt haben.
  9. Klicken Sie auf Review and Launch.
  10. Überprüfen Sie auf der Seite Review Instance Launch (Instance-Start überprüfen) Ihre Einstellungen und wählen Sie anschließend Launch (Starten) aus, um ein Schlüsselpaar auszuwählen und Ihre Instance zu starten.

## Überprüfen des EFA-Anhangs

### Über die Konsole

Überprüfen Sie nach dem Start der Instance die Instance-Details in der AWS Konsole. Wählen Sie dazu die Instanz in der EC2 Konsole aus und schauen Sie sich die Registerkarte Beschreibung im unteren Bereich der Seite an. Suchen Sie den Parameter "Network Interfaces: eth0" und klicken Sie auf "eth0", wodurch ein Pop-up geöffnet wird. Stellen Sie sicher, dass der 'Elastic Fabric Adapter' aktiviert ist.

Wenn EFA nicht aktiviert ist, können Sie dies beheben, indem Sie eine der folgenden Vorgehensweisen ausführen:

- Beenden Sie die EC2 Instance und starten Sie eine neue mit den gleichen Schritten. Stellen Sie sicher, dass EFA angefügt ist.
- Anfügen von EFA an eine vorhandene Instance.
  1. Gehen Sie in der EC2 Konsole zu Netzwerkschnittstellen.
  2. Klicken Sie auf "Create a Network Interface (Netzwerkschnittstelle erstellen)".
  3. Wählen Sie dasselbe Subnetz aus, in dem sich Ihre Instance befindet.
  4. Stellen Sie sicher, dass der 'Elastic Fabric Adapter' aktiviert ist, und klicken Sie auf Erstellen.
  5. Gehen Sie zurück zum Tab EC2 Instances und wählen Sie Ihre Instance aus.
  6. Wechseln Sie zu "Actions: Instance State (Aktionen: Instance-Status)" und beenden Sie die Instance, bevor Sie EFA anfügen.
  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 sie nicht mehr reagiert oder nicht vollständig ausgeführt wird, stellen Sie sicher, dass Ihre Sicherheitsgruppe über die richtigen Regeln für eingehende/ausgehende Nachrichten verfügt.

## Verwenden von EFA auf dem DLAMI

Im folgenden Abschnitt wird beschrieben, wie Sie EFA verwenden können, um Anwendungen mit mehreren Knoten auf dem AWS Deep Learning AMLs auszuführen.

Ausführen von Anwendungen mit mehreren Knoten mit EFA

Um eine Anwendung auf einem Knotencluster auszuführen, ist die folgende Konfiguration erforderlich

Themen

- [Aktivieren Sie Passwordless SSH](#)
- [Erstellen einer Hosts-Datei](#)
- [NCCLTests](#)

Aktivieren Sie Passwordless SSH

Wählen Sie einen Knoten im Cluster als Führungsknoten aus. Die verbleibenden Knoten werden als Mitgliedsknoten bezeichnet.

1. Generieren Sie auf dem Leader-Knoten das RSA Schlüsselpaar.

```
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` in einen `~/.ssh/authorized_keys` der Mitgliedsknoten im Cluster und hängen Sie ihn an.
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
```

## NCCLTests

### Note

Diese Tests wurden mit EFA Version 1.30.0 und 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

### NCCLTest zur Nachrichtenübertragung mit mehreren Knoten

Der `nccl_message_transfer` ist ein einfacher Test, um sicherzustellen, dass das NCCL OFI Plugin wie erwartet funktioniert. Der Test validiert die Funktionalität des Verbindungsaufbaus und der Datenübertragung. NCCL APIs Stellen Sie sicher, dass Sie den vollständigen Pfad zu `mpirun` verwenden, wie im Beispiel gezeigt, wenn Sie Anwendungen mit `ausführenNCCL`. EFA Ändern Sie

die Parameter `np` `N` je nach Anzahl der Instanzen und GPUs in Ihrem Cluster. Weitere Informationen finden Sie in der [AWS OFINCCLDokumentation](#).

Der folgende `nccl_message_transfer`-Test bezieht sich auf eine generische `xx.x`-Version. CUDA Sie können die Befehle für jede verfügbare CUDA Version in Ihrer EC2 Amazon-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 diese 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 zu überprüfenEFA, führen Sie den NCCL Standard-Leistungstest aus, der im offiziellen [NCCL-Tests-Repo](#) verfügbar ist. Im DLAMI Lieferumfang ist dieser Test bereits für CUDA XX.X. Sie können auf ähnliche Weise Ihr eigenes Skript ausführen. EFA

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 Anwendungen mit ausführen. NCCL EFA
- Ändern Sie die Parameter np und N basierend auf der Anzahl der Instanzen und GPUs in Ihrem Cluster.
- Fügen Sie das NCCL \_ DEBUG = INFO -Flag hinzu und stellen Sie sicher, dass in den Protokollen die EFA Nutzung als „Ausgewählter Anbieter istEFA“ 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 EC2 Amazon-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 --mca pml ^cm \  
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/  
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib64:/opt/amazon/openmpi/  
lib64:$LD_LIBRARY_PATH \  
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to  
none \  
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n  
100 | tee ${TRAINING_LOG}
```

- Ubuntu 20.04:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib:/opt/amazon/openmpi/
lib:$LD_LIBRARY_PATH \
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
none \
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
100 | tee ${TRAINING_LOG}
```

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
```





628.2	256	4	float	sum	-1	632.7	0.00	0.00	0
	0.00	0.00	0						
629.6	512	8	float	sum	-1	627.4	0.00	0.00	0
	0.00	0.00	0						
631.7	1024	16	float	sum	-1	632.2	0.00	0.00	0
	0.00	0.00	0						
634.2	2048	32	float	sum	-1	631.0	0.00	0.00	0
	0.00	0.00	0						
633.6	4096	64	float	sum	-1	623.3	0.01	0.01	0
	0.01	0.01	0						
633.5	8192	128	float	sum	-1	635.1	0.01	0.01	0
	0.01	0.01	0						
637.0	16384	256	float	sum	-1	634.8	0.03	0.02	0
	0.03	0.02	0						
636.8	32768	512	float	sum	-1	647.9	0.05	0.05	0
	0.05	0.05	0						
667.0	65536	1024	float	sum	-1	658.9	0.10	0.09	0
	0.10	0.09	0						
662.9	131072	2048	float	sum	-1	671.9	0.20	0.18	0
	0.20	0.19	0						
685.1	262144	4096	float	sum	-1	692.1	0.38	0.36	0
	0.38	0.36	0						
696.6	524288	8192	float	sum	-1	715.3	0.73	0.69	0
	0.75	0.71	0						
729.2	1048576	16384	float	sum	-1	734.6	1.43	1.34	0
	1.44	1.35	0						
794.5	2097152	32768	float	sum	-1	785.9	2.67	2.50	0
	2.64	2.47	0						
837.6	4194304	65536	float	sum	-1	837.2	5.01	4.70	0
	5.01	4.69	0						
931.4	8388608	131072	float	sum	-1	929.2	9.03	8.46	0
	9.01	8.44	0						
1772.8	16777216	262144	float	sum	-1	1773.6	9.46	8.87	0
	9.46	8.87	0						
2116.1	33554432	524288	float	sum	-1	2110.2	15.90	14.91	0
	15.86	14.87	0						
2658.1	67108864	1048576	float	sum	-1	2650.9	25.32	23.73	0
	25.25	23.67	0						
3945.9	134217728	2097152	float	sum	-1	3943.1	34.04	31.91	0
	34.01	31.89	0						
7178.6	268435456	4194304	float	sum	-1	7216.5	37.20	34.87	0
	37.39	35.06	0						
13676	536870912	8388608	float	sum	-1	13680	39.24	36.79	0
	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}"

```

## GPUÜberwachung und Optimierung

Der folgende Abschnitt führt Sie durch GPU Optimierungs- und Überwachungsoptionen. Dieser Abschnitt ist wie ein typischer Workflow mit Überwachung, Beaufsichtigung, Vorverarbeitung und Schulung aufgebaut.

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

### Überwachen

Auf DLAMI Ihrem sind mehrere GPU Überwachungstools vorinstalliert. Diese Anleitung erwähnt auch Tools, die heruntergeladen und installiert werden können.

- [Überwachen Sie mit GPUs CloudWatch](#)- ein vorinstalliertes Hilfsprogramm, das GPU Nutzungsstatistiken an Amazon CloudWatch meldet.
- [nvidia-smi CLI](#) — ein Hilfsprogramm zur Überwachung der allgemeinen Rechen- und Speicherauslastung. GPU Es ist auf Ihrem () vorinstalliert. AWS Deep Learning AMIs DLAMI
- [NVMLC-Bibliothek](#) — eine C-basierte Bibliothek für API den direkten Zugriff auf GPU Überwachungs- und Verwaltungsfunktionen. Dies wird vom nvidia-smi CLI unter der Haube verwendet und ist auf Ihrem vorinstalliert. DLAMI Dazu gehören weiterhin Python- und Perl-Anbindungen zur Unterstützung der Bereitstellung in diesen Sprachen. Das auf Ihrem Gerät vorinstallierte Hilfsprogramm gpumon.py verwendet das Paket pynvml von. DLAMI [nvidia-ml-py](#)
- [NVIDIADCGM](#)- Ein Cluster-Management-Tool. Besuchen Sie die Entwicklerseite, um zu erfahren, wie Sie dieses Tool installieren und konfigurieren.

 Tip

Im NVIDIA Entwickler-Blog finden Sie die neuesten Informationen zur Verwendung der auf Ihrem Computer installierten CUDA ToolsDLAMI:

- [Überwachung TensorCore der Auslastung mit Nsight IDE und nvprof.](#)

## Überwachen Sie mit GPUs CloudWatch

Wenn Sie Ihr Gerät DLAMI zusammen mit einem verwenden, suchen GPU Sie möglicherweise nach Möglichkeiten, seine Verwendung während des Trainings oder der Inferenz nachzuverfolgen. 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 Ihren in DLAMI den [Unified CloudWatch Agent](#), um GPU Metriken zu konfigurieren und die Nutzung von GPU Coprozessen in Amazon EC2 Accelerated Instances zu überwachen.

Es gibt vier Möglichkeiten, [GPUMetriken](#) mit Ihrem DLAMI zu konfigurieren:

- [Konfigurieren Sie minimale GPU Metriken](#)
- [Konfigurieren Sie GPU Teilmetriken](#)
- [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 EC2 IAM Amazon-Instance-Berechtigungen konfigurieren, an die Ihre Instance Metriken weiterleiten kann CloudWatch. Ausführliche Schritte finden Sie unter [IAMRollen und Benutzer für die Verwendung mit dem CloudWatch Agenten erstellen](#).

