



Panduan Developer

AWS Deep Learning AMIs



AWS Deep Learning AMIs: Panduan Developer

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu DLAMI?	1
Tentang panduan ini	1
Prasyarat	1
Contoh kasus penggunaan	1
Fitur	2
Kerangka kerja terinstal	2
Perangkat lunak GPU yang sudah diinstal sebelumnya	3
Penyajian model dan visualisasi	3
Catatan rilis untuk DLAMIs	4
Basis DLAMIs	4
Kerangka tunggal DLAMIs	5
Multi-kerangka DLAMIs	6
Memulai	7
Memilih DLAMI	7
Instalasi CUDA dan Binding Kerangka Kerja	8
Basis	9
Conda	9
Arsitektur	11
OS	11
Memilih sebuah instance	11
Harga	13
Ketersediaan Wilayah	13
GPU	14
CPU	15
Inferensia	15
Trainium	16
Pengaturan	17
Menemukan ID DLAMI	17
Meluncurkan sebuah instance	19
Menghubungkan ke sebuah instans	21
Menyiapkan Jupyter	21
Mengamankan server	22
Memulai server	23
Menghubungkan klien	23

Masuk	25
Membersihkan	26
Menggunakan DLAMI	28
DLAMI Conda	28
Pengantar AMI Pembelajaran Mendalam dengan Conda	28
Masuk ke DLAMI Anda	29
Mulai TensorFlow Lingkungan	29
Beralih ke Lingkungan PyTorch Python 3	30
Menghapus Lingkungan	31
DLAMI Dasar	31
Menggunakan Basis Pembelajaran Mendalam AMI	31
Mengkonfigurasi Versi CUDA	32
Notebook Jupyter	32
Menavigasi Tutorial yang Diinstal	33
Beralih Lingkungan dengan Jupyter	34
Tutorial	34
Mengaktifkan Kerangka Kerja	35
Elastic Fabric Adapter	38
Pemantauan dan Optimasi GPU	51
AWS Inferensia	61
ARM64 DLAMI	83
Inferensi	86
Penyajian Model	86
Meningkatkan DLAMI Anda	91
Peningkatan DLAMI	91
Pembaruan Perangkat Lunak	92
Pemberitahuan Rilis	93
Keamanan	95
Perlindungan data	96
Manajemen identitas dan akses	97
Mengautentikasi dengan identitas	97
Mengelola akses menggunakan kebijakan	100
IAM dengan Amazon EMR	103
Validasi kepatuhan	103
Ketahanan	104
Keamanan infrastruktur	105

Pemantauan	105
Pelacakan penggunaan	105
Kebijakan dukungan kerangka kerja	107
Dukungan kerangka kerja DLAMI FAQs	107
Versi kerangka kerja apa yang mendapatkan tambalan keamanan?	108
Gambar apa yang AWS dipublikasikan saat versi kerangka kerja baru dirilis?	108
Gambar apa yang mendapatkan AWS fitur SageMaker AI/baru?	108
Bagaimana versi saat ini didefinisikan dalam tabel Kerangka Kerja yang Didukung?	108
Bagaimana jika saya menjalankan versi yang tidak ada dalam tabel Kerangka Kerja yang Didukung?	108
Apakah DLAMIs mendukung versi sebelumnya TensorFlow?	109
Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?	109
Seberapa sering gambar baru dirilis?	109
Apakah instance saya akan ditambah di tempat saat beban kerja saya berjalan?	109
Apa yang terjadi ketika versi kerangka kerja baru yang ditambah atau diperbarui tersedia? ..	110
Apakah dependensi diperbarui tanpa mengubah versi kerangka kerja?	110
Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?	110
Akankah gambar dengan versi kerangka kerja yang tidak lagi dipelihara secara aktif ditambah?	112
Bagaimana cara menggunakan versi kerangka kerja yang lebih lama?	112
Bagaimana cara saya tetap up-to-date dengan perubahan dukungan dalam kerangka kerja dan versinya?	112
Apakah saya memerlukan lisensi komersial untuk menggunakan Repositori Anaconda?	112
Perubahan penting	113
Perubahan driver DLAMI NVIDIA FAQs	113
Apa yang berubah?	113
Mengapa perubahan ini diperlukan?	114
DLAMIs Apa yang mempengaruhi perubahan ini?	115
Apa artinya ini bagi Anda?	115
Apakah ada kehilangan fungsionalitas dengan yang lebih baru? DLAMIs	115
Apakah perubahan ini memengaruhi Deep Learning Containers?	115
Informasi terkait	116
Fitur usang	117
Riwayat dokumen	119
.....	cxxii

Apa itu AWS Deep Learning AMIs?

AWS Deep Learning AMIs (DLAMI) menyediakan gambar mesin khusus yang dapat Anda gunakan untuk pembelajaran mendalam di cloud. Sebagian besar DLAMIs tersedia Wilayah AWS untuk berbagai jenis instans Amazon Elastic Compute Cloud (Amazon EC2), dari instans kecil khusus CPU hingga instans multi-GPU berdaya tinggi terbaru. DLAMIs Datang dikonfigurasi sebelumnya dengan [NVIDIA CUDA](#) dan [NVIDIA cuDNN dan rilis terbaru dari kerangka](#) pembelajaran mendalam yang paling populer.

Tentang panduan ini

Konten di dapat membantu Anda meluncurkan dan menggunakan DLAMIs. Panduan ini mencakup beberapa kasus penggunaan pembelajaran mendalam yang umum, baik untuk pelatihan maupun inferensi. Ini juga mencakup cara memilih AMI yang tepat untuk tujuan Anda dan jenis contoh yang mungkin Anda sukai.

Selain itu, DLAMIs termasuk beberapa tutorial yang disediakan oleh kerangka kerja yang didukung mereka. Panduan ini dapat menunjukkan cara mengaktifkan setiap kerangka kerja dan menemukan tutorial yang sesuai untuk memulai. Ini juga memiliki tutorial tentang pelatihan terdistribusi, debugging, menggunakan AWS Inferentia dan AWS Trainium, dan konsep kunci lainnya. Untuk petunjuk tentang cara mengatur server notebook Jupyter untuk menjalankan tutorial di browser Anda, lihat [Menyiapkan server Jupyter Notebook pada instance DLAMI](#)

Prasyarat

Agar berhasil menjalankan DLAMIs, kami sarankan Anda terbiasa dengan alat baris perintah dan Python dasar.

Contoh kasus penggunaan DLAMI

Berikut ini adalah contoh dari beberapa kasus penggunaan umum untuk AWS Deep Learning AMIs (DLAMI).

Belajar tentang pembelajaran mendalam — DLAMI adalah pilihan tepat untuk belajar atau mengajar pembelajaran mesin dan kerangka pembelajaran mendalam. DLAMIs Menghilangkan sakit kepala dari pemecahan masalah instalasi setiap kerangka kerja dan membuat mereka bermain bersama di komputer yang sama. DLAMIs Termasuk notebook Jupyter dan membuatnya mudah untuk

menjalankan tutorial yang disediakan kerangka kerja bagi orang-orang yang baru mengenal pembelajaran mesin dan pembelajaran mendalam.

Pengembangan aplikasi — Jika Anda seorang pengembang aplikasi yang tertarik menggunakan pembelajaran mendalam untuk membuat aplikasi Anda memanfaatkan kemajuan terbaru dalam AI, maka DLAMI adalah tempat uji yang sempurna untuk Anda. Setiap kerangka kerja dilengkapi dengan tutorial tentang cara memulai pembelajaran mendalam, dan banyak dari mereka memiliki kebun binatang model yang membuatnya mudah untuk mencoba pembelajaran mendalam tanpa harus membuat jaringan saraf sendiri atau melakukan pelatihan model apa pun. Beberapa contoh menunjukkan cara membuat aplikasi deteksi gambar hanya dalam beberapa menit, atau cara membuat aplikasi pengenalan suara untuk chatbot Anda sendiri.

Pembelajaran mesin dan analitik data — Jika Anda seorang ilmuwan data atau Anda tertarik untuk memproses data Anda dengan pembelajaran mendalam, maka Anda akan menemukan bahwa banyak kerangka kerja memiliki dukungan untuk R dan Spark. Anda akan menemukan tutorial tentang cara melakukan regresi sederhana, hingga membangun sistem pemrosesan data yang dapat diskalakan untuk sistem personalisasi dan prediksi.

Penelitian — Jika Anda seorang peneliti yang ingin mencoba kerangka kerja baru, menguji model baru, atau melatih model baru, maka DLAMI AWS dan kemampuan untuk skala dapat mengurangi rasa sakit instalasi yang membosankan dan pengelolaan beberapa node pelatihan.

Note

Meskipun pilihan awal Anda mungkin memutakhirkan jenis instans Anda ke instans yang lebih besar dengan lebih banyak GPUs (hingga 8), Anda juga dapat menskalakan secara horizontal dengan membuat kluster instans DLAMI. Lihat informasi [Informasi terkait DLAMI](#) lebih lanjut tentang build cluster.

Fitur DLAMI

Fitur AWS Deep Learning AMIs (DLAMI) termasuk kerangka kerja pembelajaran mendalam yang telah diinstal sebelumnya, perangkat lunak GPU, server model, dan alat visualisasi model.

Kerangka kerja terinstal

Saat ini ada dua rasa utama DLAMI dengan variasi lain yang terkait dengan sistem operasi (OS) dan versi perangkat lunak:

- [Pembelajaran Mendalam AMI dengan Conda](#)— Kerangka kerja diinstal secara terpisah menggunakan conda paket dan lingkungan Python terpisah.
- [Dasar Pembelajaran Mendalam AMI](#)— Tidak ada kerangka kerja yang diinstal; hanya [NVIDIA CUDA dan dependensi](#) lainnya.

AMI Pembelajaran Mendalam dengan Conda menggunakan conda lingkungan untuk mengisolasi setiap kerangka kerja, sehingga Anda dapat beralih di antara mereka sesuka hati dan tidak khawatir tentang dependensinya yang bertentangan. AMI Pembelajaran Mendalam dengan Conda mendukung kerangka kerja berikut:

- PyTorch
- TensorFlow 2

Note

DLAMI tidak lagi mendukung kerangka pembelajaran mendalam berikut: Apache, Microsoft Cognitive Toolkit (CNTK) MXNet, Caffe, Caffe2, Theano, Chainer, dan Keras.

Perangkat lunak GPU yang sudah diinstal sebelumnya

[Bahkan jika Anda menggunakan instance khusus CPU, DLAMIs akan memiliki NVIDIA CUDA dan NVIDIA cuDNN.](#) Perangkat lunak yang diinstal adalah sama terlepas dari jenis instancenya. Perlu diingat bahwa alat khusus GPU hanya berfungsi pada instance yang memiliki setidaknya satu GPU. Untuk informasi selengkapnya tentang jenis instance, lihat [Memilih tipe instans DLAMI](#).

Untuk informasi lebih lanjut tentang CUDA, lihat [Instalasi CUDA dan Binding Kerangka Kerja](#).

Penyajian model dan visualisasi

Deep Learning AMI with Conda sudah diinstal sebelumnya dengan server model untuk TensorFlow, serta TensorBoard untuk visualisasi model. Untuk informasi selengkapnya, lihat [TensorFlow Melayani](#).

Catatan rilis untuk DLAMIs

Di sini Anda dapat menemukan catatan rilis terperinci untuk semua opsi yang saat ini didukung AWS Deep Learning AMIs (DLAMI).

[Untuk catatan rilis untuk kerangka kerja DLAMI yang tidak lagi kami dukung, lihat bagian Unsupported Framework Release Notes Archive pada halaman DLAMI Framework Support Policy.](#)

Note

AWS Deep Learning AMIs Memiliki irama rilis malam untuk patch keamanan. Kami tidak menyertakan patch keamanan tambahan ini dalam catatan rilis resmi.

Basis DLAMIs

GPU

- X86
 - [AWS Basis Pembelajaran Mendalam AMI \(Amazon Linux 2023\)](#)
 - [AWS AMI Dasar Pembelajaran Mendalam \(Ubuntu 22.04\)](#)
 - [AWS AMI Dasar Pembelajaran Mendalam \(Ubuntu 20.04\)](#)
 - [AWS Basis Pembelajaran Mendalam AMI \(Amazon Linux 2\)](#)
- ARM64
 - [AWS ARM64 AMI Dasar Pembelajaran Mendalam \(Ubuntu 22.04\)](#)
 - [AWS Basis Pembelajaran Mendalam ARM64 AMI \(Amazon Linux 2\)](#)
 - [AWS Basis Pembelajaran Mendalam ARM64 AMI \(Amazon Linux 2023\)](#)

Qualcomm

- X86
 - [AWS Basis Pembelajaran Mendalam Qualcomm AMI \(Amazon Linux 2\)](#)

AWS Neuron

- Lihat Panduan [DLAMI Neuron](#)

Kerangka tunggal DLAMIs

PyTorch-spesifik AMIs

GPU

- X86
 - [AWS Pembelajaran Mendalam AMI GPU PyTorch 2.6 \(Amazon Linux 2023\)](#)
 - [AWS Pembelajaran Mendalam AMI GPU PyTorch 2.6 \(Ubuntu 22.04\)](#)
 - [AWS Pembelajaran Mendalam AMI GPU PyTorch 2.5 \(Amazon Linux 2023\)](#)
 - [AWS Pembelajaran Mendalam AMI GPU PyTorch 2.5 \(Ubuntu 22.04\)](#)
 - [AWS Pembelajaran Mendalam AMI GPU PyTorch 2.4 \(Ubuntu 22.04\)](#)
 - [AWS Pembelajaran Mendalam AMI GPU PyTorch 2.3 \(Ubuntu 20.04\)](#)
 - [AWS Pembelajaran Mendalam AMI GPU PyTorch 2.3 \(Amazon Linux 2\)](#)
- ARM64
 - [AWS Pembelajaran Mendalam ARM64 AMI GPU PyTorch 2.6 \(Amazon Linux 2023\)](#)
 - [AWS Pembelajaran Mendalam ARM64 AMI GPU PyTorch 2.6 \(Ubuntu 22.04\)](#)
 - [AWS Pembelajaran Mendalam ARM64 AMI GPU PyTorch 2.5 \(Ubuntu 22.04\)](#)
 - [AWS Pembelajaran Mendalam ARM64 AMI GPU PyTorch 2.4 \(Ubuntu 22.04\)](#)
 - [AWS Pembelajaran Mendalam ARM64 AMI GPU PyTorch 2.3 \(Ubuntu 22.04\)](#)

AWS Neuron

- Lihat Panduan [DLAMI Neuron](#)

TensorFlow-spesifik AMIs

GPU

- X86
 - [AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.18 \(Amazon Linux 2023\)](#)
 - [AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.18 \(Ubuntu 22.04\)](#)

- [AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.17 \(Ubuntu 22.04\)](#)
- [AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.16 \(Amazon Linux 2\)](#)
- [AWS Pembelajaran Mendalam AMI GPU TensorFlow 2.16 \(Ubuntu 20.04\)](#)

AWS Neuron

- Lihat Panduan [DLAMI Neuron](#)

Multi-kerangka DLAMIs

Tip

Jika Anda hanya menggunakan satu kerangka pembelajaran mesin, maka kami merekomendasikan [DLAMI kerangka tunggal](#).