## Konfigurieren Sie minimale GPU Metriken

Konfigurieren Sie minimale GPU Metriken mithilfe des `dlami-cloudwatch-agent@minimal` systemd Dienstes. Dieser Dienst konfiguriert die folgenden Metriken:

- `utilization_gpu`
- `utilization_memory`

Sie finden den `systemd` Service 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 GPU Teilmetriken

Konfigurieren Sie GPU Teilmetriken mithilfe des `dlami-cloudwatch-agent@partial` systemd Dienstes. Dieser Service 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` Service für alle verfügbaren vorkonfigurierten GPU Metriken an der folgenden Stelle:

```
/opt/aws/amazon-cloudwatch-agent/etc/dlami-amazon-cloudwatch-agent-all.json
```

Aktivieren und starten Sie den `systemd` Dienst mit den folgenden Befehlen:

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

Konfigurieren Sie benutzerdefinierte GPU Metriken

Wenn die vorkonfigurierten Metriken Ihren Anforderungen nicht entsprechen, können Sie eine benutzerdefinierte CloudWatch Agentenkonfigurationsdatei erstellen.

Erstellen Sie eine benutzerdefinierte Konfigurationsdatei

Informationen zum Erstellen einer benutzerdefinierten Konfigurationsdatei finden Sie in den detaillierten Schritten unter [Manuelles Erstellen oder Bearbeiten der CloudWatch Agentenkonfigurationsdatei](#).

Gehen Sie für dieses Beispiel davon aus, dass sich die Schemadefinition unter befindet `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`.

Konfigurieren Sie Metriken mit Ihrer benutzerdefinierten Datei

Führen Sie den folgenden Befehl aus, um den CloudWatch Agenten entsprechend Ihrer benutzerdefinierten Datei zu konfigurieren:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
-a fetch-config -m ec2 -s -c \
file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
```

## Sicherheitspatches für den Agenten AWS CloudWatch

Neu veröffentlichte Versionen DLAMIs sind mit den neuesten verfügbaren AWS CloudWatch Agent-Sicherheitspatches konfiguriert. In den folgenden Abschnitten finden Sie Informationen zur Aktualisierung Ihres aktuellen Sicherheitspatches DLAMI mit den neuesten Sicherheits-Patches, abhängig vom Betriebssystem Ihrer Wahl.

### Amazon Linux 2

Verwenden Sie diese Optionum, um die neuesten AWS CloudWatch Agenten-Sicherheitspatches für Amazon Linux 2 zu erhaltenDLAMI.

```
sudo yum update
```

### Ubuntu

Um die neuesten AWS CloudWatch Sicherheitspatches für Ubuntu DLAMI 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 dem DLAMI vorinstalliert. Es integriert sich in CloudWatch und unterstützt die Überwachung von GPU Speicher, GPU Temperatur und GPU Stromverbrauch pro GPU Nutzung. Das Skript sendet die überwachten Daten regelmäßig an CloudWatch. Sie können die Granularität für die zu sendenden Daten konfigurieren, CloudWatch indem Sie einige Einstellungen im Skript ändern. Bevor Sie das Skript starten, müssen Sie jedoch einrichten, um die Metriken CloudWatch zu empfangen.


Wie richte ich die GPU Überwachung ein und führe sie aus mit 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, ist „cloudwatch:PutMetricData“. Die Richtlinie, die hinzugefügt wird, lautet wie folgt:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

 Tip

Weitere Informationen zum Erstellen eines IAM Benutzers und zum Hinzufügen von Richtlinien für CloudWatch diesen Benutzer finden Sie in der [CloudWatch Dokumentation](#).

2. Führen Sie auf Ihrem DLAMI die Option [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 das Verzeichnis, das README im folgenden Codeblock definiert ist. 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 NOT in `us-east-1` befindet.
  - Ändern Sie andere Parameter wie den CloudWatch namespace oder den Berichtszeitraum mit `store_reso`
- Derzeit unterstützt das Skript nur Python 3. Aktivieren Sie die Python 3-Umgebung des bevorzugten Frameworks oder aktivieren Sie die allgemeine Python 3-Umgebung des DLAMI.

```
$ source activate python3
```

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

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

- Öffnen Sie Ihren Browser mit <https://console.aws.amazon.com/cloudwatch/> und wählen Sie dann die Metrik. Es wird einen Namespace 'DeepLearningTrain' haben.

### Tip

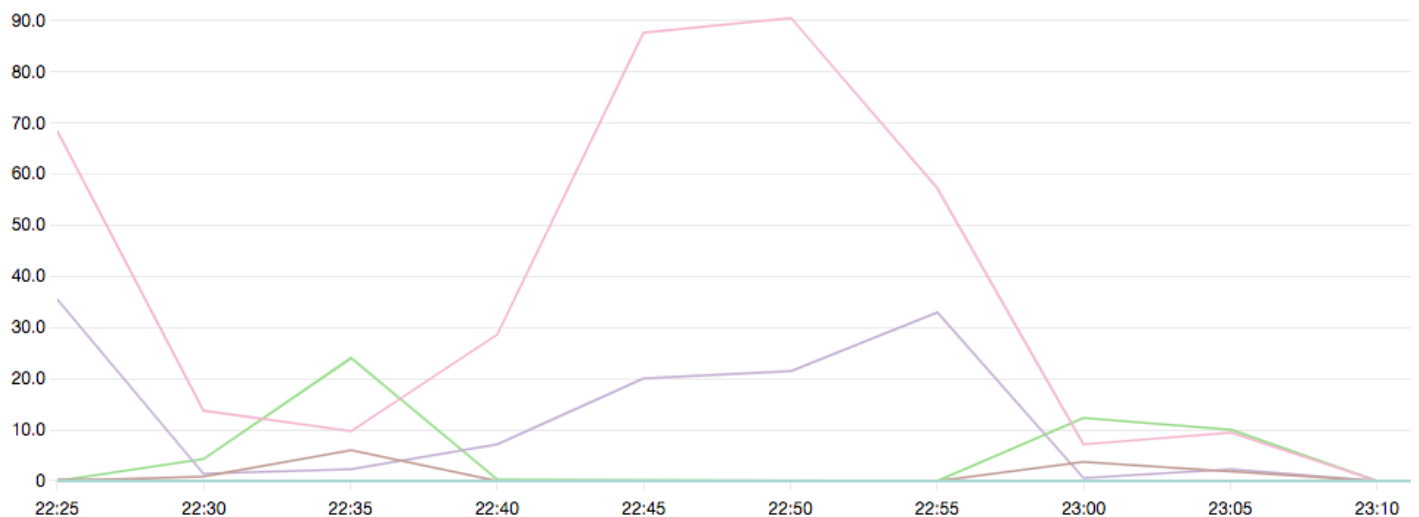
Sie können den Namespace durch Modifizierung von `gpumon.py` ändern. Sie können auch das Berichtsintervall durch Anpassung von `store_reso` ändern.

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



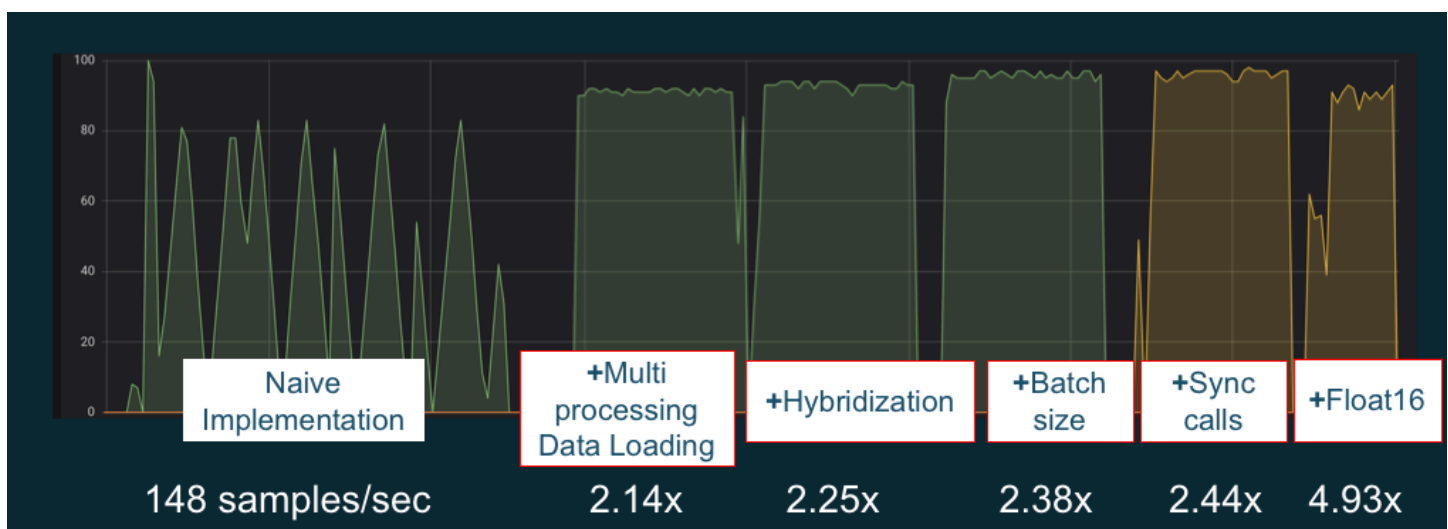
Diese anderen Themen zum Thema GPU Überwachung und Optimierung könnten Sie interessieren:

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

## Optimierung

Um das Beste aus Ihrem herauszuholen GPUs, können Sie Ihre Datenpipeline optimieren und Ihr Deep-Learning-Netzwerk optimieren. Wie in der folgenden Tabelle beschrieben, kann eine naive oder einfache Implementierung eines neuronalen Netzwerks das Potenzial GPU inkonsistent und nicht in vollem Umfang nutzen. Wenn Sie Ihre Vorverarbeitung und das Laden von Daten optimieren, können Sie den Engpass zwischen Ihren Daten verringern. CPU GPU Sie können das neuronale Netzwerk selbst anpassen, indem Sie Hybridisierung (wenn vom Framework unterstützt) verwenden, die Stapelgröße anpassen sowie Aufrufe synchronisieren. Sie können in den meisten Frameworks auch Schulungen mit mehreren Präzisionen verwenden, was dramatische Verbesserungen des Durchsatzes mit sich bringen kann.

Das folgende Diagramm zeigt die kumulativen Leistungssteigerungen bei der Anwendung verschiedener Optimierungen. Ihre Ergebnisse hängen von den verarbeiteten Daten und dem optimierten Netzwerk ab.



Beispiel für GPU Leistungsoptimierungen. Quelle des Diagramms: [Leistungstricks mit Gluon MXNet](#)

In den folgenden Anleitungen werden Optionen vorgestellt, die für Sie geeignet sind DLAMI und Ihnen helfen, die GPU Leistung zu steigern.

## Themen

- [Vorverarbeitung](#)
- [Training](#)

## Vorverarbeitung

Die Datenvorverarbeitung durch Transformationen oder Erweiterungen kann oft ein CPU gebundener Prozess sein, und dies kann zu einem Engpass in Ihrer gesamten Pipeline führen. Frameworks verfügen über integrierte Operatoren für die Bildverarbeitung, aber DALI (Data Augmentation Library) weist eine verbesserte Leistung gegenüber den integrierten Optionen von Frameworks auf.

- **NVIDIADatenerweiterungsbibliothek (DALI):** DALI verlagert die Datenerweiterung auf die GPU. Es ist nicht auf dem vorinstalliert DLAMI, aber Sie können darauf zugreifen, indem Sie es installieren oder einen unterstützten Framework-Container auf Ihrer DLAMI oder einer anderen Amazon Elastic Compute Cloud-Instance laden. Einzelheiten finden Sie auf der [DALIProjektseite](#) auf der NVIDIA Website. Ein Beispiel für einen Anwendungsfall und zum Herunterladen von Codebeispielen finden Sie im Beispiel [SageMaker Preprocessing Training Performance](#).
- **nvJPEG:** eine GPU beschleunigte JPEG Decoder-Bibliothek für C-Programmierer. Sie unterstützt die Dekodierung einzelner Bilder oder Stapel sowie nachfolgende Transformationsoperationen, wie sie beim Deep Learning üblich sind. nv JPEG ist bereits integriert DALI, oder Sie können es von der [NVIDIAnvjpeg-Seite der Website](#) herunterladen und separat verwenden.

Diese anderen Themen zur Überwachung und Optimierung könnten Sie interessieren: GPU

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

## Training

Mit Schulungen mit gemischter Präzision können Sie größere Netzwerke mit derselben Speichergröße bereitstellen oder die Speichernutzung gegenüber Ihrem Netzwerk mit einzelner oder doppelter Präzision reduzieren - die Rechenleistung wird dadurch sicher gesteigert. Dazu kommt der Vorteil kleinerer und schnellerer Datenübertragungen, ein wichtiger Faktor für verteilte Schulungen mit mehreren Knoten. Um die Schulung mit gemischter Präzision nutzen zu können, müssen Sie das Data Casting und die Verlustskalierung anpassen. Nachfolgend finden Sie Anleitungen dazu, wie Sie dies für Frameworks tun können, die gemischte Präzisionen unterstützen.

- [NVIDIA Deep Learning SDK](#) — Dokumente auf der NVIDIA Website, in denen die Implementierung mit gemischter Genauigkeit für MXNet PyTorch, und beschrieben wird. 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 Blogseite. NVIDIA
- [Training mit gemischter Präzision mit Float16 mit MXNet - einem Artikel](#) auf der Website. FAQ MXNet
- [NVIDIA Apex: ein Tool für einfaches Training mit gemischter Präzision mit PyTorch](#) - einem Blogartikel auf der Website. NVIDIA

Diese anderen Themen zum Thema GPU Überwachung und Optimierung könnten Sie interessieren:

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

## Der AWS Inferentia-Chip mit DLAMI

AWS Inferentia ist ein maßgeschneiderter Chip für maschinelles Lernen, der speziell für leistungsstarke AWS Inferenzvorhersagen entwickelt wurde. Um den Chip zu verwenden, richten Sie eine Amazon Elastic Compute Cloud-Instance 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 die () integriert. AWS Deep Learning AMIs DLAMI

Die folgenden Themen zeigen Ihnen, wie Sie mit der Nutzung von Inferentia beginnen können.  
DLAMI

### Inhalt

- [Starten einer DLAMI Instanz mit Neuron AWS](#)
- [Verwendung des mit Neuron DLAMI AWS](#)

## Starten einer DLAMI Instanz mit Neuron AWS

Die neueste Version DLAMI ist sofort mit AWS Inferentia einsatzbereit und wird mit dem AWS Neuron-Paket geliefert. API Informationen zum Starten einer DLAMI Instanz finden Sie unter [Starten und Konfigurieren](#) einer DLAMI. Wenn Sie eine haben DLAMI, 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 mit Neuron eingerichtet und 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:

- **NEURONDEVICE:** Die logische ID, die dem zugewiesen ist. NeuronDevice Diese ID wird verwendet, wenn mehrere Laufzeiten für die Verwendung verschiedener NeuronDevices Laufzeiten konfiguriert werden.
- **NEURONCORES:** Die Anzahl der NeuronCores vorhandenen in der NeuronDevice.
- **NEURONMEMORY:** Die Menge an DRAM Speicher in der NeuronDevice.
- **CONNECTEDDEVICES:** Andere, die NeuronDevices mit dem verbunden sind NeuronDevice.
- **PCIBDF:** Die PCI Bus-Gerätefunktions-ID (BDF) des NeuronDevice.

## Anzeigen des Ressourcenverbrauchs

Mit dem `neuron-top` Befehl können Sie nützliche Informationen über die NeuronCore und CPU v-Auslastung, die Speichernutzung, geladene Modelle und Neuron-Anwendungen abrufen. 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:

```

neuron-top
Neuroncore Utilization
NC0          NC1          NC2          NC3
ND0 [|||||] [ 100%] [|||||] [ 100%] [|||||] [ 100%] [|||||] [ 100%]
ND1 [|||||] [ 0.00%] [|||||] [ 0.00%] [|||||] [ 0.00%] [|||||] [ 0.00%]
ND2 [|||||] [ 0.00%] [|||||] [ 0.00%] [|||||] [ 0.00%] [|||||] [ 0.00%]
ND3 [|||||] [ 0.00%] [|||||] [ 0.00%] [|||||] [ 0.00%] [|||||] [ 0.00%]

vCPU and Memory Info
System vCPU Usage [|||||] [ 8.69%, 9.47%] Runtime vCPU Usage [|||||] [ 3.22%, 5.30%]
Runtime Memory Host [|||||] [ 2.5MB/ 46.0GB] Runtime Memory Device [|||||] [ 198.3MB]

Loaded Models
[ - ] ND 0
[ - ] NC0
  -integ-tests/out-test7_resnet50_v2_fp16_b1_tpb1_tf
[ + ] NC1
[ + ] NC2
[ + ] NC3

Model ID          Host Memory          Device Memory
-----
10001             638.5KB             49.6MB
638.5KB          49.6MB
638.5KB          49.6MB
638.5KB          49.6MB

Neuron Apps
q: quit          [1]:inference app 1          [2]:inference app 2          [3]:inference app 3          [4]:inference app 4
arrows: move tree selection  enter: expand/collapse tree item  x: expand/collapse entire tree  a/d: previous/next tab  1-9: select tab

```

[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 Format auf stdout. JSON 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 zur Aktualisierung der SDK Neuron-Software finden Sie im DLAMI AWS Neuron Setup Guide.](#)

## Nächster Schritt

### [Verwendung des mit Neuron DLAMI AWS](#)

## Verwendung des mit Neuron DLAMI 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 danach die Artefakte zur Ausführung an die Inf1-Instanzen. AWS Deep Learning AMIs (DLAMI) enthält vorinstalliert alles, 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 mit Inferentia verwenden. DLAMI

### Inhalt

- [Verwendung von TensorFlow -Neuron und dem Neuron Compiler AWS](#)
- [Verwenden von AWS Neuron Serving TensorFlow](#)
- [Verwenden von MXNet -Neuron und dem Neuron-Compiler AWS](#)
- [Verwenden von MXNet -Neuron Model Serving](#)
- [Verwenden von PyTorch -Neuron und dem Neuron Compiler AWS](#)

## Verwendung von TensorFlow -Neuron und dem Neuron Compiler AWS

Dieses Tutorial zeigt, wie Sie mit dem AWS Neuron-Compiler das Keras ResNet -50-Modell kompilieren und als gespeichertes Modell im Format exportieren. SavedModel Dieses Format ist ein typisches TensorFlow austauschbares Modellformat. Sie lernen außerdem, wie die Inferenz für eine Inf1-Instance mit Beispieldaten ausgeführt wird.

Weitere Informationen zum Neuron finden Sie in der SDK [AWS SDKNeuron-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 außerdem mit Deep Learning und mit der Verwendung von 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 Beispield 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 dasselbe 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 in `/usr/local/bin/tensorflow_model_server_neuron` und ist im DLAMI vorinstalliert.

[Weitere Informationen zum Neuron finden Sie in der SDK Neuron-Dokumentation.AWS SDK](#)

#### 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 außerdem mit Deep Learning und mit der Verwendung von 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
model_dir = "./resnet50/1"
tf.saved_model.simple_save(sess, model_dir, inputs, outputs)

# compile the model for Inferentia
neuron_model_dir = os.path.join(os.path.expanduser('~'), 'resnet50_inf1', '1')
tf.neuron.saved_model.compile(model_dir, neuron_model_dir, batch_size=1)
```

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 auf den Inferentia-Geräten bereitgestellt, DRAM 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 G ausRPC, was ein Service-Framework ist.

```
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)]]
```

## Verwenden von MXNet -Neuron und dem Neuron-Compiler AWS

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

In diesem Beispiel verwenden Sie die, um ein ResNet -50-Modell API zu kompilieren und damit Inferenzen auszuführen.

[Weitere Informationen zum Neuron finden Sie in der SDK Neuron-Dokumentation.AWS SDK](#)

## 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 außerdem mit Deep Learning und mit der Verwendung von DLAMI vertraut sein.

## Aktivieren der Conda-Umgebung

Aktivieren Sie die MXNet -Neuron Conda-Umgebung mit dem folgenden Befehl:

```
source activate aws_neuron_mxnet_p36
```

Führen Sie Folgendes aus, um die aktuelle Conda-Umgebung zu verlassen:

```
source deactivate
```

## Resnet50-Kompilierung

Erstellen Sie das Python-Skript namens **mxnet\_compile\_resnet50.py** mit folgendem Inhalt. Dieses Skript verwendet die MXNet -Neuron-Kompilierungspython, um ein API 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 von MXNet -Neuron Model Serving](#)

## Verwenden von MXNet -Neuron Model Serving

In diesem Tutorial lernen Sie, ein vortrainiertes MXNet Modell zu verwenden, um Bildklassifizierung in Echtzeit mit Multi Model Server (MMS) durchzuführen. MMS ist ein flexibles easy-to-use Tool zur Bereitstellung von Deep-Learning-Modellen, die mit einem beliebigen Framework für maschinelles Lernen oder Deep Learning trainiert wurden. Dieses Tutorial beinhaltet einen Kompilierungsschritt mit AWS Neuron und eine Implementierung von MMS. MXNet

Weitere Informationen zum Neuron finden Sie in der SDK [AWS SDKNeuron-Dokumentation](#).