GPU

- X86
 - [AWS Pembelajaran Mendalam AMI \(Amazon Linux 2\)](#)

AWS Neuron

- Lihat Panduan [DLAMI Neuron](#)

Memulai dengan DLAMI

Panduan ini mencakup tips tentang memilih DLAMI yang tepat untuk Anda, memilih jenis instans yang sesuai dengan kasus penggunaan dan anggaran Anda, [Informasi terkait DLAMI](#) dan yang menjelaskan pengaturan khusus yang mungkin menarik.

Jika Anda baru menggunakan AWS atau menggunakan Amazon EC2, mulailah dengan [Pembelajaran Mendalam AMI dengan Conda](#). Jika Anda terbiasa dengan Amazon EC2 dan AWS layanan lain seperti Amazon EMR, Amazon EFS, atau Amazon S3, dan tertarik untuk mengintegrasikan layanan tersebut untuk proyek yang memerlukan pelatihan atau inferensi terdistribusi, [Informasi terkait DLAMI](#) periksa untuk melihat apakah sesuai dengan kasus penggunaan Anda.

Kami menyarankan Anda memeriksa [Memilih DLAMI](#) untuk mendapatkan gambaran tentang jenis instans mana yang terbaik untuk aplikasi Anda.

Langkah selanjutnya

[Memilih DLAMI](#)

Memilih DLAMI

Kami menawarkan berbagai opsi DLAMI seperti yang disebutkan dalam catatan rilis DLAMI [GPU](#). Untuk membantu Anda memilih DLAMI yang benar untuk kasus penggunaan Anda, kami mengelompokkan gambar berdasarkan jenis perangkat keras atau fungsionalitas yang dikembangkan. Pengelompokan tingkat atas kami adalah:

- Jenis DLAMI: Basis, Kerangka Tunggal, Multi-Kerangka (Conda DLAMI)
- Arsitektur Komputasi: Graviton berbasis [x86](#), [berbasis ARM64 AWS](#)
- Jenis Prosesor: [GPU](#), [CPU](#), [Inferensia](#), [Trainer](#)
- SDK: [CUDA](#), [Neuron AWS](#)
- OS: Amazon Linux, Ubuntu

Topik lainnya dalam panduan ini membantu memberi tahu Anda lebih lanjut dan masuk ke detail lebih lanjut.

Topik

- [Instalasi CUDA dan Binding Kerangka Kerja](#)

- [Dasar Pembelajaran Mendalam AMI](#)
- [Pembelajaran Mendalam AMI dengan Conda](#)
- [Pilihan Arsitektur DLAMI](#)
- [Opsi Sistem Operasi DLAMI](#)

Selanjutnya

[Pembelajaran Mendalam AMI dengan Conda](#)

Instalasi CUDA dan Binding Kerangka Kerja

Sementara pembelajaran mendalam semuanya cukup canggih, setiap kerangka kerja menawarkan versi “stabil”. Versi stabil ini mungkin tidak berfungsi dengan implementasi dan fitur CUDA atau cuDNN terbaru. Kasus penggunaan Anda dan fitur yang Anda butuhkan dapat membantu Anda memilih kerangka kerja. Jika Anda tidak yakin, maka gunakan AMI Pembelajaran Mendalam terbaru dengan Conda. Ini memiliki pip binari resmi untuk semua kerangka kerja dengan CUDA, menggunakan versi terbaru mana pun yang didukung oleh setiap kerangka kerja. Jika Anda menginginkan versi terbaru, dan untuk menyesuaikan lingkungan pembelajaran mendalam Anda, gunakan AMI Dasar Pembelajaran Mendalam.

Lihat panduan kami [Kandidat Stabil Versus Rilis](#) untuk panduan lebih lanjut.

Pilih DLAMI dengan CUDA

[Dasar Pembelajaran Mendalam AMI](#) Memiliki semua seri versi CUDA yang tersedia

[Pembelajaran Mendalam AMI dengan Conda](#) Memiliki semua seri versi CUDA yang tersedia

Note

Kami tidak lagi menyertakan lingkungan MXNet, CNTK, Caffe, Caffe2, Theano, Chainer, atau Keras Conda di. AWS Deep Learning AMIs

Untuk nomor versi kerangka kerja tertentu, lihat [Catatan rilis untuk DLAMIs](#)

Pilih jenis DLAMI ini atau pelajari lebih lanjut tentang DLAMIs perbedaan dengan opsi Berikutnya.

Pilih salah satu versi CUDA dan tinjau daftar lengkap DLAMIs yang memiliki versi itu di Lampiran, atau pelajari lebih lanjut tentang perbedaan DLAMIs dengan opsi Next Up.

Selanjutnya

[Dasar Pembelajaran Mendalam AMI](#)

Topik Terkait

- Untuk petunjuk tentang beralih antara versi CUDA, lihat [Menggunakan Basis Pembelajaran Mendalam AMI](#) tutorial.

Dasar Pembelajaran Mendalam AMI

AMI Basis Pembelajaran Mendalam seperti kanvas kosong untuk pembelajaran mendalam. Muncul dengan semua yang Anda butuhkan sampai titik instalasi kerangka kerja tertentu, dan memiliki pilihan versi CUDA Anda.

Mengapa Memilih DLAMI Dasar

Grup AMI ini berguna bagi kontributor proyek yang ingin melakukan fork project deep learning dan membangun yang terbaru. Ini untuk seseorang yang ingin menggulung lingkungan mereka sendiri dengan keyakinan bahwa perangkat lunak NVIDIA terbaru diinstal dan berfungsi sehingga mereka dapat fokus memilih kerangka kerja dan versi mana yang ingin mereka instal.

Pilih jenis DLAMI ini atau pelajari lebih lanjut tentang DLAMIs perbedaan dengan opsi Next Up.

Selanjutnya

[DLAMI dengan Conda](#)

Topik Terkait

- [Menggunakan Basis Pembelajaran Mendalam AMI](#)

Pembelajaran Mendalam AMI dengan Conda

Conda DLAMI conda menggunakan lingkungan virtual, mereka hadir baik multi-framework atau kerangka kerja tunggal. DLAMIs Lingkungan ini dikonfigurasi untuk menjaga instalasi kerangka kerja yang berbeda terpisah dan merampingkan peralihan antar kerangka kerja. Ini bagus untuk belajar dan bereksperimen dengan semua kerangka kerja yang ditawarkan DLAMI. Sebagian besar pengguna menemukan bahwa AMI Pembelajaran Mendalam baru dengan Conda sangat cocok untuk mereka.

Mereka sering diperbarui dengan versi terbaru dari kerangka kerja, dan memiliki driver dan perangkat lunak GPU terbaru. Mereka umumnya disebut sebagai [AWS Deep Learning AMIs dalam](#) sebagian besar dokumen. Ini DLAMIs mendukung sistem operasi Ubuntu 20.04, Ubuntu 22.04, Amazon Linux 2, Amazon Linux 2023. Dukungan sistem operasi tergantung pada dukungan dari OS upstream.

Kandidat Stabil Versus Rilis

Conda AMIs menggunakan binari yang dioptimalkan dari rilis formal terbaru dari setiap kerangka kerja. Kandidat rilis dan fitur eksperimental tidak diharapkan. Pengoptimalan tergantung pada dukungan kerangka kerja untuk teknologi akselerasi seperti Intel MKL DNN, yang mempercepat pelatihan dan inferensi pada jenis instans CPU C5 dan C4. Binarinya juga dikompilasi untuk mendukung set instruksi Intel tingkat lanjut termasuk namun tidak terbatas pada AVX, AVX-2, .1, dan SSE4 .2. SSE4 ini mempercepat operasi vektor dan floating point pada arsitektur CPU Intel. Selain itu, untuk jenis instans GPU, CUDA dan cuDNN diperbarui dengan versi mana pun yang didukung rilis resmi terbaru.

AMI Pembelajaran Mendalam dengan Conda secara otomatis menginstal versi kerangka kerja yang paling dioptimalkan untuk EC2 instans Amazon Anda pada aktivasi pertama kerangka kerja. Untuk informasi lebih lanjut, lihat [Menggunakan AMI Pembelajaran Mendalam dengan Conda](#).

Jika Anda ingin menginstal dari sumber, menggunakan opsi build khusus atau yang dioptimalkan, [Dasar Pembelajaran Mendalam AMI](#) s mungkin merupakan opsi yang lebih baik untuk Anda.

Pengakhiran Python 2

Komunitas open source Python telah secara resmi mengakhiri dukungan untuk Python 2 pada 1 Januari 2020. PyTorch Komunitas TensorFlow dan telah mengumumkan bahwa rilis TensorFlow 2.1 dan PyTorch 1.4 adalah yang terakhir mendukung Python 2. Rilis DLAMI sebelumnya (v26, v25, dll) yang berisi lingkungan Python 2 Conda terus tersedia. Namun, kami menyediakan pembaruan untuk lingkungan Python 2 Conda pada versi DLAMI yang diterbitkan sebelumnya hanya jika ada perbaikan keamanan yang diterbitkan oleh komunitas sumber terbuka untuk versi tersebut. Rilis DLAMI dengan versi terbaru PyTorch dan kerangka kerja tidak mengandung lingkungan Python 2 Conda. TensorFlow

Dukungan CUDA

Nomor versi CUDA tertentu dapat ditemukan di catatan rilis [DLAMI GPU](#).

Selanjutnya

[Pilihan Arsitektur DLAMI](#)

Topik Terkait

- Untuk tutorial tentang menggunakan AMI Pembelajaran Mendalam dengan Conda, lihat [Menggunakan AMI Pembelajaran Mendalam dengan Conda](#) tutorialnya.

Pilihan Arsitektur DLAMI

AWS Deep Learning AMIs [ditawarkan dengan arsitektur Graviton2 berbasis x86 atau berbasis AWS ARM64](#).

Untuk informasi tentang memulai dengan DLAMI ARM64 GPU, lihat. [ARM64 DLAMI](#) Untuk detail selengkapnya tentang jenis instans yang tersedia, lihat [Memilih tipe instans DLAMI](#).

Selanjutnya

[Opsis Sistem Operasi DLAMI](#)

Opsis Sistem Operasi DLAMI

DLAMIs ditawarkan dalam sistem operasi berikut.

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

Versi lama dari sistem operasi tersedia pada usang DLAMIs. [Untuk informasi selengkapnya tentang penghentian DLAMI, lihat Deprecations for DLAMI](#)

Sebelum memilih DLAMI, nilai jenis instans apa yang Anda butuhkan dan identifikasi Wilayah Anda. AWS

Selanjutnya

[Memilih tipe instans DLAMI](#)

Memilih tipe instans DLAMI

Secara lebih umum, pertimbangkan hal berikut ketika memilih jenis instance untuk DLAMI.

- Jika Anda baru mengenal pembelajaran mendalam, maka instance dengan satu GPU mungkin sesuai dengan kebutuhan Anda.
- Jika Anda sadar anggaran, maka Anda dapat menggunakan instance khusus CPU.
- Jika Anda ingin mengoptimalkan kinerja tinggi dan efisiensi biaya untuk inferensi model pembelajaran mendalam, maka Anda dapat menggunakan instance dengan chip AWS Inferentia.
- Jika Anda mencari instans GPU berkinerja tinggi dengan arsitektur CPU berbasis ARM64, maka Anda dapat menggunakan jenis instans G5G.
- Jika Anda tertarik untuk menjalankan model terlatih untuk inferensi dan prediksi, Anda dapat melampirkan [Amazon Elastic Inference ke instans Amazon](#) Anda. EC2 Amazon Elastic Inference memberi Anda akses ke akselerator dengan sebagian kecil dari GPU.
- Untuk layanan inferensi volume tinggi, satu instance CPU dengan banyak memori, atau sekelompok instance semacam itu, mungkin merupakan solusi yang lebih baik.
- Jika Anda menggunakan model besar dengan banyak data atau ukuran batch tinggi, maka Anda memerlukan instance yang lebih besar dengan lebih banyak memori. Anda juga dapat mendistribusikan model Anda ke sekelompok GPUs. Anda mungkin menemukan bahwa menggunakan instance dengan memori lebih sedikit adalah solusi yang lebih baik untuk Anda jika Anda mengurangi ukuran batch Anda. Ini dapat memengaruhi akurasi dan kecepatan pelatihan Anda.
- Jika Anda tertarik untuk menjalankan aplikasi pembelajaran mesin menggunakan NVIDIA Collective Communications Library (NCCL) yang membutuhkan komunikasi antar-simpul tingkat tinggi dalam skala besar, Anda mungkin ingin menggunakan [Elastic Fabric Adapter \(EFA\)](#).

Untuk detail selengkapnya tentang instance, lihat .

Topik berikut memberikan informasi tentang pertimbangan jenis instance.

Important

Deep Learning AMIs mencakup driver, perangkat lunak, atau toolkit yang dikembangkan, dimiliki, atau disediakan oleh NVIDIA Corporation. Anda setuju untuk menggunakan driver, perangkat lunak, atau toolkit NVIDIA ini hanya pada EC2 instans Amazon yang menyertakan perangkat keras NVIDIA.

Topik

- [Harga untuk DLAMI](#)
- [Ketersediaan Wilayah DLAMI](#)
- [Instans GPU yang Direkomendasikan](#)
- [Instans CPU yang Direkomendasikan](#)
- [Contoh Inferensia yang Direkomendasikan](#)
- [Instans Trainium yang Direkomendasikan](#)

Harga untuk DLAMI

Kerangka kerja pembelajaran mendalam yang termasuk dalam DLAMI gratis, dan masing-masing memiliki lisensi sumber terbuka sendiri. Meskipun perangkat lunak yang disertakan dalam DLAMI gratis, Anda masih harus membayar untuk perangkat keras instans Amazon EC2 yang mendasarinya.