### 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 außerdem mit Deep Learning und mit der Verwendung von DLAMI vertraut sein.

### Aktivieren der Conda-Umgebung

Aktivieren Sie die MXNet -Neuron Conda-Umgebung mit dem folgenden Befehl:

```
source activate aws_neuron_mxnet_p36
```

Führen Sie Folgendes aus, um die aktuelle Conda-Umgebung zu verlassen:

```
source deactivate
```

### Herunterladen des Beispielcodes

Um dieses Beispiel auszuführen, laden Sie den Beispielcode mit den folgenden Befehlen herunter:

```
git clone https://github.com/awslabs/multi-model-server
cd multi-model-server/examples/mxnet_vision
```

## Kompilieren des Modells

Erstellen Sie das Python-Skript namens `multi-model-server-compile.py` mit folgendem Inhalt. Dieses Skript kompiliert das Modell ResNet 50 für das Inferentia-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/squeezenet_v1.1/synset.txt
```

Erstellen Sie eine benutzerdefinierte Service-Klasse unter Verwendung der Vorlage im Ordner `model_server_template`. Kopieren Sie die Vorlage mit dem folgenden Befehl in das aktuelle Arbeitsverzeichnis:

```
cp -r ../model_service_template/* .
```

Bearbeiten Sie das Modul `mxnet_model_service.py`, indem Sie den Kontext `mx.cpu()` wie folgt durch den Kontext `mx.neuron()` ersetzen. Sie müssen auch die überflüssige Datenkopie für `auskommentierenmodel_input`, weil MXNet -Neuron das NDArray und Gluon nicht unterstützt. APIs

```
...
self.mxnet_ctx = mx.neuron() if gpu_id is None else mx.gpu(gpu_id)
...
#model_input = [item.as_in_context(self.mxnet_ctx) for item in model_input]
```



Erstellen Sie mit den folgenden Befehlen ein Paket des Modells mit model-archiver:

```
cd ~/multi-model-server/examples
model-archiver --force --model-name resnet-50_compiled --model-path mxnet_vision --
handler mxnet_vision_service:handle
```

## Ausführen der Inferenz

Starten Sie den Multi Model Server und laden Sie das Modell, das den verwendet, RESTful API mithilfe der folgenden Befehle. Stellen Sie sicher, dass neuron-rtd mit den Standardeinstellungen ausgeführt wird.

```
cd ~/multi-model-server/
multi-model-server --start --model-store examples > /dev/null # Pipe to log file if you
want to keep a log of MMS
curl -v -X POST "http://localhost:8081/models?
initial_workers=1&max_workers=4&synchronous=true&url=resnet-50_compiled.mar"
sleep 10 # allow sufficient time to load model
```

Führen Sie die Inferenz mit den folgenden Befehlen mit einem Beispielbild aus:

```
curl -O https://raw.githubusercontent.com/awslabs/multi-model-server/master/docs/
images/kitten_small.jpg
curl -X POST http://127.0.0.1:8080/predictions/resnet-50_compiled -T kitten_small.jpg
```

Die Ausgabe sollte folgendermaßen aussehen:

```
[
  {
    "probability": 0.6388034820556641,
    "class": "n02123045 tabby, tabby cat"
  },
  {
    "probability": 0.16900072991847992,
    "class": "n02123159 tiger cat"
  },
  {
    "probability": 0.12221276015043259,
    "class": "n02124075 Egyptian cat"
  },
  {
```

```
"probability": 0.028706775978207588,  
"class": "n02127052 lynx, catamount"  
},  
{  
"probability": 0.01915954425930977,  
"class": "n02129604 tiger, Panthera tigris"  
}  
]
```

Führen Sie zur Bereinigung nach dem Test einen Löschbefehl über den RESTful API und beenden Sie den Modellserver mit den folgenden Befehlen:

```
curl -X DELETE http://127.0.0.1:8081/models/resnet-50_compiled  
  
multi-model-server --stop
```

Die Ausgabe sollte folgendermaßen aussehen:

```
{  
  "status": "Model \"resnet-50_compiled\" unregistered"  
}  
Model server stopped.  
Found 1 models and 1 NCGs.  
Unloading 10001 (MODEL_STATUS_STARTED) :: success  
Destroying NCG 1 :: success
```

## Verwenden von PyTorch -Neuron und dem Neuron Compiler AWS

Die PyTorch -Neuron-Kompilierung 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-Dokumentation. PyTorch SDK](#)

## 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 außerdem mit Deep Learning und mit der Verwendung von 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 PyTorch -Neuron-Kompilierungspython, um ein API 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/aws-labs/mxnet-model-server/
master/docs/images/kitten_small.jpg",
                    "./torch_neuron_test/images/kitten_small.jpg")

## Fetch labels to output the top classifications
request.urlretrieve("https://s3.amazonaws.com/deep-learning-models/image-models/
imagenet_class_index.json", "imagenet_class_index.json")
idx2label = []

with open("imagenet_class_index.json", "r") as read_file:
    class_idx = json.load(read_file)
    idx2label = [class_idx[str(k)][1] for k in range(len(class_idx))]

## Import a sample image and normalize it into a tensor
normalize = transforms.Normalize(
    mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225])

eval_dataset = datasets.ImageFolder(
    os.path.dirname("./torch_neuron_test/"),
    transforms.Compose([
        transforms.Resize([224, 224]),
        transforms.ToTensor(),
        normalize,
    ])
)

image, _ = eval_dataset[0]
image = torch.tensor(image.numpy()[np.newaxis, ...])

## Load model
model_neuron = torch.jit.load( 'resnet50_neuron.pt' )

## Predict
results = model_neuron( image )

# Get the top 5 results
top5_idx = results[0].sort()[1][-5:]

# Lookup and print the top 5 labels
top5_labels = [idx2label[idx] for idx in top5_idx]
```

```
print("Top 5 labels:\n {}".format(top5_labels) )
```

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 Tool ARM64 DLAMI

AWS ARM64GPUDLAMIs sind so konzipiert, dass sie eine hohe Leistung und Kosteneffizienz für Deep-Learning-Workloads bieten. Insbesondere der G5G-Instanztyp verfügt über den ARM64-basierten [AWS Graviton2-Prozessor](#), der von Grund auf neu entwickelt wurde und für AWS optimiert wurde, wie Kunden ihre Workloads in der Cloud ausführen. AWS ARM64GPUDLAMIs sind mit Docker, NVIDIA Docker, NVIDIA Driver, CuCUDA, sowie gängigen Frameworks für maschinelles DNN Lernen wie NCCL 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 Beschleunigung bereitzustellen. GPU

### Wählen Sie ein ARM64 DLAMI

Starten Sie eine [G5g-Instance](#) mit ARM64 DLAMI der Ihrer Wahl.

step-by-step Anweisungen zum Starten von finden Sie DLAMI unter [Starten und Konfigurieren von a. DLAMI](#)

Eine Liste der neuesten ARM64 DLAMIs Versionen finden Sie in den [Versionshinweisen für DLAMI](#).

### Erste Schritte

Die folgenden Themen zeigen Ihnen, wie Sie mit der Verwendung von beginnen ARM64DLAMI.

### Inhalt

- [Verwenden von ARM64 GPU PyTorch DLAMI](#)

## Verwenden von ARM64 GPU PyTorch DLAMI

Der AWS Deep Learning AMIs ist sofort einsatzbereit für den Einsatz mit dem Arm64-Prozessor und ist für ihn optimiert. GPUs PyTorch ARM64GPU PyTorch DLAMIDazu gehört eine Python-Umgebung, die mit [PyTorchTorchVision](#), und [TorchServe](#)für Deep-Learning-Training und Inferenz-Anwendungsfälle vorkonfiguriert ist.

### Inhalt

- [PyTorch Python-Umgebung überprüfen](#)
- [Trainingsbeispiel ausführen mit PyTorch](#)
- [Führen Sie das 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 das 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 mithilfe der DLAMI Frameworks und Tools von.

## Inferenz-Tools

- [TensorFlow Servieren](#)

## Modellbereitstellung

Im Folgenden sind die Optionen für die Modellbereitstellung aufgeführt, die auf Deep Learning AMI with 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.

Auf dem `tensorflow-serving-api` ist Deep Learning AMI mit Conda vorinstalliert! Sie finden ein Beispielskript zum Trainieren, Exportieren und Bereitstellen eines MNIST Modells. `~/examples/tensorflow-serving/`

Um eines dieser Beispiele auszuführen, stellen Sie zunächst AMI mit Conda eine Verbindung zu Ihrem Deep Learning 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 müssen ein bereitstellungsfähiges Modell und Client-Skripts bereits auf Ihr DLAMI heruntergeladen werden.

## 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
```

## Trainiere und diene einem MNIST Modell

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 Deep Learning AMI mit Conda vorinstalliert.

Weitere Informationen zur Verwendung TorchServe finden Sie in der Dokumentation zu [Model Server](#). PyTorch

### Topics

Stellen Sie ein Bildklassifizierungsmodell bereit auf TorchServe

Dieses Tutorial zeigt, wie Sie ein Bildklassifizierungsmodell mit bereitstellen TorchServe. Es verwendet ein DenseNet -161-Modell, das von bereitgestellt wird PyTorch. Sobald der Server läuft, wartet er auf Vorhersageanfragen. Wenn Sie ein Bild hochladen, in diesem Fall das Bild eines Kätzchens, gibt der Server eine Vorhersage der fünf am besten passenden Klassen aus den Klassen zurück, für die das Modell trainiert wurde.

Um ein Beispiel für ein Bildklassifizierungsmodell bereitzustellen TorchServe

1. Stellen Sie mit Deep Learning AMI mit Conda v34 oder höher eine Connect zu einer Amazon Elastic Compute Cloud (AmazonEC2) -Instance 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 zurück, die den folgenden fünf wichtigsten Vorhersagen JSON 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 Instance 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.

# Upgraden Ihres DLAMI

Hier finden Sie Informationen zum Upgraden Ihres DLAMI und Tipps zum Aktualisieren der Software auf Ihrem DLAMI.

Halten Sie Ihr Betriebssystem und sonstige installierte Software immer auf dem neuesten Stand, indem Sie Patches und Updates anwenden, sobald diese verfügbar werden.

Wenn Sie Amazon Linux oder Ubuntu verwenden, wenn Sie sich bei DLAMI anmelden, werden Sie benachrichtigt, wenn Aktualisierungen verfügbar sind, und erhalten Anweisungen zum Aktualisieren. Weitere Informationen zur Wartung von Amazon Linux finden Sie unter [Instance-Software aktualisieren](#). Für Ubuntu-Instances lesen Sie in der offiziellen [Ubuntu-Dokumentation](#) nach.

Unter Windows überprüfen Sie Windows Update regelmäßig auf Software- und Sicherheits-Updates. Wenn Sie möchten, können neue Updates auch automatisch angewendet.

## Important

Informationen zu den Sicherheitslücken Meltdown und Spectre und dazu, wie Sie Ihr Betriebssystem patchen können, um diese zu beheben, finden Sie im [Security Bulletin -2018-013](#). AWS

## Themen

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

## Upgraden auf eine neue DLAMI-Version

DLAMIDie System-Images werden regelmäßig aktualisiert, um die Vorteile neuer Deep-Learning-Framework-Versionen CUDA und anderer Softwareupdates sowie Leistungsoptimierungen zu nutzen. Wenn Sie schon seit geraumer Zeit einen DLAMI verwenden und die Vorteile einer Aktualisierung nutzen möchten, müssten Sie eine neue Instance starten. Sie müssten auch manuell Datensätze, Kontrollpunkte oder andere wichtige Daten übertragen. Stattdessen können Sie Amazon verwenden, EBS um Ihre Daten aufzubewahren und sie an neue anzuhängenDLAMI. Auf diese Weise können Sie



häufig Aktualisierungen durchführen und dabei die Zeit, die zum Übertragen Ihrer Daten erforderlich ist, minimieren.

### Note

Wenn Sie EBS Amazon-Volumes anhängen und zwischen diesen verschieben DLAMIs, müssen Sie DLAMIs sowohl das als auch das neue Volume in derselben Availability Zone haben.

1. Verwenden Sie AmazonEC2console, um ein neues EBS Amazon-Volume zu erstellen. Eine detaillierte Anleitung finden Sie unter [Erstellen eines EBS Amazon-Volumes](#).
2. Hängen Sie Ihr neu erstelltes EBS Amazon-Volume an Ihr vorhandenes DLAMI. Eine detaillierte Anleitung finden Sie unter [Anhängen eines EBS Amazon-Volumes](#).
3. Übertragen Sie Ihre Daten, wie z. B. Datensätze, Kontrollpunkte und Konfigurationsdateien.
4. Starten Sie ein DLAMI. Detaillierte Anweisungen finden Sie unter [Eine DLAMI Instanz einrichten](#).
5. Trennen Sie das EBS Amazon-Volume von Ihrem alten DLAMI. Eine detaillierte Anleitung finden Sie unter [Trennen eines EBS Amazon-Volumes](#).
6. Hängen Sie das EBS Amazon-Volume an Ihr neues DLAMI. Folgen Sie den Anweisungen aus Schritt 2 zum Verknüpfen des Volumes.
7. Nachdem Sie überprüft haben, ob Ihre Daten auf dem neuen DLAMI verfügbar sind, stoppen und beenden Sie Ihr altes DLAMI. Detaillierte Anweisungen zur Bereinigung finden Sie unter [Eine Instanz bereinigen DLAMI](#).

## Tipps für Software-Updates

Von Zeit zu Zeit können Sie 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 Deep Learning AMI mit Conda zu aktualisieren. Weitere Anweisungen zum Upgrade und zur Installation finden Sie auf der jeweiligen Framework- oder Software-Website.

### Note

Wir können nicht garantieren, dass ein Paket-Update erfolgreich sein wird. Der Versuch, ein Paket in einer Umgebung mit inkompatiblen Abhängigkeiten zu aktualisieren, kann zu einem

Fehler führen. In einem solchen Fall sollten Sie sich an den Bibliotheksbetreuer wenden, um zu erfahren, ob es möglich ist, die Paketabhängigkeiten zu aktualisieren. Alternativ können Sie versuchen, die Umgebung so zu ändern, dass das Update möglich ist. Diese Änderung wird jedoch wahrscheinlich bedeuten, dass bestehende Pakete entfernt oder aktualisiert werden, was bedeutet, dass wir die Stabilität dieser Umgebung nicht mehr garantieren können.

Das AWS Deep Learning AMIs wird mit vielen Conda-Umgebungen und vielen vorinstallierten Paketen geliefert. Aufgrund der Anzahl der vorinstallierten Pakete ist es schwierig, eine Reihe von Paketen zu finden, die garantiert kompatibel sind. Möglicherweise wird die Warnung „Die Umgebung ist inkonsistent, bitte überprüfen Sie den Paketplan sorgfältig“ angezeigt. DLAMI stellt sicher, dass alle von DLAMI -bereitgestellten Umgebungen korrekt sind, kann jedoch nicht garantieren, dass alle vom Benutzer installierten Pakete korrekt funktionieren.

## Erhalten Sie Benachrichtigungen über neue Updates

### Note

AWS Deep Learning AMIs hat einen wöchentlichen Veröffentlichungsrhythmus für Sicherheitspatches. Für diese inkrementellen Sicherheitspatches werden Versionsbenachrichtigungen gesendet, obwohl sie möglicherweise nicht in den offiziellen Versionshinweisen enthalten sind.

Sie können Benachrichtigungen erhalten, wenn eine neue Version veröffentlicht DLAMI wird. Benachrichtigungen werden SNS unter dem folgenden Thema [bei Amazon](#) veröffentlicht.

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

Nachrichten werden hier veröffentlicht, wenn eine neue veröffentlicht DLAMI wird. Die Version, die Metadaten und die regionalen AMI IDs von AMI werden in der Nachricht enthalten sein.

Diese Nachrichten können mit verschiedenen Methoden empfangen werden. Wir empfehlen Ihnen, die folgende Methode zu verwenden.

1. Öffnen Sie die [SNSAmazon-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 als 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 [AmazonSQS, 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 AMIs

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

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das [Modell der geteilten Verantwortung](#) beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS-Services in der läuft AWS Cloud. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der [AWS](#). Weitere Informationen zu den Compliance-Programmen, die für gelten AWS Deep Learning AMIs, finden Sie unter [AWS Services im Umfang nach Compliance-Programmen AWS](#).
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS-Service, was Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, einschließlich der Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von DLAMI einsetzen können. Die folgenden Themen veranschaulichen, wie Sie DLAMI zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie lernen auch, wie Sie andere verwenden können AWS-Services, die Ihnen bei der Überwachung und Sicherung Ihrer DLAMI Ressourcen helfen.

Weitere Informationen finden Sie unter [Sicherheit bei Amazon EC2](#) im EC2Amazon-Benutzerhandbuch.

## Themen

- [Datenschutz in AWS Deep Learning AMIs](#)
- [Identitäts- und Zugriffsmanagement für AWS Deep Learning AMIs](#)
- [Überprüfung der Einhaltung der Vorschriften für AWS Deep Learning AMIs](#)
- [Widerstandsfähigkeit in AWS Deep Learning AMIs](#)
- [Sicherheit der Infrastruktur in AWS Deep Learning AMIs](#)

- [AWS Deep Learning AMIs Instanzen überwachen](#)

## Datenschutz in AWS Deep Learning AMIs

Das [Modell der AWS gemeinsamen Verantwortung](#) und geteilter Verantwortung gilt für den Datenschutz in AWS Deep Learning AMIs. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie im [Abschnitt Datenschutz FAQ](#). Informationen zum Datenschutz in Europa finden Sie im [AWS Shared Responsibility Model](#) und im GDPR Blogbeitrag auf dem AWS Security Blog.

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre 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 eine Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Einrichtung API und Protokollierung von Benutzeraktivitäten mit AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter [Arbeiten mit CloudTrail Pfaden](#) im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie FIPS 140-3 validierte kryptografische Module für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine benötigen API, verwenden Sie einen Endpunkt. FIPS Weitere Informationen zu den verfügbaren FIPS Endpunkten finden Sie unter [Federal Information Processing Standard](#) ( ) 140-3. FIPS

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

der Konsole arbeiten DLAMI oder sie anderweitig AWS-Services verwenden,API, AWS CLI oder. AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie einem externen Server eine URL zur Verfügung stellen, empfehlen wir dringend, dass Sie keine Anmeldeinformationen in den angebenURL, um Ihre Anfrage an diesen Server zu überprüfen.