Beberapa jenis EC2 instans Amazon diberi label gratis. Dimungkinkan untuk menjalankan DLAMI pada salah satu contoh gratis ini. Ini berarti bahwa menggunakan DLAMI sepenuhnya gratis ketika Anda hanya menggunakan kapasitas instance itu. Jika Anda membutuhkan instance yang lebih kuat dengan lebih banyak core CPU, lebih banyak ruang disk, lebih banyak RAM, atau satu atau lebih GPUs, maka Anda memerlukan instance yang tidak ada di kelas instance free-tier.

Untuk informasi selengkapnya tentang pilihan dan harga instans, lihat [EC2 harga Amazon](#).

Ketersediaan Wilayah DLAMI

Setiap Wilayah mendukung berbagai jenis instans yang berbeda dan seringkali jenis instans memiliki biaya yang sedikit berbeda di Wilayah yang berbeda. DLAMIs tidak tersedia di setiap Wilayah, tetapi dimungkinkan untuk menyalin DLAMIs ke Wilayah pilihan Anda. Lihat [Menyalin AMI](#) untuk informasi selengkapnya. Perhatikan daftar pilihan Wilayah dan pastikan Anda memilih Wilayah yang dekat dengan Anda atau pelanggan Anda. Jika Anda berencana untuk menggunakan lebih dari satu DLAMI dan berpotensi membuat cluster, pastikan untuk menggunakan Region yang sama untuk semua node di cluster.

Untuk info lebih lanjut tentang Wilayah, kunjungi [titik akhir EC2 layanan Amazon](#).

Selanjutnya

[Instans GPU yang Direkomendasikan](#)

Instans GPU yang Direkomendasikan

Kami merekomendasikan instance GPU untuk sebagian besar tujuan pembelajaran mendalam. Melatih model baru lebih cepat pada instance GPU daripada instance CPU. Anda dapat menskalakan secara sub-linier ketika Anda memiliki instans multi-GPU atau jika Anda menggunakan pelatihan terdistribusi di banyak instance dengan GPUs

Jenis contoh berikut mendukung DLAMI. Untuk informasi tentang opsi tipe instans GPU dan kegunaannya, lihat dan pilih Accelerated Computing.

Note

Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi RAM instans yang tersedia, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- [Instans Amazon EC2 P5e](#) memiliki hingga 8 NVIDIA Tesla H200. GPUs
- [Instans Amazon EC2 P5](#) memiliki hingga 8 NVIDIA Tesla H100. GPUs
- [Instans Amazon EC2 P4](#) memiliki hingga 8 NVIDIA Tesla A100. GPUs
- [Instans Amazon EC2 P3](#) memiliki hingga 8 NVIDIA Tesla V100. GPUs
- [Instans Amazon EC2 G3](#) memiliki hingga 4 NVIDIA Tesla M60. GPUs
- [Instans Amazon EC2 G4](#) memiliki hingga 4 NVIDIA T4. GPUs
- [Instans Amazon EC2 G5](#) memiliki hingga 8 NVIDIA A10G. GPUs
- [Instans Amazon EC2 G6](#) memiliki hingga 8 NVIDIA L4. GPUs
- [Instans Amazon EC2 G6e](#) memiliki hingga 8 NVIDIA L40S Tensor Core. GPUs
- [Instans Amazon EC2 G5G memiliki prosesor Graviton2 berbasis ARM64 AWS](#) .

Instans DLAMI menyediakan perkakas untuk memantau dan mengoptimalkan proses GPU Anda. Untuk informasi selengkapnya tentang memantau proses GPU Anda, lihat [Pemantauan dan Optimasi GPU](#).

Untuk tutorial khusus tentang bekerja dengan instans G5G, lihat. [ARM64 DLAMI](#)

Selanjutnya

[Instans CPU yang Direkomendasikan](#)

Instans CPU yang Direkomendasikan

Baik Anda memiliki anggaran terbatas, belajar tentang pembelajaran mendalam, atau hanya ingin menjalankan layanan prediksi, Anda memiliki banyak opsi terjangkau dalam kategori CPU. Beberapa kerangka kerja memanfaatkan Intel MKL DNN, yang mempercepat pelatihan dan inferensi pada jenis instans CPU C5 (tidak tersedia di semua Wilayah). Untuk informasi tentang tipe instans CPU, lihat [Instance](#) dan pilih Compute Optimized.

Note

Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi RAM instans yang tersedia, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- [Instans Amazon EC2 C5](#) memiliki hingga 72 Intel v. CPUs Instans C5 unggul dalam pemodelan ilmiah, pemrosesan batch, analitik terdistribusi, komputasi kinerja tinggi (HPC), dan inferensi pembelajaran mesin dan mendalam.

Selanjutnya

[Contoh Inferensia yang Direkomendasikan](#)

Contoh Inferensia yang Direkomendasikan

AWS Instance inferensia dirancang untuk memberikan kinerja tinggi dan efisiensi biaya untuk beban kerja inferensi model pembelajaran mendalam. Secara khusus, jenis instans Inf2 menggunakan chip AWS Inferentia dan [AWS Neuron SDK](#), yang terintegrasi dengan kerangka kerja pembelajaran mesin populer seperti dan. TensorFlow PyTorch

Pelanggan dapat menggunakan instans Inf2 untuk menjalankan aplikasi inferensi pembelajaran mesin skala besar seperti pencarian, mesin rekomendasi, visi komputer, pengenalan suara, pemrosesan bahasa alami, personalisasi, dan deteksi penipuan, dengan biaya terendah di cloud.

Note

Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi RAM instans yang tersedia, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- [Instans Amazon EC2 Inf2](#) memiliki hingga 16 chip AWS Inferentia dan throughput jaringan 100 Gbps.

Untuk informasi lebih lanjut tentang memulai dengan AWS Inferentia DLAMIs, lihat [Chip AWS Inferentia Dengan DLAMI](#).

Selanjutnya

[Instans Trainium yang Direkomendasikan](#)

Instans Trainium yang Direkomendasikan

AWS Instans Trainium dirancang untuk memberikan kinerja tinggi dan efisiensi biaya untuk beban kerja inferensi model pembelajaran mendalam. Secara khusus, jenis instans Trn1 menggunakan chip AWS Trainium dan [AWS Neuron SDK](#), yang terintegrasi dengan kerangka kerja pembelajaran mesin populer seperti TensorFlow dan PyTorch.

Pelanggan dapat menggunakan instans Trn1 untuk menjalankan aplikasi inferensi pembelajaran mesin skala besar seperti pencarian, mesin rekomendasi, visi komputer, pengenalan suara, pemrosesan bahasa alami, personalisasi, dan deteksi penipuan, dengan biaya terendah di cloud.

Note

Ukuran model Anda harus menjadi faktor dalam memilih instance. Jika model Anda melebihi RAM instans yang tersedia, pilih jenis instans yang berbeda dengan memori yang cukup untuk aplikasi Anda.

- [Instans Amazon EC2 Trn1](#) memiliki hingga 16 chip AWS Trainium dan throughput jaringan 100 Gbps.

Menyiapkan instance DLAMI

Setelah Anda [memilih DLAMI](#) dan [memilih jenis instans Amazon Elastic Compute Cloud \(EC2 Amazon\)](#) yang ingin Anda gunakan, maka Anda siap untuk menyiapkan instans DLAMI baru Anda.

Jika Anda belum memilih jenis DLAMI EC2 dan instance, lihat. [Memulai dengan DLAMI](#)

Topik

- [Menemukan ID DLAMI](#)
- [Meluncurkan instance DLAMI](#)
- [Menghubungkan ke instans DLAMI](#)
- [Menyiapkan server Jupyter Notebook pada instance DLAMI](#)
- [Membersihkan contoh DLAMI](#)

Menemukan ID DLAMI

Setiap DLAMI memiliki pengenal unik (ID). Saat meluncurkan instans DLAMI menggunakan konsol EC2 Amazon, Anda dapat menggunakan ID DLAMI secara opsional untuk mencari DLAMI yang ingin Anda gunakan. Saat Anda meluncurkan instance DLAMI menggunakan AWS Command Line Interface AWS CLI(), ID ini diperlukan.

Anda dapat menemukan ID untuk DLAMI pilihan Anda dengan menggunakan AWS CLI perintah untuk EC2 Amazon atau Parameter Store, kemampuan dari AWS Systems Manager Untuk petunjuk tentang menginstal dan mengonfigurasi AWS CLI, lihat [Memulai dengan AWS CLI di Panduan AWS Command Line Interface Pengguna](#).

Using Parameter Store

Untuk menemukan ID DLAMI menggunakan ssm get-parameter

Dalam [ssm get-parameter](#) perintah berikut, untuk --name opsi, format nama parameter adalah `aws/service/deeplearning/ami/$architecture/$ami_type/latest/ami-id`. Dalam format nama ini, *architecture* bisa berupa `x86_64` atau `arm64`. Tentukan *ami_type* dengan mengambil nama DLAMI dan menghapus kata kunci “mendalam”, “belajar”, dan “ami”. Nama AMI dapat ditemukan di [Catatan rilis untuk DLAMIs](#).

⚠ Important

Untuk menggunakan perintah ini, prinsipal AWS Identity and Access Management (IAM) yang Anda gunakan harus memiliki `ssm:GetParameter` izin. Untuk informasi selengkapnya tentang prinsip IAM, lihat bagian [Sumber daya tambahan](#) dari peran IAM di Panduan Pengguna IAM.

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

Outputnya akan serupa dengan yang berikut ini:

```
ami-09ee1a996ac214ce7
```

**ℹ Tip**

Untuk beberapa kerangka kerja DLAMI yang saat ini didukung, Anda dapat menemukan perintah contoh yang lebih spesifik di `ssm get-parameter` [Catatan rilis untuk DLAMIs](#) Pilih tautan ke catatan rilis DLAMI pilihan Anda, lalu temukan kueri ID-nya di catatan rilis.

**Using Amazon EC2 CLI**

Untuk menemukan ID DLAMI menggunakan `ec2 describe-images`

Dalam [ec2 describe-images](#) perintah berikut, untuk nilai `filterName=name`, masukkan nama DLAMI. Anda dapat menentukan versi rilis untuk kerangka kerja tertentu, atau Anda bisa mendapatkan rilis terbaru dengan mengganti nomor versi dengan tanda tanya (?).

- ```
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
```

Outputnya akan serupa dengan yang berikut ini:

```
ami-09ee1a996ac214ce7
```

i Tip

Untuk contoh ec2 describe-images perintah yang khusus untuk DLAMI pilihan Anda, lihat. [Catatan rilis untuk DLAMIs](#) Pilih tautan ke catatan rilis DLAMI pilihan Anda, lalu temukan kueri ID-nya di catatan rilis.

Langkah selanjutnya

[Meluncurkan instance DLAMI](#)

Meluncurkan instance DLAMI

Setelah Anda [menemukan ID](#) DLAMI yang ingin Anda gunakan untuk meluncurkan instance DLAMI, Anda siap untuk meluncurkan instance. Untuk meluncurkannya, Anda dapat menggunakan EC2 konsol Amazon atau AWS Command Line Interface (AWS CLI).

i Note

Untuk panduan ini, kami mungkin membuat referensi khusus untuk Deep Learning Base OSS Nvidia Driver GPU AMI (Ubuntu 22.04). Bahkan jika Anda memilih DLAMI yang berbeda, Anda harus dapat mengikuti panduan ini.

EC2 console

i Note

Untuk mempercepat aplikasi komputasi berkinerja tinggi (HPC) dan pembelajaran mesin, Anda dapat meluncurkan instans DLAMI dengan Elastic Fabric Adapter (EFA). Untuk instruksi khusus, lihat [Meluncurkan AWS Deep Learning AMIs Instance Dengan EFA](#).

1. Buka [EC2 konsol](#).

2. Perhatikan saat ini Wilayah AWS di navigasi paling atas. Jika ini bukan Wilayah yang Anda inginkan, maka ubah opsi ini sebelum Anda melanjutkan. Untuk informasi selengkapnya, lihat [Titik akhir EC2 layanan Amazon EC2](#) di Referensi Umum Amazon Web Services
3. Pilih Luncurkan Instans.
4. Masukkan nama untuk instans Anda dan pilih DLAMI yang tepat untuk Anda.
 - a. Temukan DLAMI yang ada di AMIs My atau pilih Mulai Cepat.
 - b. Cari berdasarkan ID DLAMI. Jelajahi opsi lalu pilih pilihan Anda.
5. Pilih jenis instance. Anda dapat menemukan keluarga contoh yang direkomendasikan untuk DLAMI Anda di [Catatan rilis untuk DLAMIs](#) Untuk rekomendasi umum tentang jenis instans DLAMI, lihat [Memilih tipe instans DLAMI](#)
6. Pilih Luncurkan Instans.

AWS CLI

- Untuk menggunakan AWS CLI, Anda harus memiliki ID DLAMI yang ingin Anda gunakan, jenis EC2 dan instance, Wilayah AWS dan informasi token keamanan Anda. Kemudian, Anda dapat meluncurkan instance menggunakan [ec2 run-instances](#) AWS CLI perintah.

Untuk petunjuk tentang menginstal dan mengonfigurasi AWS CLI, lihat [Memulai dengan AWS CLI di](#) Panduan AWS Command Line Interface Pengguna. Untuk informasi selengkapnya, termasuk contoh perintah, lihat [Meluncurkan, mencantumkan, dan menutup EC2 instans Amazon untuk](#). AWS CLI

Setelah meluncurkan instans menggunakan EC2 konsol Amazon atau AWS CLI, tunggu instans siap. Ini biasanya hanya membutuhkan beberapa menit. Anda dapat memverifikasi status instans di [EC2 konsol Amazon](#). Untuk informasi selengkapnya, lihat [Pemeriksaan status untuk EC2 instans](#) Amazon di Panduan EC2 Pengguna Amazon.

Langkah selanjutnya

[Menghubungkan ke instans DLAMI](#)

Menghubungkan ke instans DLAMI

Setelah [meluncurkan instans DLAMI](#) dan instans sedang berjalan, Anda dapat menghubungkannya dari klien (Windows, macOS, atau Linux) menggunakan SSH. Untuk petunjuknya, lihat [Connect ke instans Linux menggunakan SSH](#) di Panduan EC2 Pengguna Amazon.

Simpan salinan perintah login SSH jika Anda ingin mengatur server Notebook Jupyter setelah Anda masuk. Untuk terhubung ke halaman web Jupyter, Anda menggunakan variasi dari perintah itu.