## Identitäts- und Zugriffsmanagement für AWS Deep Learning AMIs

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAMAdministratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um DLAMI Ressourcen zu verwenden. IAMist eine AWS-Service , die Sie ohne zusätzliche Kosten verwenden können.

Weitere Informationen zur Identitäts- und Zugriffsverwaltung finden Sie unter [Identitäts- und Zugriffsmanagement für Amazon EC2](#).

Themen

- [Authentifizierung mit Identitäten](#)
- [Verwalten des Zugriffs mit Richtlinien](#)
- [IAMmit 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 übernehmen. 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 (IAMIdentity Center-) Nutzer, 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, mit der Sie Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch signieren können. 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 IAMBenutzerhandbuch unter [AWS Signature Version 4 für API Anfragen](#).

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter [Multi-Faktor-Authentifizierung](#) im AWS IAM Identity Center Benutzerhandbuch und [AWS Multi-Faktor-Authentifizierung IAM 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. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie im Benutzerhandbuch unter [Aufgaben, für die Root-Benutzeranmeldedaten erforderlich](#) sind. IAM

## IAM-Benutzer und -Gruppen

Ein [IAMBenutzer](#) ist eine Identität innerhalb Ihres Unternehmens AWS-Konto, die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wir empfehlen, sich nach Möglichkeit auf temporäre Anmeldeinformationen zu verlassen, anstatt IAM Benutzer mit langfristigen Anmeldeinformationen wie Passwörtern und Zugriffsschlüsseln zu erstellen. Wenn Sie jedoch spezielle Anwendungsfälle haben, für die langfristige Anmeldeinformationen von IAM Benutzern erforderlich sind, empfehlen wir, die Zugriffsschlüssel abwechselnd zu verwenden. Weitere Informationen finden Sie im Benutzerhandbuch unter [Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, für die IAM langfristige Anmeldeinformationen erforderlich](#) sind.

Ein [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 einen Namen geben IAMAdmins und dieser Gruppe Berechtigungen zur Verwaltung von IAM Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie im Benutzerhandbuch unter [Anwendungsfälle für IAM IAM Benutzer](#).

## IAM-Rollen

Eine [IAMRolle](#) ist eine Identität innerhalb von Ihrem AWS-Konto, die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM Rolle in der zu übernehmen AWS Management Console, können Sie [von einem Benutzer zu einer IAM Rolle \(Konsole\) wechseln](#). Sie können eine Rolle übernehmen, indem Sie eine AWS CLI AWS API OR-Operation aufrufen oder eine benutzerdefinierte Operation verwenden URL. Weitere Informationen zu Methoden zur Verwendung von Rollen finden Sie unter [Methoden zur Übernahme einer Rolle](#) im IAM Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- **Verbundbenutzerzugriff** – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie [im IAM Benutzerhandbuch unter Erstellen einer Rolle für einen externen Identitätsanbieter \(Federation\)](#). Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Um zu kontrollieren, 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 verschiedene Berechtigungen für eine bestimmte Aufgabe zu übernehmen.
- **Kontoübergreifender Zugriff**: Sie können eine IAM-Rolle verwenden, um jemandem (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 [IAM Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff](#). IAM

- **Serviceübergreifender Zugriff** — Einige AWS-Services verwenden Funktionen in anderen. AWS-Services Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
- **Zugriffssitzungen weiterleiten (FAS)** — Wenn Sie einen IAM Benutzer oder eine Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Wenn Sie einige Dienste verwenden, führen Sie möglicherweise eine Aktion aus, die dann eine weitere Aktion in einem anderen Dienst auslöst. FASverwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen AWS-Service an nachgeschaltete Dienste zu stellen. FASAnfragen 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 beim Stellen von FAS Anfragen finden Sie unter [Zugriffssitzungen weiterleiten](#).
- **Servicerolle** — Eine Servicerolle ist eine [IAMRolle](#), die ein Dienst übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter [Erstellen einer Rolle zum Delegieren von Berechtigungen AWS-Service an eine](#).
- **Dienstbezogene Rolle** — Eine dienstverknüpfte 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 serviceverknüpfte Rollen anzeigen, aber nicht bearbeiten.
- **Auf Amazon ausgeführte Anwendungen EC2** — Sie können eine IAM Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API Anfragen stellen AWS CLI . Das ist empfehlenswerter, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2 Instance eine AWS Rolle zuzuweisen und sie all 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 im IAMBenutzerhandbuch unter [Verwenden einer IAM](#)

[Rolle, um Berechtigungen für Anwendungen zu erteilen, die auf EC2 Amazon-Instances ausgeführt werden.](#)

## Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS Form von JSON Dokumenten gespeichert. Weitere Informationen zur Struktur und zum Inhalt von JSON Richtliniendokumenten finden Sie im IAMBenutzerhandbuch unter [Überblick über JSON Richtlinien](#).

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Um Benutzern die Erlaubnis zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

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 aus dem AWS Management Console AWS CLI, dem oder dem abrufen AWS API.

### Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind Dokumente mit JSON Berechtigungsrichtlinien, die Sie an eine Identität anhängen können, z. B. an einen IAM Benutzer, eine Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie im Benutzerhandbuch unter [Definieren benutzerdefinierter IAM Berechtigungen mit vom Kunden verwalteten Richtlinien](#). IAM

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 zur Auswahl zwischen einer verwalteten Richtlinie oder einer Inline-Richtlinie finden [Sie im IAMBenutzerhandbuch unter Wählen Sie zwischen verwalteten Richtlinien und Inline-Richtlinien](#).

## Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 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 nicht IAM in einer ressourcenbasierten Richtlinie verwenden.

## Zugriffskontrolllisten (ACLs)

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON Richtliniendokumentformat.

Amazon S3 und AWS WAF Amazon VPC sind Beispiele für Dienste, die Unterstützung bieten ACLs. Weitere Informationen finden Sie unter [Übersicht über ACLs die Zugriffskontrollliste \(ACL\)](#) im Amazon Simple Storage Service Developer Guide.

## Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- **Berechtigungsgrenzen** — Eine Berechtigungsgrenze ist eine erweiterte Funktion, mit der Sie die maximalen Berechtigungen festlegen, die eine identitätsbasierte Richtlinie einer IAM Entität (IAMBenutzer oder Rolle) gewähren 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 zu Berechtigungsgrenzen finden Sie im IAMBenutzerhandbuch unter [Berechtigungsgrenzen für IAM Entitäten](#).

- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen AWS Organizations. AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Geräte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter [Richtlinien zur Servicesteuerung](#) im AWS Organizations Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — RCPs sind JSON Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Sie RCP schränken die Berechtigungen für Ressourcen in Mitgliedskonten ein und können sich auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu OrganizationsRCPs, einschließlich einer Liste AWS-Services dieser Support-LeistungenRCPs, finden Sie unter [Resource Control Policies \(RCPs\)](#) im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter [Sitzungsrichtlinien](#) im Benutzerhandbuch zu IAM.

## 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 IAMBenutzerhandbuch unter [Bewertungslogik für Richtlinien](#).

## IAMmit Amazon EMR

Sie können Amazon verwendenIAM, EMR um Benutzer, AWS Ressourcen, Gruppen, Rollen und Richtlinien zu definieren. Sie können auch steuern, AWS-Services auf welche Benutzer und Rollen Zugriff haben.

Weitere Informationen zur Verwendung IAM mit Amazon EMR finden Sie unter [AWS Identity and Access Management Für Amazon EMR](#).

## Überprüfung der Einhaltung der Vorschriften für AWS Deep Learning AMIs

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

Eine Liste der AWS-Services im Umfang der spezifischen Compliance-Programme enthaltenen [AWS Services finden Sie unter Services im Bereich nach Compliance-Programm AWS](#) . Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#) .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Herunterladen von Berichten in AWS Artifact](#) .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung DLAMI hängt von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften 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 Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- [Bewertung von Ressourcen anhand von AWS Config Regeln](#) im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.

- [AWS Security Hub](#)— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS Ressourcen zu bewerten und Ihre Einhaltung der Sicherheitsstandards und Best Practices der Sicherheitsbranche zu überprüfen.

## Widerstandsfähigkeit in AWS Deep Learning AMIs

Die AWS globale Infrastruktur basiert auf Availability AWS-Regionen Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter [AWS Globale](#) Infrastruktur.

Informationen zu EC2 Amazon-Funktionen zur Unterstützung Ihrer Datenausfallsicherheit und Ihrer Sicherungsanforderungen finden Sie unter [Resilience EC2 in Amazon](#) im EC2Amazon-Benutzerhandbuch.

## Sicherheit der Infrastruktur in AWS Deep Learning AMIs

Die Infrastruktursicherheit von AWS Deep Learning AMIs wird von Amazon unterstützt EC2. Weitere Informationen finden Sie unter [Infrastruktursicherheit in Amazon EC2](#) im EC2Amazon-Benutzerhandbuch.

## AWS Deep Learning AMIs Instanzen überwachen

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer AWS Deep Learning AMIs Instance und Ihrer anderen AWS Lösungen. Ihre DLAMI Instance verfügt über mehrere GPU Überwachungstools, darunter ein Hilfsprogramm, das GPU Nutzungsstatistiken an Amazon meldet CloudWatch. Weitere Informationen finden Sie unter [GPUÜberwachung und Optimierung](#) und unter [Überwachen von EC2 Amazon-Ressourcen](#) im EC2Amazon-Benutzerhandbuch.

## Deaktivierung der Nutzungsverfolgung für Instances DLAMI

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

### Note

AWS sammelt oder speichert keine anderen Informationen über dieDLAMI, wie z. B. die Befehle, die Sie in der DLAMI verwenden.

- Amazon Linux 2
- Amazon Linux 2023
- Ubuntu 20.04
- Ubuntu 22.04

Um die Nutzungsverfolgung zu deaktivieren

Wenn Sie möchten, können Sie die Nutzungsverfolgung für eine neue DLAMI Instanz deaktivieren. Um sich abzumelden, müssen Sie Ihrer EC2 Amazon-Instance beim Start ein Tag hinzufügen. Das Tag sollte den Schlüssel verwenden, dessen OPT\_OUT\_TRACKING zugeordneter Wert auf gesetzt ist `true`. Weitere Informationen finden Sie unter [Taggen Ihrer EC2 Amazon-Ressourcen](#) im EC2Amazon-Benutzerhandbuch.



# DLAMIRahmenunterstützungspolitik

Hier finden Sie Einzelheiten zur Support-Richtlinie für AWS Deep Learning AMIs (DLAMI) - Frameworks.

Eine Liste der DLAMI Frameworks, die AWS derzeit unterstützt werden, finden Sie auf der Seite [DLAMIFramework-Supportrichtlinie](#). Beachten Sie in den Tabellen auf dieser Seite Folgendes:

- Die aktuelle Version gibt die Framework-Version im Format x.y.z an. In diesem Format bezieht sich x auf die Hauptversion, y auf die Nebenversion und z auf die Patch-Version. Beispielsweise ist für TensorFlow 2.10.1 die Hauptversion 2, die Nebenversion 10 und die Patch-Version 1.
- Das Ende des Patches gibt an, wie lange diese Framework-Version AWS unterstützt wird.

Ausführliche Informationen zu bestimmten Themen DLAMIs finden Sie unter [Versionshinweise für DLAMIs](#).

## DLAMIFramework-Unterstützung FAQs

- [Welche Framework-Versionen erhalten Sicherheitspatches?](#)
- [Welche Images werden AWS veröffentlicht, wenn neue Framework-Versionen veröffentlicht werden?](#)
- [Welche Bilder erhalten neue SageMaker AWS KI/Funktionen?](#)
- [Wie ist die aktuelle Version in der Tabelle Unterstützte Frameworks definiert?](#)
- [Was ist, wenn ich eine Version verwende, die nicht in der Tabelle Unterstützte Frameworks aufgeführt ist?](#)
- [Werden frühere Versionen von DLAMIs unterstützt TensorFlow?](#)
- [Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version?](#)
- [Wie oft werden neue Images veröffentlicht?](#)
- [Wird meine Instance 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 auf up-to-date dem Laufenden, wenn Änderungen an Frameworks und ihren Versionen unterstützt werden?](#)
- [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 AMIs Framework-Support-Richtlinien](#) als Unterstützt gekennzeichnet ist, erhält sie Sicherheitspatches.

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

Wir veröffentlichen DLAMIs bald nach der Veröffentlichung neuer Versionen von TensorFlow und PyTorch neue. Dazu gehören Hauptversionen, Haupt-Nebenversionen und major-minor-patch Versionen von Frameworks. Wir aktualisieren auch Images, wenn neue Versionen von Treibern und Bibliotheken verfügbar werden. Weitere Informationen zur Image-Wartung finden Sie unter [Wann endet der aktive Support für meine Framework-Version?](#)

## Welche Bilder erhalten neue SageMaker AWS KI/Funktionen?

Neue Funktionen werden normalerweise in der neuesten Version von DLAMIs for PyTorch und TensorFlow veröffentlicht. Einzelheiten zu neuen SageMaker KI oder AWS Funktionen finden Sie in den Versionshinweisen für ein bestimmtes Bild. Eine Liste der verfügbaren DLAMIs Versionen finden Sie in den [Versionshinweisen für DLAMI](#). Weitere Informationen zur Image-Wartung finden Sie unter [Wann endet der aktive Support für meine Framework-Version?](#)

## Wie ist die aktuelle Version in der Tabelle Unterstützte Frameworks definiert?

Die aktuelle Version in der [Tabelle AWS Deep Learning AMIs 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 inDLAMI. 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 AMIs 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, dass Sie ein Upgrade auf eines der unterstützten Frameworks durchführen und dabei die neueste Version DLAMI Ihrer Wahl verwenden. Eine Liste der verfügbaren DLAMIs Versionen finden Sie in den [Versionshinweisen für DLAMI](#).

## Werden frühere Versionen von DLAMIs unterstützt 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 AMIs 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 eine DLAMI mit der neuesten Framework-Version zu verwenden, rufen Sie die [DLAMIID](#) ab und verwenden Sie sie, um das DLAMI über die [EC2Konsole](#) zu starten. AWS CLI Beispielfehle zum Abrufen der AWS Deep Learning AMIs ID finden Sie auf der Seite mit den DLAMI Versionshinweisen zu den [einzelnen DLAMI Framework-Versionshinweisen](#). Die von Ihnen gewählte Framework-Version muss in der [Tabelle AWS Deep Learning AMIs 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 bestehende Version von mit einer neuen Version von veröffentlicht TensorFlow wird CUDA, veröffentlichen wir eine neue DLAMI Version von TensorFlow mit Unterstützung für die neue CUDA Version von.

## Wird meine Instance gepatcht, während mein Workload läuft?

Nein. Patch-Updates für DLAMI sind keine „direkten“ Updates.

Sie müssen eine neue EC2 Instanz einschalten, Ihre Workloads und Skripts migrieren und dann Ihre vorherige Instanz ausschalten.

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

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 Versionen 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. Achten Sie darauf, in den [Versionshinweisen](#) nach aktualisierten Abhängigkeitsinformationen zu suchen. DLAMI

## Wann endet der aktive Support für meine Framework-Version?

DLAMIBilder sind unveränderlich. Sobald sie erstellt wurden, ändern sie sich nicht. Es gibt vier Hauptgründe, warum der aktive Support für eine Framework-Version endet:

- [Upgrades der Framework-Version \(Patch\)](#)
- [AWS Sicherheitspatches](#)
- [Ende des Patch-Datums \(Alterung abgelaufen\)](#)
- [Abhängigkeit end-of-support](#)

### Note

Aufgrund der Häufigkeit von Versionspatch-Upgrades und Sicherheitspatches empfehlen wir, regelmäßig auf der Seite mit den Versionshinweisen DLAMI nachzuschauen 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 er DLAMI mit TensorFlow 2.7.1 einen neuen. Die vorherigen Images mit 2.7.0 werden nicht mehr aktiv verwaltet, sobald das neue Image mit TensorFlow 2.7.1 veröffentlicht wird. Das DLAMI mit TensorFlow 2.7.0 erhält keine weiteren Patches. Die Seite mit den DLAMI Versionshinweisen für TensorFlow 2.7 wird dann mit den neuesten Informationen aktualisiert. Es gibt keine eigene Seite mit den Versionshinweisen für jeden kleineren Patch.

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

## AWS Sicherheitspatches

Wenn Sie einen Workload haben, der auf einem Image mit TensorFlow 2.7.0 basiert und AWS Sie einen Sicherheitspatch erstellen, dann DLAMI wird eine neue Version von für TensorFlow 2.7.0 veröffentlicht. Die vorherige Version der Images mit TensorFlow 2.7.0 wird nicht mehr aktiv gepflegt. Weitere Informationen finden Sie unter Anweisungen [Wird meine Instance gepatcht, während mein Workload läuft?](#) zur Suche nach der neuesten Version finden Sie DLAMI unter [Wie finde ich das neueste gepatchte Image für eine unterstützte Framework-Version?](#)

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

## Ende des Patch-Datums (Alterung abgelaufen)

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

Wenn bei [Multi-Frameworks DLAMIs](#) eine der Framework-Versionen aktualisiert wird, ist eine neue Version DLAMI mit der aktualisierten Version erforderlich. Die Version 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 DLAMI 2.7.0-Image mit Python 3.6 ausführen und für diese Version von Python 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, nicht mehr end-of-support aktiv verwaltet, wenn eine Betriebssystemversion wie Ubuntu 16.04 markiert ist.

## Werden Images mit Framework-Versionen, die nicht mehr aktiv verwaltet werden, gepatcht?

Nein. Für Bilder, die nicht mehr aktiv gepflegt werden, wird es keine Neuveröffentlichungen geben.

## Wie verwende ich eine ältere Framework-Version?

Um eine DLAMI mit einer älteren Framework-Version zu verwenden, rufen Sie die [DLAMIID](#) ab und verwenden Sie sie, um die DLAMI über die [EC2Konsole](#) zu starten. AWS CLIBefehle zum Abrufen der AMI ID finden Sie auf der Seite mit den Versionshinweisen in den [DLAMISingle-Framework-Versionshinweisen](#).

## Wie bleibe ich auf up-to-date dem Laufenden, wenn Änderungen an Frameworks und ihren Versionen unterstützt werden?

Bleiben Sie up-to-date bei DLAMI Frameworks und Versionen, indem Sie die [Tabelle mit den AWS Deep Learning AMIs Framework-Supportrichtlinien](#) und den [DLAMIVersionshinweisen](#) verwenden.

## Benötige ich eine kommerzielle Lizenz, um das Anaconda Repository nutzen zu können?

Anaconda hat für bestimmte Benutzer auf ein kommerzielles Lizenzmodell umgestellt. Aktiv gepflegt DLAMIs wurden vom Anaconda-Kanal auf die öffentlich verfügbare Open-Source-Version von Conda ([Conda-Forge](#)) migriert.