Langkah selanjutnya

[Menyiapkan server Jupyter Notebook pada instance DLAMI](#)

Menyiapkan server Jupyter Notebook pada instance DLAMI

Dengan server Jupyter Notebook, Anda dapat membuat dan menjalankan notebook Jupyter dari instance DLAMI Anda. Dengan notebook Jupyter, Anda dapat melakukan eksperimen pembelajaran mesin (ML) untuk pelatihan dan inferensi saat menggunakan AWS infrastruktur dan mengakses paket yang dibangun ke dalam DLAMI. Untuk informasi selengkapnya tentang notebook Jupyter, lihat Notebook [Jupyter di situs web Dokumentasi Pengguna Jupyter](#).

Untuk menyiapkan server Jupyter Notebook, Anda harus:

- Konfigurasi server Jupyter Notebook pada instance DLAMI Anda.
- Konfigurasi klien Anda untuk terhubung ke server Jupyter Notebook. Kami menyediakan instruksi konfigurasi untuk klien Windows, macOS, dan Linux.
- Uji penyiapan dengan masuk ke server Jupyter Notebook.

Untuk menyelesaikan langkah-langkah ini, ikuti instruksi dalam topik berikut. Setelah menyiapkan server Jupyter Notebook, Anda dapat menjalankan contoh tutorial notebook yang dikirimkan di file. DLAMIs Untuk informasi selengkapnya, lihat [Menjalankan Tutorial Notebook Jupyter](#).

Topik

- [Mengamankan server Jupyter Notebook pada instance DLAMI](#)
- [Memulai server Jupyter Notebook pada instance DLAMI](#)
- [Menghubungkan klien ke server Jupyter Notebook pada instance DLAMI](#)
- [Masuk ke server Jupyter Notebook pada instance DLAMI](#)

Mengamankan server Jupyter Notebook pada instance DLAMI

Agar server Jupyter Notebook Anda tetap aman, kami sarankan untuk menyiapkan kata sandi dan membuat sertifikat SSL untuk server. Untuk mengonfigurasi kata sandi dan SSL, pertama-tama [sambungkan ke instans DLAMI Anda](#), lalu ikuti petunjuk ini.

Untuk mengamankan server Notebook Jupyter

1. Jupyter menyediakan utilitas kata sandi. Jalankan perintah berikut dan masukkan kata sandi pilihan Anda pada prompt.

```
$ jupyter notebook password
```

Outputnya akan terlihat seperti ini:

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

2. Buat sertifikat SSL yang ditandatangani sendiri. Ikuti petunjuk untuk mengisi wilayah Anda sesuai keinginan Anda. Anda harus masuk `.` jika Anda ingin membiarkan prompt kosong. Jawaban Anda tidak akan memengaruhi fungsionalitas sertifikat.

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

Note

Anda mungkin tertarik untuk membuat sertifikat SSL reguler yang ditandatangani oleh pihak ketiga dan tidak menyebabkan browser memberi Anda peringatan keamanan. Proses ini jauh lebih terlibat. Untuk informasi selengkapnya, lihat [Mengamankan server notebook di dokumentasi pengguna](#) Jupyter Notebook.

Langkah selanjutnya

[Memulai server Jupyter Notebook pada instance DLAMI](#)

Memulai server Jupyter Notebook pada instance DLAMI

Setelah [mengamankan server Jupyter Notebook dengan kata sandi dan SSL](#), Anda dapat memulai server. Masuk ke instans DLAMI Anda dan jalankan perintah berikut yang menggunakan sertifikat SSL yang Anda buat sebelumnya.

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

Dengan server dimulai, Anda sekarang dapat terhubung ke sana melalui terowongan SSH dari komputer klien Anda. Ketika server berjalan, Anda akan melihat beberapa output dari Jupyter yang mengonfirmasi bahwa server sedang berjalan. Pada titik ini, abaikan callout bahwa Anda dapat mengakses server melalui URL host lokal, karena itu tidak akan berfungsi sampai Anda membuat terowongan.

Note

Jupyter akan menangani lingkungan switching untuk Anda saat Anda mengganti kerangka kerja menggunakan antarmuka web Jupyter. Untuk informasi selengkapnya, lihat [Beralih Lingkungan dengan Jupyter](#).

Langkah selanjutnya

[Menghubungkan klien ke server Jupyter Notebook pada instance DLAMI](#)

Menghubungkan klien ke server Jupyter Notebook pada instance DLAMI

Setelah Anda [memulai server Jupyter Notebook pada instans DLAMI Anda](#), konfigurasi klien Windows, macOS, atau Linux Anda untuk terhubung ke server. Saat Anda terhubung, Anda dapat membuat dan mengakses notebook Jupyter di server di ruang kerja Anda dan menjalankan kode pembelajaran mendalam Anda di server.

Prasyarat

Pastikan Anda memiliki yang berikut ini, yang Anda perlukan untuk mengatur terowongan SSH:

- Nama DNS publik dari EC2 instans Amazon Anda. Untuk informasi selengkapnya, lihat [jenis nama host EC2 instans Amazon](#) di Panduan EC2 Pengguna Amazon.
- Key pair untuk file kunci pribadi. Untuk informasi selengkapnya tentang mengakses key pair, lihat [pasangan EC2 kunci Amazon dan EC2 instans Amazon](#) di EC2 Panduan Pengguna Amazon.

Connect dari klien Windows, macOS, atau Linux

Untuk terhubung ke instans DLAMI Anda dari klien Windows, macOS, atau Linux, ikuti instruksi untuk sistem operasi klien Anda.

Windows

Untuk terhubung ke instans DLAMI Anda dari klien Windows menggunakan SSH

1. Gunakan klien SSH untuk Windows, seperti PuTTY. Untuk petunjuknya, lihat [Connect ke instance Linux menggunakan PuTTY](#) di EC2 Panduan Pengguna Amazon. Untuk opsi koneksi SSH lainnya, lihat [Connect to Linux Anda menggunakan SSH](#).
2. (Opsional) Buat terowongan SSH ke server Jupyter yang sedang berjalan. Instal Git Bash di klien Windows Anda, lalu ikuti instruksi koneksi untuk klien macOS dan Linux.

macOS or Linux

Untuk terhubung ke instans DLAMI Anda dari klien macOS atau Linux menggunakan SSH

1. Buka terminal.
2. Jalankan perintah berikut untuk meneruskan semua permintaan pada port lokal 8888 ke port 8888 pada instans Amazon EC2 jarak jauh Anda. Perbarui perintah dengan mengganti lokasi kunci Anda untuk mengakses EC2 instans Amazon dan nama DNS publik EC2 instans Amazon Anda. Catatan, untuk Amazon Linux AMI, nama pengguna `ec2-user` bukan `ubuntu`.

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

Perintah ini membuka terowongan antara klien Anda dan EC2 instans Amazon jarak jauh yang menjalankan server Jupyter Notebook.

Langkah selanjutnya

[Masuk ke server Jupyter Notebook pada instance DLAMI](#)

Masuk ke server Jupyter Notebook pada instance DLAMI

Setelah Anda [menghubungkan klien Anda ke server Jupyter Notebook pada instance DLAMI](#) Anda, Anda dapat masuk ke server.

Untuk masuk ke server di browser Anda

1. Di bilah alamat browser Anda, masukkan URL berikut, atau klik tautan ini: <https://localhost:8888>
2. Dengan sertifikat SSL yang ditandatangani sendiri, browser Anda akan memperingatkan Anda dan meminta Anda untuk menghindari terus mengunjungi situs web.



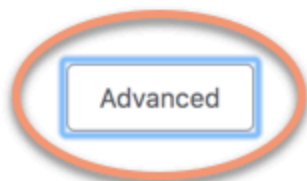
Your connection is not private

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

Karena Anda mengatur ini sendiri, aman untuk melanjutkan. Bergantung pada browser Anda, Anda akan mendapatkan tombol “lanjutan”, “tampilkan detail”, atau serupa.



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 \(unsafe\)](#)

Klik ini, lalu klik tautan “lanjutkan ke localhost”. Jika koneksi berhasil, Anda melihat halaman web server Jupyter Notebook. Pada titik ini, Anda akan diminta kata sandi yang sebelumnya Anda atur.

Sekarang Anda memiliki akses ke server Jupyter Notebook yang berjalan pada instance DLAMI. Anda dapat membuat notebook baru atau menjalankan yang disediakan [Tutorial](#).

Membersihkan contoh DLAMI

Saat Anda tidak lagi membutuhkan instans DLAMI, Anda dapat menghentikannya atau menghentikannya di EC2 Amazon untuk menghindari biaya tak terduga.

Jika Anda menghentikan sebuah instance, Anda dapat menyimpannya dan memulainya nanti ketika Anda ingin menggunakannya lagi. Konfigurasi, file, dan informasi non-volatile lainnya disimpan dalam volume di Amazon Simple Storage Service (Amazon S3). Saat instans dihentikan, Anda dikenakan biaya S3 untuk mempertahankan volume, tetapi Anda tidak dikenakan biaya untuk sumber daya komputasi. Ketika Anda memulai instance lagi, itu akan memasang volume penyimpanan itu dengan data Anda.

Jika Anda menghentikan sebuah instance, itu hilang, dan Anda tidak dapat memulainya lagi. Tentu saja, Anda tidak akan dikenakan biaya lagi untuk sumber daya komputasi dengan instance yang dihentikan. Namun, data Anda masih berada di Amazon S3, dan Anda dapat terus dikenakan biaya S3. Untuk mencegah semua biaya lebih lanjut yang terkait dengan instans yang dihentikan, Anda juga harus menghapus volume penyimpanan di Amazon S3. Untuk petunjuk, lihat [Menghentikan EC2 instans Amazon](#) di EC2 Panduan Pengguna Amazon.

Untuk informasi selengkapnya tentang status EC2 instans Amazon, seperti stopped dan terminated, lihat [perubahan status EC2 instans Amazon](#) di Panduan EC2 Pengguna Amazon.

Menggunakan DLAMI

Topik

- [Menggunakan AMI Pembelajaran Mendalam dengan Conda](#)
- [Menggunakan Basis Pembelajaran Mendalam AMI](#)
- [Menjalankan Tutorial Notebook Jupyter](#)
- [Tutorial](#)

Bagian berikut menjelaskan bagaimana AMI Pembelajaran Mendalam dengan Conda dapat digunakan untuk beralih lingkungan, menjalankan kode sampel dari masing-masing kerangka kerja, dan menjalankan Jupyter sehingga Anda dapat mencoba berbagai tutorial notebook.

Menggunakan AMI Pembelajaran Mendalam dengan Conda

Topik

- [Pengantar AMI Pembelajaran Mendalam dengan Conda](#)
- [Masuk ke DLAMI Anda](#)
- [Mulai TensorFlow Lingkungan](#)
- [Beralih ke Lingkungan PyTorch Python 3](#)
- [Menghapus Lingkungan](#)

Pengantar AMI Pembelajaran Mendalam dengan Conda

Conda adalah sistem manajemen paket open source dan sistem manajemen lingkungan yang berjalan di Windows, macOS, dan Linux. Conda dengan cepat menginstal, menjalankan, dan memperbarui paket dan dependensinya. Conda dengan mudah membuat, menyimpan, memuat, dan beralih antar lingkungan di komputer lokal Anda.

AMI Pembelajaran Mendalam dengan Conda telah dikonfigurasi agar Anda dapat dengan mudah beralih di antara lingkungan pembelajaran yang mendalam. Instruksi berikut memandu Anda pada beberapa perintah dasar denganconda. Mereka juga membantu Anda memverifikasi bahwa impor dasar kerangka kerja berfungsi, dan Anda dapat menjalankan beberapa operasi sederhana dengan

kerangka kerja. Anda kemudian dapat beralih ke tutorial yang lebih menyeluruh yang disediakan dengan DLAMI atau contoh kerangka kerja yang ditemukan di setiap situs proyek kerangka kerja.

Masuk ke DLAMI Anda

Setelah Anda masuk ke server Anda, Anda akan melihat server “message of the day” (MOTD) yang menjelaskan berbagai perintah Conda yang dapat Anda gunakan untuk beralih di antara kerangka kerja pembelajaran mendalam yang berbeda. Di bawah ini adalah contoh MOTD. MOTD spesifik Anda dapat bervariasi saat versi baru DLAMI dirilis.

```
=====
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
=====
```

Mulai TensorFlow Lingkungan

Note

Saat Anda meluncurkan lingkungan Conda pertama Anda, harap bersabar saat memuat. AMI Pembelajaran Mendalam dengan Conda secara otomatis menginstal versi kerangka kerja

yang paling dioptimalkan untuk EC2 instans Anda pada aktivasi pertama kerangka kerja. Anda seharusnya tidak mengharapkan penundaan berikutnya.

1. Aktifkan lingkungan TensorFlow virtual untuk Python 3.

```
$ source activate tensorflow2_p310
```

2. Mulai terminal IPython.

```
(tensorflow2_p310)$ ipython
```

3. Jalankan TensorFlow program cepat.

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Anda akan melihat “Halo, Tensorflow!”

Selanjutnya

[Menjalankan Tutorial Notebook Jupyter](#)

Beralih ke Lingkungan PyTorch Python 3

Jika Anda masih berada di konsol IPython, `quit()` gunakan, lalu bersiaplah untuk beralih lingkungan.

- Aktifkan lingkungan PyTorch virtual untuk Python 3.

```
$ source activate pytorch_p310
```

Uji Beberapa PyTorch Kode

Untuk menguji instalasi Anda, gunakan Python untuk menulis PyTorch kode yang membuat dan mencetak array.

1. Mulai terminal IPython.

```
(pytorch_p310)$ ipython
```

2. Impor PyTorch.

```
import torch
```

Anda mungkin melihat pesan peringatan tentang paket pihak ketiga. Anda dapat mengabaikannya.

3. Buat matriks 5x3 dengan elemen yang diinisialisasi secara acak. Cetak array.

```
x = torch.rand(5, 3)
print(x)
```

Verifikasi hasilnya.

```
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]])
```

Menghapus Lingkungan

Jika Anda kehabisan ruang pada DLAMI, Anda dapat memilih untuk menghapus paket Conda yang tidak Anda gunakan:

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

Menggunakan Basis Pembelajaran Mendalam AMI

Menggunakan Basis Pembelajaran Mendalam AMI

Base AMI hadir dengan platform dasar driver GPU dan pustaka akselerasi untuk menerapkan lingkungan pembelajaran mendalam Anda sendiri yang disesuaikan. Secara default AMI dikonfigurasi

dengan salah satu lingkungan versi NVIDIA CUDA. Anda juga dapat beralih di antara berbagai versi CUDA. Lihat instruksi berikut untuk cara melakukan ini.