# Wichtige NVIDIA Treiberänderungen für DLAMIs

Am 15. November 2023 AWS wurden wichtige Änderungen an AWS Deep Learning AMIs (DLAMI) vorgenommen, die sich auf den NVIDIA Treiber beziehen, den ich DLAMIs verwende. Informationen darüber, was sich geändert hat und ob sich dies auf Ihre Verwendung von DLAMIs auswirkt, finden Sie unter [DLAMINVIDIATreiberwechsel FAQs](#).

## DLAMINVIDIATreiberwechsel FAQs

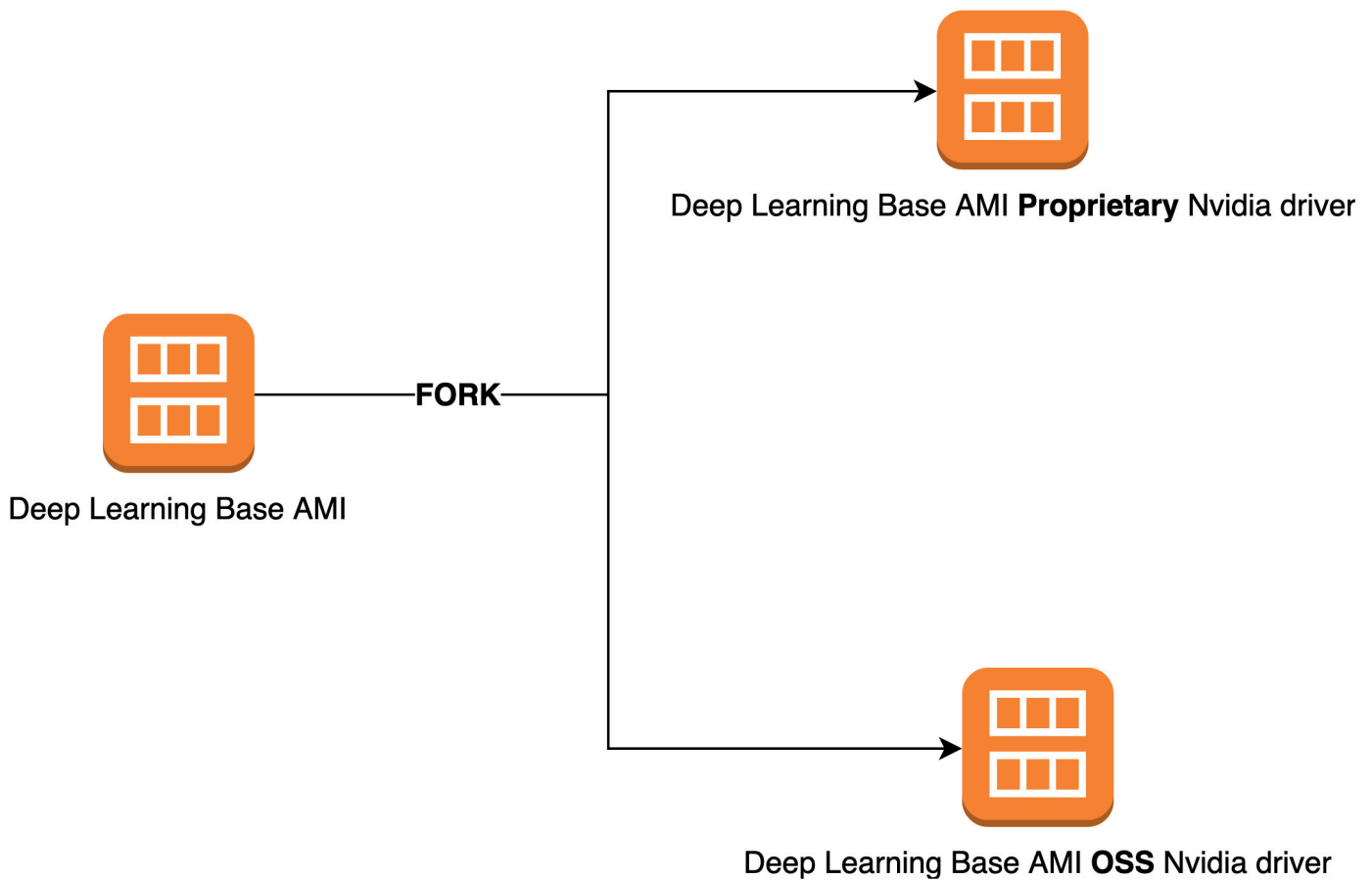
- [Was hat sich geändert?](#)
- [Warum war diese Änderung erforderlich?](#)
- [DLAMIs Worauf hat sich diese Änderung ausgewirkt?](#)
- [Was bedeutet das für dich?](#)
- [Gibt es bei der neueren Version einen Verlust an Funktionalität? DLAMIs](#)
- [Hatte diese Änderung Auswirkungen auf Deep Learning Containers?](#)

## Was hat sich geändert?

Wir haben uns DLAMIs in zwei separate Gruppen aufgeteilt:

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

Aus diesem Grund haben wir DLAMIs für jede der beiden Kategorien neu erstellt, mit neuen Namen und neuen AMI IDs. Diese DLAMIs sind nicht austauschbar. Das heißt, DLAMIs von einer Gruppe werden keine Instances unterstützt, die von der anderen Gruppe unterstützt werden. Beispielsweise unterstützt derjenige DLAMI, der P5 unterstützt, G3 nicht, und derjenige, der G3 unterstützt DLAMI, unterstützt P5 nicht.



## Warum war diese Änderung erforderlich?

Zuvor NVIDIA GPUs enthielt DLAMIs for einen proprietären Kernel-Treiber von NVIDIA. Die Upstream-Linux-Kernel-Community akzeptierte jedoch eine Änderung, mit der proprietäre Kerneltreiber, wie der Treiber, von der NVIDIA GPU Kommunikation mit anderen Kerneltreibern isoliert werden. Diese Änderung deaktiviert Instances GPUDirect RDMA der Serien P4 und P5. Dies ist der Mechanismus, der eine effiziente Nutzung EFA für GPUs verteilte Schulungen ermöglicht. Verwenden Sie daher DLAMIs jetzt den OpenRM-Treiber (Open-Source-Treiber), der mit den NVIDIA EFA Open-Source-Treibern verknüpft ist, um G4dn, G5, P4 und P5 zu unterstützen. Dieser OpenRM-Treiber unterstützt jedoch keine älteren Instanzen (wie P3 und G3). Um sicherzustellen, dass wir weiterhin aktuelle, performante und sichere Produkte anbieten, DLAMIs die beide Instanztypen unterstützen, haben wir uns DLAMIs in zwei Gruppen aufgeteilt: eine mit dem OpenRM-Treiber (der G4dn, G5, P4 und P5 unterstützt) und eine mit dem älteren proprietären Treiber (der P3, P3dn und G3 unterstützt).

## DLAMIs Worauf hat sich diese Änderung ausgewirkt?

Diese Änderung betraf alle DLAMIs.

## Was bedeutet das für dich?

Alle bieten DLAMIs weiterhin Funktionalität, Leistung und Sicherheit, solange Sie sie auf einem unterstützten Amazon Elastic Compute Cloud (Amazon EC2) Instance-Typ ausführen. Um zu ermitteln, welche EC2 Instance-Typen von a DLAMI unterstützt werden, lesen Sie die entsprechenden Versionshinweise und suchen Sie dann nach Unterstützten EC2 Instances. DLAMI Eine Liste der derzeit unterstützten DLAMI Optionen und Links zu den jeweiligen Versionshinweisen finden Sie unter [Versionshinweise für DLAMIs](#).

Außerdem müssen Sie die richtigen Befehle AWS Command Line Interface (AWS CLI) verwenden, um die aktuelle DLAMIs Version aufzurufen.

Verwenden Sie für BaseDLAMIs, die P3, P3dn und G3 unterstützen, diesen Befehl:

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

Verwenden Sie für BasenDLAMIs, die G4dn, G5, P4 und P5 unterstützen, diesen Befehl:

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

## Gibt es bei der neueren Version einen Verlust an Funktionalität? DLAMIs

Nein, es gibt keinen Funktionsverlust. Die aktuellen DLAMIs Versionen bieten die gesamte Funktionalität, Leistung und Sicherheit der Vorgängerversionen DLAMIs, sofern Sie sie auf einem unterstützten EC2 Instance-Typ ausführen.



## Hatte diese Änderung Auswirkungen auf Deep Learning Containers?

Nein, diese Änderung hatte keine Auswirkungen auf AWS Deep Learning Containers, da sie den NVIDIA Treiber nicht enthalten. Stellen Sie jedoch sicher, dass Sie Deep Learning Containers auf Geräten ausführen AMIs, die mit den zugrunde liegenden Instances kompatibel sind.

## Verwandte Informationen über DLAMI

Weitere Ressourcen mit verwandten Informationen finden Sie DLAMI außerhalb des AWS Deep Learning AMIs Entwicklerhandbuchs. Schauen Sie AWS re:Post sich unter Fragen DLAMI anderer Kunden an oder stellen Sie Ihre eigenen Fragen. Lesen Sie im AWS Machine Learning Blog und anderen AWS Blogs offizielle Beiträge überDLAMI.

AWS re:Post

[Schlagwort: AWS Deep Learning AMIs](#)

AWS Blog

- [AWS Blog Machine Learning | Kategorie: AWS Deep Learning AMIs](#)
- [AWS Blog Machine Learning | Schnelleres Training mit optimiertem TensorFlow 1.6 auf Amazon EC2 C5- und P3-Instances](#)
- [AWS Blog Machine Learning | Neu AWS Deep Learning AMIs für Praktiker des Machine Learning](#)
- [AWS Partner Network \(APN\) Blog | Neue Schulungskurse verfügbar: Einführung in Machine Learning und Deep Learning am AWS](#)
- [AWS Nachrichtenblog | Eine Reise ins Deep Learning mit AWS](#)

## Veraltete Funktionen von DLAMI

In der folgenden Tabelle sind die veralteten Funktionen von AWS Deep Learning AMIs (DLAMI), das Datum, an dem wir sie eingestellt haben, und Einzelheiten darüber aufgeführt, warum wir sie als veraltet eingestuft haben.

Funktion	Datum	Details
Ubuntu 16.04	10.07.2021	Ubuntu Linux 16.04 LTS hat am 30. April 2021 das Ende seines LTS Fünfjahresfensters erreicht und wird von seinem Anbieter nicht mehr unterstützt. Ab Oktober 2021 gibt es in neuen Versionen keine Updates mehr für die Deep Learning Base AMI (Ubuntu 16.04). Frühere Versionen werden weiterhin verfügbar sein.
Amazon Linux	10.07.2021	Amazon Linux ist <a href="#">end-of-life</a> ab Dezember 2020 verfügbar. Ab Oktober 2021 gibt es in neuen Versionen keine Updates mehr für Deep Learning AMI (Amazon Linux). Frühere Versionen von Deep Learning AMI (Amazon Linux) werden weiterhin verfügbar sein.
Chainer	01.07.2020	Chainer hat <a href="#">das Ende der Hauptversionen</a> ab Dezember 2019 angekündigt.

Funktion	Datum	Details
		<p>gt. Folglich werden ab Juli 2020 keine Chainer Conda-Umgebungen mehr im DLAMI einbezogen. Frühere Versionen von DLAMI, die diese Umgebungen enthalten, sind weiterhin verfügbar. 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 Kundenanfragen wechseln wir für neue TF/MX/PT Versionen zu Python 3.7.
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 Communitys haben außerdem angekündigt, dass die Versionen TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4 und MXNet 1.6.0 die letzten sein werden, die Python 2 unterstützen.</p>

# Dokumentverlauf für DLAMI

Die folgende Tabelle enthält einen Überblick über die neuesten DLAMI Versionen und die damit verbundenen Änderungen am AWS Deep Learning AMIs Entwicklerhandbuch.

## Letzte Änderungen

Änderung	Beschreibung	Datum
<a href="#">ARM64 DLAMI</a>	Das unterstützt AWS Deep Learning AMIs jetzt Bilder auf Basis eines Arm64-ProzessorsGPUs.	29. November 2021
<a href="#">TensorFlow 2</a>	Das Deep Learning AMI mit Conda umfasst jetzt 2 mit 10. TensorFlow CUDA	3. Dezember 2019
<a href="#">AWS Inferentia</a>	Das Deep Learning unterstützt AMI jetzt die AWS Inferentia-Hardware und das Neuron. AWS SDK	3. Dezember 2019
<a href="#">TensorFlow Serving mit einem Inception-Modell verwenden</a>	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
<a href="#">Von einem Nightly PyTorch Build aus installieren</a>	Es wurde ein Tutorial hinzugefügt, das erklärt, wie Sie AMI mit Conda einen Nightly-Build von deinstallieren PyTorch und anschließend PyTorch auf Ihrem Deep Learning installieren können.	25. September 2018

[Conda-Tutorial](#)

Das Beispiel MOTD wurde aktualisiert, um eine neuere Version widerzuspiegeln.

23. Juli 2018

## Frühere Änderungen

Die folgende Tabelle enthält eine Historie früherer DLAMI Versionen und verwandter Änderungen vor Juli 2018.

Änderung	Beschreibung	Datum
TensorFlow mit Horovod	Es wurde ein Tutorial für das Training ImageNet mit TensorFlow und Horovod hinzugefügt.	6. Juni 2018
Aktualisierungsanleitung	Aktualisierungsanleitung wurde hinzugefügt.	15. Mai 2018
Neue Regionen und neues 10-Minuten-Tutorial	Neue Regionen hinzugefügt: US West (Nordkalifornien), Südamerika, Kanada (Zentral), EU (London) und EU (Paris). Außerdem die erste Version eines 10-minütigen Tutorials mit dem Titel: „Erste Schritte mit Deep LearningAMI“.	26. April 2018
Chainer-Tutorial	Ein Tutorial zur Verwendung von Chainer in Multi-GPUGPU, Single- und CPU Modi wurde hinzugefügt. CUDADie Integration wurde für mehrere Frameworks von CUDA 8 auf CUDA 9 aktualisiert.	28. Februar 2018

Änderung	Beschreibung	Datum
Linux AMIs v3.0 sowie Einführung von MXNet Model Server, TensorFlow Serving und TensorBoard	Es wurden Tutorials für Conda AMIs mit neuen Funktionen zur Bereitstellung von Modellen und Visualisierungen unter Verwendung von MXNet Model Server v0.1.5, TensorFlow Serving v1.4.0 und v0.4.0 hinzugefügt. TensorBoard AMI und die in Conda und Übersichten beschriebenen CUDA Framework-Funktionen. CUDA Neueste Versionshinweise nach <a href="https://aws.amazon.com/releases/notes/">https://aws.amazon.com/releases/notes/</a> verschoben	25. Januar 2018
Linux v2.0 AMIs	Base, Source und Conda AMIs wurden mit NCCL 2.1 aktualisiert. Source und Conda AMIs wurden mit MXNet v1.0, PyTorch 0.3.0 und Keras 2.0.9 aktualisiert.	11. Dezember 2017
AMIZwei Windows-Optionen hinzugefügt	Windows 2012 R2 und 2016 AMIs veröffentlicht: zur AMI Auswahlhilfe hinzugefügt und zu den Versionshinweisen hinzugefügt.	30. November 2017
Erste Dokumentationsversion	Detaillierte Beschreibung der Änderung mit Link zum geänderten Thema/Abschnitt.	15. November 2017

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.