Mengkonfigurasi Versi CUDA

Anda dapat memverifikasi versi CUDA dengan menjalankan `nvcc` program NVIDIA.

```
nvcc --version
```

Anda dapat memilih dan memverifikasi versi CUDA tertentu dengan perintah bash berikut:

```
sudo rm /usr/local/cuda
sudo ln -s /usr/local/cuda-12.0 /usr/local/cuda
```

Untuk informasi selengkapnya, lihat catatan [rilis DLAMI Dasar](#).

Menjalankan Tutorial Notebook Jupyter

Tutorial dan contoh dikirimkan dengan masing-masing sumber proyek pembelajaran mendalam dan dalam banyak kasus mereka akan berjalan pada DLAMI apa pun. Jika Anda memilih [Pembelajaran Mendalam AMI dengan Conda](#), Anda mendapatkan manfaat tambahan dari beberapa tutorial pilihan yang sudah disiapkan dan siap untuk dicoba.

Important

Untuk menjalankan tutorial notebook Jupyter yang diinstal pada DLAMI, Anda harus melakukannya. [Menyiapkan server Jupyter Notebook pada instance DLAMI](#)

Setelah server Jupyter berjalan, Anda dapat menjalankan tutorial melalui browser web Anda. Jika Anda menjalankan AMI Pembelajaran Mendalam dengan Conda atau jika Anda telah menyiapkan lingkungan Python, Anda dapat mengganti kernel Python dari antarmuka notebook Jupyter. Pilih kernel yang sesuai sebelum mencoba menjalankan tutorial khusus kerangka kerja. Contoh lebih lanjut dari ini disediakan untuk pengguna AMI Pembelajaran Mendalam dengan Conda.

Note

Banyak tutorial memerlukan modul Python tambahan yang mungkin tidak diatur pada DLAMI Anda. Jika Anda mendapatkan kesalahan seperti "xyz module not found", masuk ke DLAMI, aktifkan lingkungan seperti dijelaskan di atas, lalu instal modul yang diperlukan.

Tip

Tutorial dan contoh pembelajaran mendalam sering bergantung pada satu atau lebih GPUs. Jika tipe instans Anda tidak memiliki GPU, Anda mungkin perlu mengubah beberapa kode contoh untuk menjalankannya.

Menavigasi Tutorial yang Diinstal

Setelah Anda masuk ke server Jupyter dan dapat melihat direktori tutorial (pada Deep Learning AMI dengan Conda saja), Anda akan disajikan dengan folder tutorial dengan masing-masing nama kerangka kerja. Jika Anda tidak melihat kerangka kerja yang terdaftar, maka tutorial tidak tersedia untuk kerangka kerja itu pada DLAMI Anda saat ini. Klik pada nama framework untuk melihat tutorial yang terdaftar, lalu klik tutorial untuk meluncurkannya.

Pertama kali Anda menjalankan notebook pada AMI Pembelajaran Mendalam dengan Conda, ia akan ingin tahu lingkungan mana yang ingin Anda gunakan. Ini akan meminta Anda untuk memilih dari daftar. Setiap lingkungan diberi nama sesuai dengan pola ini:

`Environment (conda_framework_python-version)`

Misalnya, Anda mungkin melihat `Environment (conda_mxnet_p36)`, yang menandakan bahwa lingkungan memiliki MXNet dan Python 3. Variasi lain dari ini adalah `Environment (conda_mxnet_p27)`, yang menandakan bahwa lingkungan memiliki MXNet dan Python 2.

Tip

Jika Anda khawatir tentang versi CUDA mana yang aktif, salah satu cara untuk melihatnya adalah di MOTD saat Anda pertama kali masuk ke DLAMI.

Beralih Lingkungan dengan Jupyter

Jika Anda memutuskan untuk mencoba tutorial untuk kerangka kerja yang berbeda, pastikan untuk memverifikasi kernel yang sedang berjalan. Info ini dapat dilihat di kanan atas antarmuka Jupyter, tepat di bawah tombol logout. Anda dapat mengubah kernel pada notebook yang terbuka dengan mengklik item menu Jupyter Kernel, lalu Change Kernel, dan kemudian mengklik lingkungan yang sesuai dengan notebook yang sedang Anda jalankan.

Pada titik ini Anda harus menjalankan ulang sel apa pun karena perubahan pada kernel akan menghapus status apa pun yang telah Anda jalankan sebelumnya.

Tip

Beralih antar kerangka kerja bisa menyenangkan dan mendidik, namun Anda bisa kehabisan memori. Jika Anda mulai mengalami kesalahan, lihat jendela terminal yang menjalankan server Jupyter. Ada pesan bermanfaat dan pencatatan kesalahan di sini, dan Anda mungkin melihat out-of-memory kesalahan. Untuk memperbaikinya, Anda dapat pergi ke halaman beranda server Jupyter Anda, klik tab Running, lalu klik Shutdown untuk setiap tutorial yang mungkin masih berjalan di latar belakang dan memakan semua memori Anda.

Tutorial

Berikut ini adalah tutorial tentang cara menggunakan AMI Pembelajaran Mendalam dengan perangkat lunak Conda.

Topik

- [Mengaktifkan Kerangka Kerja](#)
- [Pelatihan terdistribusi menggunakan Adaptor Kain Elastis](#)
- [Pemantauan dan Optimasi GPU](#)
- [Chip AWS Inferentia Dengan DLAMI](#)
- [ARM64 DLAMI](#)
- [Inferensi](#)
- [Penyajian Model](#)

Mengaktifkan Kerangka Kerja

Berikut ini adalah kerangka kerja pembelajaran mendalam yang diinstal pada AMI Pembelajaran Mendalam dengan Conda. Klik pada kerangka kerja untuk mempelajari cara mengaktifkannya.

Topik

- [PyTorch](#)
- [TensorFlow 2](#)

PyTorch

Mengaktifkan PyTorch

Ketika paket Conda stabil dari kerangka kerja dirilis, itu diuji dan diinstal sebelumnya pada DLAMI. Jika Anda ingin menjalankan build malam terbaru yang belum teruji, Anda dapat secara manual.

[Instal PyTorch Nightly Build \(eksperimental\)](#)

Untuk mengaktifkan kerangka kerja yang saat ini diinstal, ikuti petunjuk ini pada AMI Pembelajaran Mendalam Anda dengan Conda.

Untuk PyTorch pada Python 3 dengan CUDA dan MKL-DNN, jalankan perintah ini:

```
$ source activate pytorch_p310
```

Mulai terminal IPython.

```
(pytorch_p310)$ ipython
```

Jalankan PyTorch program cepat.

```
import torch
x = torch.rand(5, 3)
print(x)
print(x.size())
y = torch.rand(5, 3)
print(torch.add(x, y))
```

Anda akan melihat array acak awal dicetak, kemudian ukurannya, dan kemudian penambahan array acak lainnya.

Instal PyTorch Nightly Build (eksperimental)

Cara menginstal PyTorch dari build malam

Anda dapat menginstal PyTorch build terbaru ke salah satu atau kedua lingkungan PyTorch Conda di AMI Pembelajaran Mendalam Anda dengan Conda.

- (Opsi untuk Python 3) - Aktifkan lingkungan Python 3: PyTorch

```
$ source activate pytorch_p310
```

- Langkah-langkah yang tersisa mengasumsikan Anda menggunakan `pytorch_p310` lingkungan. Hapus yang saat ini diinstal PyTorch:

```
(pytorch_p310)$ pip uninstall torch
```

- (Opsi untuk instance GPU) - Instal build malam terbaru dengan CUDA.0: PyTorch

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

- (Opsi untuk instance CPU) - Instal build malam terbaru PyTorch untuk instance tanpa: GPUs

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

- Untuk memverifikasi bahwa Anda telah berhasil menginstal build nightly terbaru, mulai IPython terminal dan periksa versi. PyTorch

```
(pytorch_p310)$ ipython
```

```
import torch
print (torch.__version__)
```

Outputnya harus mencetak sesuatu yang mirip dengan `1.0.0.dev20180922`

- Untuk memverifikasi bahwa PyTorch nightly build berfungsi dengan baik dengan contoh MNIST, Anda dapat menjalankan skrip pengujian dari PyTorch repositori contoh:

```
(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
```

Lebih Banyak Tutorial

Untuk tutorial dan contoh lebih lanjut, lihat dokumen resmi kerangka kerja, [PyTorch dokumentasi](#), dan [PyTorch](#) situs web.

TensorFlow 2

Tutorial ini menunjukkan cara mengaktifkan TensorFlow 2 pada instance yang menjalankan AMI Pembelajaran Mendalam dengan Conda (DLAMI di Conda) dan menjalankan program 2. TensorFlow

Ketika paket Conda stabil dari kerangka kerja dirilis, itu diuji dan diinstal sebelumnya pada DLAMI.

Mengaktifkan 2 TensorFlow

Untuk menjalankan TensorFlow DLAMI dengan Conda

1. Untuk mengaktifkan TensorFlow 2, buka instance Amazon Elastic Compute Cloud (Amazon EC2) dari DLAMI dengan Conda.
2. Untuk TensorFlow 2 dan Keras 2 pada Python 3 dengan CUDA 10.1 dan MKL-DNN, jalankan perintah ini:

```
$ source activate tensorflow2_p310
```

3. Mulai terminal IPython:

```
(tensorflow2_p310)$ ipython
```

4. Jalankan program TensorFlow 2 untuk memverifikasi bahwa itu berfungsi dengan baik:

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
tf.print(hello)
```

Hello, TensorFlow! akan muncul di layar Anda.

Lebih Banyak Tutorial

Untuk tutorial dan contoh lainnya, lihat TensorFlow dokumentasi untuk [API TensorFlow Python](#) atau lihat situs webnya. [TensorFlow](#)

Pelatihan terdistribusi menggunakan Adaptor Kain Elastis

[Elastic Fabric Adapter](#) (EFA) adalah perangkat jaringan yang dapat Anda lampirkan ke instans DLAMI Anda untuk mempercepat aplikasi High Performance Computing (HPC). EFA memungkinkan Anda mencapai kinerja aplikasi kluster HPC lokal, dengan skalabilitas, fleksibilitas, dan elastisitas yang disediakan oleh Cloud. AWS

Topik berikut menunjukkan kepada Anda bagaimana memulai menggunakan EFA dengan DLAMI.

Note

Pilih DLAMI Anda dari daftar DLAMI GPU [Dasar](#) ini

Topik

- [Meluncurkan AWS Deep Learning AMIs Instance Dengan EFA](#)
- [Menggunakan EFA pada DLAMI](#)

Meluncurkan AWS Deep Learning AMIs Instance Dengan EFA

[Base DLAMI terbaru siap digunakan dengan EFA dan dilengkapi dengan driver yang diperlukan, modul kernel, libfabric, openmpi dan plugin NCCL OFI untuk instance GPU.](#)

[Anda dapat menemukan versi CUDA yang didukung dari DLAMI Dasar di catatan rilis.](#)

Catatan:

- Saat menjalankan Aplikasi NCCL menggunakan `mpirun` EFA, Anda harus menentukan jalur lengkap ke instalasi yang didukung EFA sebagai:

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

- Untuk mengaktifkan aplikasi Anda menggunakan EFA, tambahkan `FI_PROVIDER="efa"` ke `mpirun` perintah seperti yang ditunjukkan pada [Menggunakan EFA pada DLAMI](#).

Topik

- [Mempersiapkan Grup Keamanan Berkemampuan EFA](#)
- [Luncurkan Instance Anda](#)
- [Verifikasi Lampiran EFA](#)

Mempersiapkan Grup Keamanan Berkemampuan EFA

EFA membutuhkan grup keamanan yang memungkinkan semua lalu lintas masuk dan keluar ke dan dari grup keamanan itu sendiri. Untuk informasi selengkapnya, lihat [Dokumentasi EFA](#).

1. Buka EC2 konsol Amazon di <https://console.aws.amazon.com/ec2/>.
2. Di panel navigasi, pilih Grup Keamanan lalu pilih Buat Grup Keamanan.
3. Di jendela Buat Grup Keamanan, lakukan hal berikut:
 - Untuk Nama grup keamanan, masukkan nama deskriptif untuk grup keamanan, seperti `EFA-enabled security group`.
 - (Opsional) Untuk Deskripsi, masukkan deskripsi singkat grup keamanan.
 - Untuk VPC, pilih VPC untuk tujuan peluncuran instans Anda yang didukung EFA.
 - Pilih Buat.
4. Pilih grup keamanan yang Anda buat, dan pada tab Deskripsi, salin ID Grup.
5. Pada tab Inbound dan Outbound, lakukan hal berikut:
 - Pilih Edit.
 - Untuk Jenis, pilih Semua lalu lintas.
 - Untuk Sumber, pilih Kustom.
 - Rekatkan ID grup keamanan yang Anda salin ke bidang.
 - Pilih Simpan.
6. Aktifkan lalu lintas masuk yang mengacu pada [Otorisasi Lalu Lintas Masuk untuk Instans Linux Anda](#). Jika Anda melewati langkah ini, Anda tidak akan dapat berkomunikasi dengan instans DLAMI Anda.

Luncurkan Instance Anda

EFA pada saat AWS Deep Learning AMIs ini didukung dengan jenis instance dan sistem operasi berikut:

- 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

Bagian berikut menunjukkan cara meluncurkan instance DLAMI yang diaktifkan EFA. Untuk informasi selengkapnya tentang meluncurkan instans berkemampuan EFA, lihat [Meluncurkan Instans Berkemampuan EFA ke dalam](#) Grup Penempatan Cluster.

1. Buka EC2 konsol Amazon di <https://console.aws.amazon.com/ec2/>.
2. Pilih Luncurkan Instans.
3. Pada halaman Pilih AMI, pilih DLAMI yang didukung yang ditemukan di Halaman Catatan Rilis [DLAMI](#)
4. Pada halaman Pilih Jenis Instance, pilih salah satu jenis instans yang didukung berikut, lalu pilih Berikutnya: Konfigurasi Detail Instance. Lihat tautan ini untuk daftar instans yang didukung: [Memulai EFA dan MPI](#)
5. Pada halaman Konfigurasi Detail Instans, lakukan langkah berikut:
 - Untuk Jumlah instans, masukkan jumlah instans yang diaktifkan EFA yang ingin Anda luncurkan.
 - Untuk Jaringan dan Subnet, pilih VPC dan subnet sebagai tujuan peluncuran instans.
 - [Opsional] Untuk grup Penempatan, pilih Tambahkan instance ke grup penempatan. Untuk performa terbaik, luncurkan instance dalam grup penempatan.
 - [Opsional] Untuk nama grup Penempatan, pilih Tambahkan ke grup penempatan baru, masukkan nama deskriptif untuk grup penempatan, lalu untuk strategi grup Penempatan, pilih klaster.
 - Pastikan untuk mengaktifkan “Adaptor Kain Elastis” di halaman ini. Jika opsi ini dinonaktifkan, ubah subnet menjadi subnet yang mendukung jenis instans yang Anda pilih.
 - Di bagian Antarmuka Jaringan, untuk perangkat eth0, pilih Antarmuka jaringan baru. Anda dapat secara opsional menentukan IPv4 alamat utama dan satu atau lebih IPv4 alamat sekunder. Jika Anda meluncurkan instance ke subnet yang memiliki blok IPv6 CIDR terkait, Anda dapat secara opsional menentukan IPv6 alamat utama dan satu atau beberapa alamat sekunder. IPv6
 - Pilih Berikutnya: Tambahkan Penyimpanan.

6. Di halaman Tambahkan Penyimpanan, tentukan volume yang akan dilampirkan ke instans selain volume yang ditentukan oleh AMI (seperti volume perangkat root), lalu pilih Selanjutnya: Tambahkan Tanda.
7. Di halaman Tambahkan Tanda, tentukan tanda untuk instans, seperti nama yang mudah digunakan, lalu pilih Selanjutnya: Konfigurasi Grup Keamanan.
8. Pada halaman Konfigurasi Grup Keamanan, untuk Menetapkan grup keamanan, pilih Pilih grup keamanan yang ada, lalu pilih grup keamanan yang Anda buat sebelumnya.
9. Pilih Tinjau dan Luncurkan.
10. Di halaman Tinjau Peluncuran Instans, tinjau pengaturannya, lalu pilih Luncurkan untuk memilih pasangan kunci dan meluncurkan instans Anda.

Verifikasi Lampiran EFA

Dari Konsol

Setelah meluncurkan instance, periksa detail instance di AWS Console. Untuk melakukan ini, pilih instance di EC2 konsol dan lihat Tab Deskripsi di panel bawah pada halaman. Temukan parameter 'Network Interfaces: eth0' dan klik eth0 yang membuka pop-up. Pastikan 'Adaptor Kain Elastis' diaktifkan.

Jika EFA tidak diaktifkan, Anda dapat memperbaikinya dengan:

- Mengakhiri EC2 instance dan meluncurkan yang baru dengan langkah yang sama. Pastikan EFA terpasang.
- Lampirkan EFA ke instance yang ada.
 1. Di EC2 konsol, buka Network Interfaces.
 2. Klik Buat Antarmuka Jaringan.
 3. Pilih subnet yang sama dengan instans Anda.
 4. Pastikan untuk mengaktifkan 'Adaptor Kain Elastis' dan klik Buat.
 5. Kembali ke Tab EC2 Instances dan pilih instance Anda.
 6. Buka Actions: Instance State dan hentikan instance sebelum Anda melampirkan EFA.
 7. Dari Tindakan, pilih Jaringan: Lampirkan Antarmuka Jaringan.
 8. Pilih antarmuka yang baru saja Anda buat dan klik lampirkan.
 9. Mulai ulang instans Anda.

Dari Instance

Skrip pengujian berikut sudah ada di DLAMI. Jalankan untuk memastikan bahwa modul kernel dimuat dengan benar.

```
$ fi_info -p efa
```

Output-nya semestinya mirip dengan yang berikut.

```
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
```

Verifikasi Konfigurasi Grup Keamanan

Skrip pengujian berikut sudah ada di DLAMI. Jalankan untuk memastikan bahwa grup keamanan yang Anda buat dikonfigurasi dengan benar.

```
$ cd /opt/amazon/efa/test/
$ ./efa_test.sh
```

Output-nya semestinya mirip dengan yang berikut.

```
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

Jika berhenti merespons atau tidak selesai, pastikan bahwa grup keamanan Anda memiliki aturan masuk/keluar yang benar.

Menggunakan EFA pada DLAMI

Bagian berikut menjelaskan cara menggunakan EFA untuk menjalankan aplikasi multi-node pada AWS Deep Learning AMIs

Menjalankan Aplikasi Multi-Node dengan EFA

Untuk menjalankan aplikasi di seluruh cluster node konfigurasi berikut diperlukan

Topik

- [Aktifkan SSH Tanpa Kata Sandi](#)
- [Buat File Host](#)
- [Tes NCCL](#)

Aktifkan SSH Tanpa Kata Sandi

Pilih satu node di cluster Anda sebagai node pemimpin. Node yang tersisa disebut sebagai node anggota.

1. Pada node pemimpin, hasilkan keypair RSA.

```
ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
```

2. Ubah izin kunci privat pada simpul pemimpin.

```
chmod 600 ~/.ssh/id_rsa
```

3. Salin kunci publik `~/.ssh/id_rsa.pub` ke dan tambahkan ke `~/.ssh/authorized_keys` node anggota di cluster.

4. Anda sekarang harus dapat langsung masuk ke node anggota dari node pemimpin menggunakan ip pribadi.

```
ssh <member private ip>
```

5. Nonaktifkan strictHostKey Memeriksa dan mengaktifkan penerusan agen pada node pemimpin dengan menambahkan yang berikut ini ke file ~/.ssh/config pada node pemimpin:

```
Host *  
    ForwardAgent yes  
Host *  
    StrictHostKeyChecking no
```

6. Pada instans Amazon Linux 2, jalankan perintah berikut pada node pemimpin untuk memberikan izin yang benar ke file konfigurasi:

```
chmod 600 ~/.ssh/config
```

Buat File Host

Pada node pemimpin, buat file host untuk mengidentifikasi node di cluster. File host harus memiliki entri untuk setiap node di cluster. Buat file ~/hosts dan tambahkan setiap node menggunakan ip pribadi sebagai berikut:

```
localhost slots=8  
<private ip of node 1> slots=8  
<private ip of node 2> slots=8
```

Tes NCCL

Note

Tes ini telah dijalankan menggunakan EFA versi 1.38.0 dan OFI NCCL Plugin 1.13.2.

Di bawah ini adalah subset dari Tes NCCL yang disediakan oleh Nvidia untuk menguji fungsionalitas dan kinerja melalui beberapa node komputasi

Instans yang Didukung: P3dn, P4, P5, P5e, P5en

Tes Kinerja

Uji Kinerja NCCL Multi-node pada P4D.24xLarge

[Untuk memeriksa Kinerja NCCL dengan EFA, jalankan uji Kinerja NCCL standar yang tersedia di Repo Pengujian NCCL resmi.](#) DLAMI dilengkapi dengan tes ini yang sudah dibangun untuk CUDA XX.X. Anda juga dapat menjalankan skrip Anda sendiri dengan EFA.

Saat membuat skrip Anda sendiri, lihat panduan berikut:

- Gunakan jalur lengkap ke mpirun seperti yang ditunjukkan pada contoh saat menjalankan aplikasi NCCL dengan EFA.
- Ubah params np dan N berdasarkan jumlah instance dan GPUs di cluster Anda.
- Tambahkan flag NCCL_DEBUG=INFO dan pastikan bahwa log menunjukkan penggunaan EFA sebagai “Penyedia Terpilih adalah EFA”.
- Mengatur Lokasi Log Pelatihan untuk mengurai validasi

```
TRAINING_LOG="testEFA_$(date +%N%).log"
```

Gunakan perintah `watch nvidia-smi` pada salah satu node anggota untuk memantau penggunaan GPU. `watch nvidia-smi` Perintah berikut adalah untuk versi CUDA xx.x generik dan bergantung pada Sistem Operasi instance Anda. Anda dapat menjalankan perintah untuk versi CUDA yang tersedia di EC2 instans Amazon Anda dengan mengganti versi CUDA dalam skrip.

- Amazon Linux 2, Amazon Linux 2023:

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

- Ubuntu 20.04, Ubuntu 20.04:

```
$ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \  

```

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

Output Anda akan terlihat seperti berikut:

```
# nThread 1 nGpus 1 minBytes 8 maxBytes 1073741824 step: 2(factor) warmup iters: 5
iters: 100 agg iters: 1 validation: 1 graph: 0
#
# Using devices
# Rank 0 Group 0 Pid 33378 on ip-172-31-42-25 device 0 [0x10] NVIDIA A100-
SXM4-40GB
# Rank 1 Group 0 Pid 33379 on ip-172-31-42-25 device 1 [0x10] NVIDIA A100-
SXM4-40GB
# Rank 2 Group 0 Pid 33380 on ip-172-31-42-25 device 2 [0x20] NVIDIA A100-
SXM4-40GB
# Rank 3 Group 0 Pid 33381 on ip-172-31-42-25 device 3 [0x20] NVIDIA A100-
SXM4-40GB
# Rank 4 Group 0 Pid 33382 on ip-172-31-42-25 device 4 [0x90] NVIDIA A100-
SXM4-40GB
# Rank 5 Group 0 Pid 33383 on ip-172-31-42-25 device 5 [0x90] NVIDIA A100-
SXM4-40GB
# Rank 6 Group 0 Pid 33384 on ip-172-31-42-25 device 6 [0xa0] NVIDIA A100-
SXM4-40GB
# Rank 7 Group 0 Pid 33385 on ip-172-31-42-25 device 7 [0xa0] NVIDIA A100-
SXM4-40GB
# Rank 8 Group 0 Pid 30378 on ip-172-31-43-8 device 0 [0x10] NVIDIA A100-SXM4-40GB
# Rank 9 Group 0 Pid 30379 on ip-172-31-43-8 device 1 [0x10] NVIDIA A100-SXM4-40GB
# Rank 10 Group 0 Pid 30380 on ip-172-31-43-8 device 2 [0x20] NVIDIA A100-SXM4-40GB
# Rank 11 Group 0 Pid 30381 on ip-172-31-43-8 device 3 [0x20] NVIDIA A100-SXM4-40GB
# Rank 12 Group 0 Pid 30382 on ip-172-31-43-8 device 4 [0x90] NVIDIA A100-SXM4-40GB
# Rank 13 Group 0 Pid 30383 on ip-172-31-43-8 device 5 [0x90] NVIDIA A100-SXM4-40GB
# Rank 14 Group 0 Pid 30384 on ip-172-31-43-8 device 6 [0xa0] NVIDIA A100-SXM4-40GB
# Rank 15 Group 0 Pid 30385 on ip-172-31-43-8 device 7 [0xa0] NVIDIA A100-SXM4-40GB
ip-172-31-42-25:33385:33385 [7] NCCL INFO cudaDriverVersion 12060
ip-172-31-43-8:30383:30383 [5] NCCL INFO Bootstrap : Using ens32:172.31.43.8
ip-172-31-43-8:30383:30383 [5] NCCL INFO NCCL version 2.23.4+cuda12.5
```

```

...
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.13.2-aws
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Using Libfabric version 1.22
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Using CUDA driver version 12060 with
runtime 12050
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Configuring AWS-specific options
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting provider_filter to efa
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting FI_EFA_FORK_SAFE environment
variable to 1
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting NCCL_NVLSTREE_MAX_CHUNKSIZE
to 512KiB
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Setting NCCL_NVLS_CHUNKSIZE to 512KiB
ip-172-31-42-25:33384:33451 [6] NCCL INFO NET/OFI Running on p4d.24xlarge platform,
Setting NCCL_TOPO_FILE environment variable to /opt/amazon/of-nccl/share/aws-ofi-
nccl/xml/p4d-24x1-topo.xml
...

```

-----some output truncated-----

#	in-place				out-of-place						
#	size	count	type	redop	root	time	algbw	busbw	#wrong		
#	(B)	(elements)				(us)	(GB/s)	(GB/s)			
(us)	(GB/s)	(GB/s)									
179.3	8	2	float	sum	-1	180.3	0.00	0.00	0		
177.6	16	4	float	sum	-1	178.1	0.00	0.00	0		
177.9	32	8	float	sum	-1	178.5	0.00	0.00	0		
178.7	64	16	float	sum	-1	178.8	0.00	0.00	0		
177.8	128	32	float	sum	-1	178.2	0.00	0.00	0		
178.8	256	64	float	sum	-1	178.6	0.00	0.00	0		
177.1	512	128	float	sum	-1	177.2	0.00	0.01	0		
179.3	1024	256	float	sum	-1	179.2	0.01	0.01	0		
181.2	2048	512	float	sum	-1	181.3	0.01	0.02	0		
183.9	4096	1024	float	sum	-1	184.2	0.02	0.04	0		

```

      8192      2048      float      sum      -1      191.2      0.04      0.08      0
190.6  0.04    0.08      0
      16384     4096      float      sum      -1      202.5      0.08      0.15      0
202.3  0.08    0.15      0
      32768     8192      float      sum      -1      233.0      0.14      0.26      0
232.1  0.14    0.26      0
      65536    16384     float      sum      -1      238.6      0.27      0.51      0
235.1  0.28    0.52      0
      131072    32768     float      sum      -1      237.2      0.55      1.04      0
236.8  0.55    1.04      0
      262144    65536     float      sum      -1      248.3      1.06      1.98      0
247.0  1.06    1.99      0
      524288    131072    float      sum      -1      309.2      1.70      3.18      0
307.7  1.70    3.20      0
      1048576    262144    float      sum      -1      408.7      2.57      4.81      0
404.3  2.59    4.86      0
      2097152    524288    float      sum      -1      613.5      3.42      6.41      0
607.9  3.45    6.47      0
      4194304    1048576    float      sum      -1      924.5      4.54      8.51      0
914.8  4.58    8.60      0
      8388608    2097152    float      sum      -1      1059.5     7.92     14.85     0
1054.3  7.96    14.92     0
      16777216    4194304    float      sum      -1      1269.9     13.21     24.77     0
1272.0  13.19   24.73     0
      33554432    8388608    float      sum      -1      1642.7     20.43     38.30     0
1636.7  20.50   38.44     0
      67108864    16777216    float      sum      -1      2446.7     27.43     51.43     0
2445.8  27.44   51.45     0
      134217728    33554432    float      sum      -1      4143.6     32.39     60.73     0
4142.4  32.40   60.75     0
      268435456    67108864    float      sum      -1      7351.9     36.51     68.46     0
7346.7  36.54   68.51     0
      536870912    134217728    float      sum      -1      13717     39.14     73.39     0
13703  39.18   73.46     0
      1073741824    268435456    float      sum      -1      26416     40.65     76.21     0
26420  40.64   76.20     0
...
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 15.5514

```

Tes Validasi

Untuk memvalidasi bahwa tes EFA mengembalikan hasil yang valid, silakan gunakan tes berikut untuk mengonfirmasi:

- Dapatkan jenis instance menggunakan EC2 Metadata Instance:

```
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCE_TYPE=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/instance-type)
```

- Jalankan [Tes Kinerja](#)
- Mengatur Parameter Berikut

```
CUDA_VERSION
CUDA_RUNTIME_VERSION
NCCL_VERSION
```

- Validasi Hasil seperti yang ditunjukkan:

```
RETURN_VAL=`echo $?`
if [ ${RETURN_VAL} -eq 0 ]; then

    # [0] NCCL INFO NET/OFI Initializing aws-ofi-nccl 1.13.2-aws
    # [0] NCCL INFO NET/OFI Using CUDA driver version 12060 with runtime 12010

    # cudaDriverVersion 12060 --> This is max supported cuda version by nvidia
    driver
    # NCCL version 2.23.4+cuda12.5 --> This is NCCL version compiled with cuda
    version

    # Validation of logs
    grep "NET/OFI Configuring AWS-specific options" ${TRAINING_LOG} || { echo "AWS-
specific options text not found"; exit 1; }
    grep "busbw" ${TRAINING_LOG} || { echo "busbw text not found"; exit 1; }
    grep "Avg bus bandwidth " ${TRAINING_LOG} || { echo "Avg bus bandwidth text not
found"; exit 1; }
    grep "NCCL version $NCCL_VERSION" ${TRAINING_LOG} || { echo "Text not found: NCCL
version $NCCL_VERSION"; exit 1; }
    if [[ ${INSTANCE_TYPE} == "p4d.24xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Text not found: NET/
Libfabric/0/GDRDMA"; exit 1; }
```

```

    grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Selected Provider is efa text not found"; exit 1; }
    elif [[ ${INSTANCE_TYPE} == "p4de.24xlarge" ]]; then
        grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
        elif [[ ${INSTANCE_TYPE} == "p5.48xlarge" ]]; then
            grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
            grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
            elif [[ ${INSTANCE_TYPE} == "p5e.48xlarge" ]]; then
                grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
                grep "NET/OFI Selected Provider is efa (found 32 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
                elif [[ ${INSTANCE_TYPE} == "p5en.48xlarge" ]]; then
                    grep "NET/Libfabric/0/GDRDMA" ${TRAINING_LOG} || { echo "Avg bus bandwidth
text not found"; exit 1; }
                    grep "NET/OFI Selected Provider is efa (found 16 nics)" ${TRAINING_LOG} ||
{ echo "Avg bus bandwidth text not found"; exit 1; }
                    elif [[ ${INSTANCE_TYPE} == "p3dn.24xlarge" ]]; then
                        grep "NET/OFI Selected Provider is efa (found 4 nics)" ${TRAINING_LOG} ||
{ echo "Selected Provider is efa text not found"; exit 1; }
                    fi
                    echo "***** check_efa_nccl_all_reduce passed for cuda
version ${CUDA_VERSION} *****"
                else
                    echo "***** check_efa_nccl_all_reduce failed for cuda
version ${CUDA_VERSION} *****"
                fi
            fi

```

- Untuk mengakses data benchmark, kita dapat mengurai baris terakhir dari output tabel dari tes Multi Node all_reduce:

```

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}"
```

Pemantauan dan Optimasi GPU

Bagian berikut akan memandu Anda melalui opsi pengoptimalan dan pemantauan GPU. Bagian ini diatur seperti alur kerja biasa dengan pemantauan mengawasi prapemrosesan dan pelatihan.

- [Pemantauan](#)
 - [Monitor GPUs dengan CloudWatch](#)
- [Pengoptimalan](#)
 - [Pemrosesan awal](#)
 - [Pelatihan](#)

Pemantauan

DLAMI Anda sudah diinstal sebelumnya dengan beberapa alat pemantauan GPU. Panduan ini juga menyebutkan alat yang tersedia untuk diunduh dan dipasang.

- [Monitor GPUs dengan CloudWatch](#) - utilitas prainstal yang melaporkan statistik penggunaan GPU ke Amazon. CloudWatch
- [nvidia-smi CLI](#) - utilitas untuk memantau komputasi GPU secara keseluruhan dan pemanfaatan memori. Ini sudah diinstal sebelumnya pada AWS Deep Learning AMIs (DLAMI) Anda.
- [Pustaka NVMLC](#) - API berbasis C untuk mengakses fungsi pemantauan dan manajemen GPU secara langsung. Ini digunakan oleh CLI `nvidia-smi` di bawah kap dan sudah diinstal sebelumnya pada DLAMI Anda. Ini juga memiliki ikatan Python dan Perl untuk memfasilitasi pengembangan dalam bahasa-bahasa tersebut. Utilitas `gpumon.py` yang sudah diinstal sebelumnya pada DLAMI Anda menggunakan paket `pynvml`. dari [nvidia-ml-py](#)
- [NVIDIA DCGM](#) - Alat manajemen cluster. Kunjungi halaman pengembang untuk mempelajari cara menginstal dan mengkonfigurasi alat ini.

Tip

Lihat blog pengembang NVIDIA untuk info terbaru tentang penggunaan alat CUDA yang diinstal DLAMI Anda:

- [Pemantauan TensorCore pemanfaatan menggunakan Nsight IDE dan nvprof.](#)

Monitor GPUs dengan CloudWatch

Saat Anda menggunakan DLAMI dengan GPU, Anda mungkin menemukan bahwa Anda mencari cara untuk melacak penggunaannya selama pelatihan atau inferensi. Ini dapat berguna untuk mengoptimalkan pipeline data Anda, dan menyetel jaringan pembelajaran mendalam Anda.

Ada dua cara untuk mengonfigurasi metrik GPU dengan: CloudWatch

- [Konfigurasi metrik dengan AWS CloudWatch agen \(Disarankan\)](#)
- [Konfigurasi metrik dengan skrip yang sudah diinstal sebelumnya gpumon.py](#)

Konfigurasi metrik dengan AWS CloudWatch agen (Disarankan)

Integrasikan DLAMI Anda dengan agen [CloudWatch terpadu](#) untuk mengonfigurasi metrik GPU dan memantau pemanfaatan proses bersama GPU di instans akselerasi Amazon. EC2

Ada empat cara untuk mengonfigurasi [metrik GPU](#) dengan DLAMI Anda:

- [Konfigurasi metrik GPU minimal](#)
- [Konfigurasi metrik GPU sebagian](#)
- [Konfigurasi semua metrik GPU yang tersedia](#)
- [Konfigurasi metrik GPU khusus](#)

Untuk informasi tentang pembaruan dan patch keamanan, lihat [Penambalan keamanan untuk agen AWS CloudWatch](#)

Prasyarat

Untuk memulai, Anda harus mengonfigurasi izin IAM EC2 instans Amazon yang memungkinkan instans Anda mendorong metrik. CloudWatch Untuk langkah-langkah mendetail, lihat [Membuat peran IAM dan pengguna untuk digunakan dengan CloudWatch agen.](#)

Konfigurasi metrik GPU minimal

Konfigurasi metrik GPU minimal menggunakan layanan. `dlami-cloudwatch-agent@minimal systemd` Layanan ini mengonfigurasi metrik berikut:

- `utilization_gpu`
- `utilization_memory`

Anda dapat menemukan `systemd` layanan untuk metrik GPU minimal yang telah dikonfigurasi sebelumnya di lokasi berikut:

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

Aktifkan dan mulai `systemd` layanan dengan perintah berikut:

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

Konfigurasi metrik GPU sebagian

Konfigurasi metrik GPU sebagian menggunakan layanan. `dlami-cloudwatch-agent@partial` `systemd` Layanan ini mengonfigurasi metrik berikut:

- `utilization_gpu`
- `utilization_memory`
- `memory_total`
- `memory_used`
- `memory_free`

Anda dapat menemukan `systemd` layanan untuk metrik GPU sebagian yang telah dikonfigurasi sebelumnya di lokasi berikut:

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

Aktifkan dan mulai `systemd` layanan dengan perintah berikut:

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

Konfigurasi semua metrik GPU yang tersedia

Konfigurasi semua metrik GPU yang tersedia menggunakan layanan `dlami-cloudwatch-agent@all systemd`. Layanan ini mengonfigurasi metrik berikut:

- `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`

Anda dapat menemukan `systemd` layanan untuk semua metrik GPU yang telah dikonfigurasi sebelumnya di lokasi berikut:

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

Aktifkan dan mulai `systemd` layanan dengan perintah berikut:

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

Konfigurasi metrik GPU khusus

Jika metrik yang telah dikonfigurasi sebelumnya tidak memenuhi persyaratan Anda, Anda dapat membuat file konfigurasi CloudWatch agen kustom.

Buat file konfigurasi khusus

Untuk membuat file konfigurasi khusus, lihat langkah-langkah terperinci di [Buat atau edit file konfigurasi CloudWatch agen secara manual](#).

Untuk contoh ini, asumsikan bahwa definisi skema terletak di `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`.

Konfigurasi metrik dengan file kustom Anda

Jalankan perintah berikut untuk mengkonfigurasi CloudWatch agen sesuai dengan file kustom Anda:

```
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
```

Penambalan keamanan untuk agen AWS CloudWatch

Yang baru dirilis DLAMIs dikonfigurasi dengan patch keamanan AWS CloudWatch agen terbaru yang tersedia. Lihat bagian berikut untuk memperbarui DLAMI Anda saat ini dengan patch keamanan terbaru tergantung pada sistem operasi pilihan Anda.

Amazon Linux 2

Gunakan yum untuk mendapatkan patch keamanan AWS CloudWatch agen terbaru untuk Amazon Linux 2 DLAMI.

```
sudo yum update
```

Ubuntu

Untuk mendapatkan patch AWS CloudWatch keamanan terbaru untuk DLAMI dengan Ubuntu, Anda perlu menginstal ulang agen AWS CloudWatch menggunakan tautan unduhan Amazon S3.

```
wget https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/
amazon-cloudwatch-agent.deb
```

Untuk informasi selengkapnya tentang menginstal AWS CloudWatch agen menggunakan tautan unduhan Amazon S3, lihat [Menginstal dan menjalankan CloudWatch agen di server Anda](#).

Konfigurasi metrik dengan skrip yang sudah diinstal sebelumnya **gpumon.py**

Sebuah utilitas yang disebut gpumon.py sudah diinstal pada DLAMI Anda. Ini terintegrasi dengan CloudWatch dan mendukung pemantauan penggunaan per-GPU: memori GPU, suhu GPU, dan Daya GPU. Script secara berkala mengirimkan data yang dipantau ke CloudWatch. Anda dapat mengonfigurasi tingkat granularitas untuk data yang dikirim CloudWatch dengan mengubah beberapa pengaturan dalam skrip. Namun, sebelum memulai skrip, Anda harus mengatur CloudWatch untuk menerima metrik.

Cara mengatur dan menjalankan pemantauan GPU dengan CloudWatch

1. Buat pengguna IAM, atau ubah pengguna yang sudah ada agar memiliki kebijakan untuk memublikasikan metrik ke CloudWatch. Jika Anda membuat pengguna baru, harap perhatikan kredensialnya karena Anda akan membutuhkannya di langkah berikutnya.

Kebijakan IAM untuk mencari adalah "cloudwatch:". PutMetricData Kebijakan yang ditambahkan adalah sebagai berikut:

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

Tip

Untuk informasi selengkapnya tentang membuat pengguna IAM dan menambahkan kebijakan untuk CloudWatch, lihat [CloudWatch dokumentasi](#).

2. Pada DLAMI Anda, [AWS jalankan](#) configure dan tentukan kredensial pengguna IAM.

```
$ aws configure
```

3. Anda mungkin perlu membuat beberapa modifikasi pada utilitas gpumon sebelum menjalankannya. Anda dapat menemukan utilitas gpumon dan README di lokasi yang ditentukan dalam blok kode berikut. Untuk informasi selengkapnya tentang gpumon.py skrip, lihat [lokasi skrip Amazon S3](#).

```
Folder: ~/tools/GPUCloudWatchMonitor  
Files:  ~/tools/GPUCloudWatchMonitor/gpumon.py  
        ~/tools/GPUCloudWatchMonitor/README
```

Opsi:

- Ubah wilayah di gpumon.py jika instance Anda TIDAK ada di us-east-1.
 - Ubah parameter lain seperti CloudWatch namespace atau periode pelaporan dengan `store_reso`.
4. Saat ini skrip hanya mendukung Python 3. Aktifkan lingkungan Python 3 kerangka kerja pilihan Anda atau aktifkan lingkungan umum Python 3 DLAMI.

```
$ source activate python3
```

5. Jalankan utilitas gpumon di latar belakang.


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

6. Buka browser Anda ke metrik <https://console.aws.amazon.com/cloudwatch/> lalu pilih. Ini akan memiliki namespace ". DeepLearningTrain

Tip

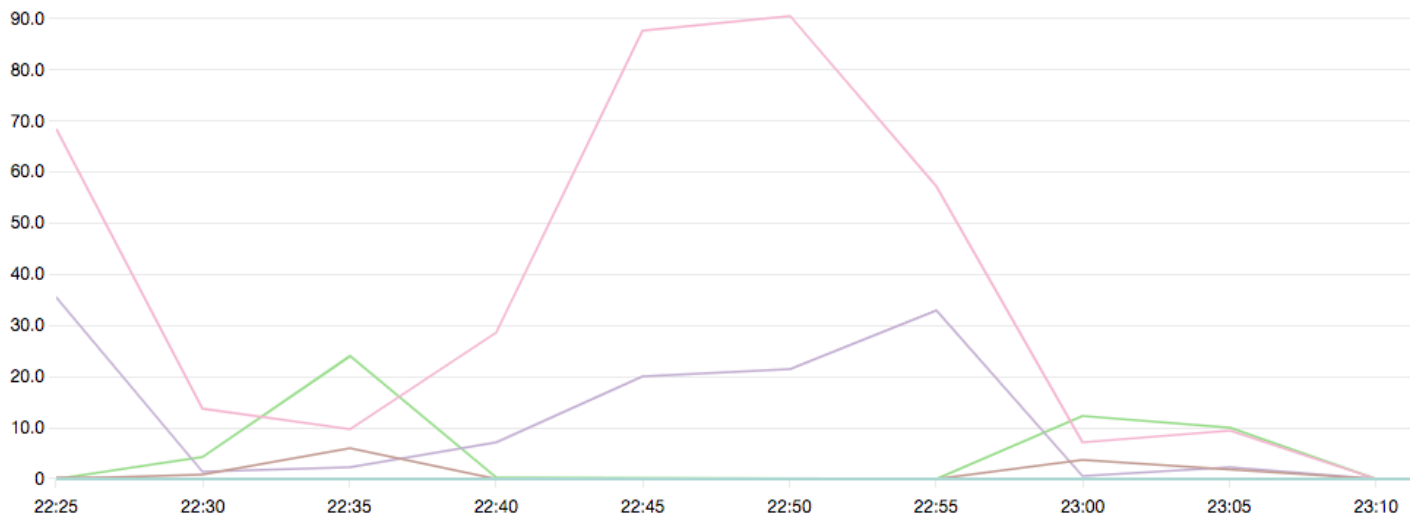
Anda dapat mengubah namespace dengan memodifikasi gpumon.py. Anda juga dapat mengubah interval pelaporan dengan menyesuaikan `store_reso`.

Berikut ini adalah contoh CloudWatch bagan pelaporan pada menjalankan gpumon.py memantau pekerjaan pelatihan pada instance p2.8xlarge.

GPU usage, Memory usage 

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

Various units



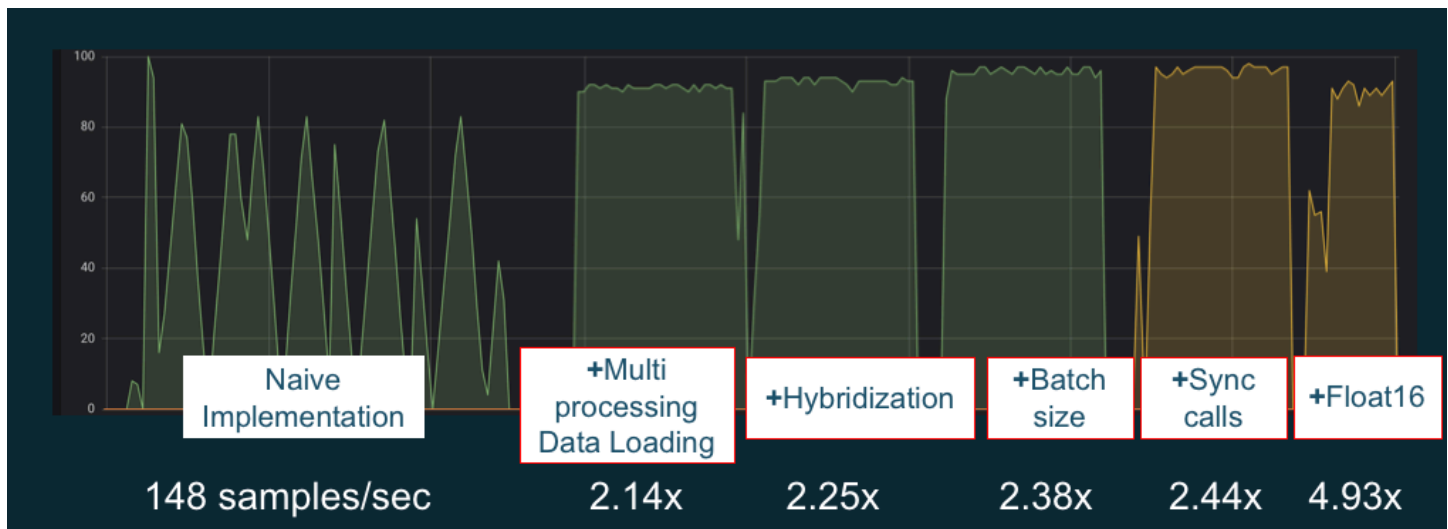
Anda mungkin tertarik dengan topik lain tentang pemantauan dan pengoptimalan GPU ini:

- [Pemantauan](#)
 - [Monitor GPUs dengan CloudWatch](#)
- [Pengoptimalan](#)
 - [Pemrosesan awal](#)
 - [Pelatihan](#)

Pengoptimalan

Untuk memaksimalkan GPUs, Anda dapat mengoptimalkan pipeline data dan menyetel jaringan pembelajaran mendalam Anda. Seperti yang dijelaskan bagan berikut, implementasi naif atau dasar dari jaringan saraf mungkin menggunakan GPU secara tidak konsisten dan tidak secara maksimal. Saat Anda mengoptimalkan preprocessing dan pemuatan data, Anda dapat mengurangi hambatan dari CPU ke GPU Anda. Anda dapat menyesuaikan jaringan saraf itu sendiri, dengan menggunakan hibridisasi (bila didukung oleh kerangka kerja), menyesuaikan ukuran batch, dan menyinkronkan panggilan. Anda juga dapat menggunakan pelatihan presisi ganda (float16 atau int8) di sebagian besar kerangka kerja, yang dapat memiliki efek dramatis pada peningkatan throughput.

Bagan berikut menunjukkan peningkatan kinerja kumulatif saat menerapkan pengoptimalan yang berbeda. Hasil Anda akan tergantung pada data yang Anda proses dan jaringan yang Anda optimalkan.



Contoh optimasi kinerja GPU. Sumber bagan: [Trik Kinerja dengan MXNet Gluon](#)

Panduan berikut memperkenalkan opsi yang akan bekerja dengan DLAMI Anda dan membantu Anda meningkatkan kinerja GPU.

Topik

- [Pemrosesan awal](#)
- [Pelatihan](#)

Pemrosesan awal

Preprocessing data melalui transformasi atau augmentasi seringkali dapat menjadi proses yang terikat CPU, dan ini bisa menjadi hambatan dalam keseluruhan pipeline Anda. Kerangka kerja memiliki operator bawaan untuk pemrosesan gambar, tetapi DALI (Data Augmentation Library) menunjukkan peningkatan kinerja dibandingkan opsi bawaan kerangka kerja.

- Perpustakaan Augmentasi Data NVIDIA (DALI): DALI membongkar augmentasi data ke GPU. Ini tidak diinstal sebelumnya pada DLAMI, tetapi Anda dapat mengaksesnya dengan menginstalnya atau memuat wadah kerangka kerja yang didukung pada DLAMI Anda atau instans Amazon Elastic Compute Cloud lainnya. Lihat [halaman proyek DALI](#) di situs web NVIDIA untuk detailnya. Untuk contoh kasus penggunaan dan untuk mengunduh contoh kode, lihat contoh Kinerja [Pelatihan SageMaker Pra-Pemrosesan](#).
- NVJPEG: perpustakaan dekoder JPEG yang dipercepat GPU untuk pemrogram C. [Ini mendukung decoding gambar tunggal atau batch serta operasi transformasi berikutnya yang umum dalam](#)

[pembelajaran mendalam. nvJpeg dilengkapi dengan DALI, atau Anda dapat mengunduh dari halaman nvjpeg situs web NVIDIA dan menggunakannya secara terpisah.](#)

Anda mungkin tertarik dengan topik lain tentang pemantauan dan pengoptimalan GPU ini:

- [Pemantauan](#)
 - [Monitor GPUs dengan CloudWatch](#)
- [Pengoptimalan](#)
 - [Pemrosesan awal](#)
 - [Pelatihan](#)

Pelatihan

Dengan pelatihan presisi campuran, Anda dapat menggunakan jaringan yang lebih besar dengan jumlah memori yang sama, atau mengurangi penggunaan memori dibandingkan dengan jaringan presisi tunggal atau ganda Anda, dan Anda akan melihat peningkatan kinerja komputasi. Anda juga mendapatkan manfaat dari transfer data yang lebih kecil dan lebih cepat, faktor penting dalam pelatihan terdistribusi beberapa node. Untuk memanfaatkan pelatihan presisi campuran, Anda perlu menyesuaikan pengecoran data dan penskalaan kerugian. Berikut ini adalah panduan yang menjelaskan cara melakukan ini untuk kerangka kerja yang mendukung presisi campuran.

- [NVIDIA Deep Learning SDK](#) - dokumen di situs web NVIDIA yang menjelaskan implementasi presisi campuran untuk, dan. MXNet PyTorch TensorFlow

Tip

Pastikan untuk memeriksa situs web untuk kerangka pilihan Anda, dan cari “presisi campuran” atau “fp16” untuk teknik pengoptimalan terbaru. Berikut adalah beberapa panduan presisi campuran yang mungkin berguna bagi Anda:

- [Pelatihan presisi campuran dengan TensorFlow \(video\)](#) - di situs blog NVIDIA.
- [Pelatihan presisi campuran menggunakan float16 dengan MXNet - artikel](#) FAQ di situs web. MXNet
- [NVIDIA Apex: alat untuk pelatihan presisi campuran yang mudah dengan PyTorch](#) - artikel blog di situs web NVIDIA.

Anda mungkin tertarik dengan topik lain tentang pemantauan dan pengoptimalan GPU ini:

- [Pemantauan](#)
 - [Monitor GPUs dengan CloudWatch](#)
- [Pengoptimalan](#)
 - [Pemrosesan awal](#)
 - [Pelatihan](#)

Chip AWS Inferentia Dengan DLAMI

AWS Inferentia adalah chip pembelajaran mesin khusus yang dirancang oleh AWS yang dapat Anda gunakan untuk prediksi inferensi kinerja tinggi. Untuk menggunakan chip, siapkan instans Amazon Elastic Compute Cloud dan gunakan kit pengembangan perangkat lunak AWS Neuron (SDK) untuk memanggil chip Inferentia. Untuk memberikan pengalaman Inferentia terbaik kepada pelanggan, Neuron telah dibangun ke dalam AWS Deep Learning AMIs (DLAMI).

Topik berikut menunjukkan kepada Anda bagaimana memulai menggunakan Inferentia dengan DLAMI.

Daftar Isi

- [Meluncurkan Instance DLAMI dengan Neuron AWS](#)
- [Menggunakan DLAMI dengan Neuron AWS](#)

Meluncurkan Instance DLAMI dengan Neuron AWS

DLAMI terbaru siap digunakan AWS dengan Inferentia dan dilengkapi dengan AWS paket Neuron API. Untuk meluncurkan instans DLAMI, [lihat Meluncurkan dan Mengonfigurasi DLAMI](#). Setelah Anda memiliki DLAMI, gunakan langkah-langkah di sini untuk memastikan bahwa chip Inferentia AWS dan sumber daya Neuron AWS Anda aktif.

Daftar Isi

- [Verifikasi Instance Anda](#)
- [Mengidentifikasi Perangkat AWS Inferentia](#)
- [Lihat Penggunaan Sumber Daya](#)
- [Menggunakan Neuron Monitor \(neuron-monitor\)](#)
- [Meningkatkan Perangkat Lunak Neuron](#)

Verifikasi Instance Anda

Sebelum menggunakan instance Anda, verifikasi bahwa itu diatur dan dikonfigurasi dengan benar dengan Neuron.

Mengidentifikasi Perangkat AWS Inferensia

Untuk mengidentifikasi jumlah perangkat Inferentia pada instans Anda, gunakan perintah berikut:

```
neuron-ls
```

Jika instans Anda memiliki perangkat Inferentia yang terpasang padanya, output Anda akan terlihat mirip dengan yang berikut ini:

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

Output yang disediakan diambil dari instance INF1.6xLarge dan menyertakan kolom berikut:

- **PERANGKAT NEURON:** ID logis yang ditetapkan ke NeuronDevice. ID ini digunakan saat mengonfigurasi beberapa runtime untuk menggunakan yang berbeda. NeuronDevices
- **NEURON CORES:** Jumlah NeuronCores hadir di NeuronDevice.
- **MEMORI NEURON:** Jumlah memori DRAM di. NeuronDevice
- **PERANGKAT YANG TERHUBUNG:** Lainnya NeuronDevices terhubung ke NeuronDevice.
- **PCI BDF:** ID Fungsi Perangkat Bus PCI (BDF) dari file. NeuronDevice

Lihat Penggunaan Sumber Daya

Lihat informasi yang berguna tentang NeuronCore dan pemanfaatan vCPU, penggunaan memori, model yang dimuat, dan aplikasi Neuron dengan perintah. `neuron-top` Peluncuran tanpa argumen akan menampilkan data untuk semua aplikasi pembelajaran mesin yang memanfaatkan NeuronCores.

```
neuron-top
```

Ketika aplikasi menggunakan empat NeuronCores, output akan terlihat mirip dengan gambar berikut:

```

neuron-top
-----
Neuroncore Utilization
NC0 [ 100%] [ 100%] [ 100%] [ 100%]
NC1 [ 0.00%] [ 0.00%] [ 0.00%] [ 0.00%]
NC2 [ 0.00%] [ 0.00%] [ 0.00%] [ 0.00%]
NC3 [ 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/ 46.86B] 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

```

[Untuk informasi lebih lanjut tentang sumber daya untuk memantau dan mengoptimalkan aplikasi inferensi berbasis Neuron, lihat Alat Neuron.](#)

Menggunakan Neuron Monitor (neuron-monitor)

Neuron Monitor mengumpulkan metrik dari runtime Neuron yang berjalan di sistem dan mengalirkan data yang dikumpulkan ke stdout dalam format JSON. Metrik ini diatur ke dalam grup metrik yang Anda konfigurasi dengan menyediakan file konfigurasi. Untuk informasi lebih lanjut tentang Monitor Neuron, lihat [Panduan Pengguna untuk Monitor Neuron](#).

Meningkatkan Perangkat Lunak Neuron

[Untuk informasi tentang cara memperbarui perangkat lunak Neuron SDK dalam DLAMI, lihat AWS Panduan Pengaturan Neuron.](#)

Langkah Selanjutnya

[Menggunakan DLAMI dengan Neuron AWS](#)

Menggunakan DLAMI dengan Neuron AWS

Alur kerja khas dengan AWS Neuron SDK adalah mengkompilasi model pembelajaran mesin yang dilatih sebelumnya di server kompilasi. Setelah ini, distribusikan artefak ke instance Inf1 untuk dieksekusi. AWS Deep Learning AMIs (DLAMI) sudah diinstal sebelumnya dengan semua yang Anda butuhkan untuk mengkompilasi dan menjalankan inferensi dalam instance Inf1 yang menggunakan Inferentia.

Bagian berikut menjelaskan cara menggunakan DLAMI dengan Inferentia.

Daftar Isi

- [Menggunakan TensorFlow -Neuron dan Kompiler AWS Neuron](#)
- [Menggunakan TensorFlow Penyajian AWS Neuron](#)
- [Menggunakan MXNet -Neuron dan Kompiler AWS Neuron](#)
- [Menggunakan Penyajian Model MXNet -Neuron](#)
- [Menggunakan PyTorch -Neuron dan Kompiler AWS Neuron](#)

Menggunakan TensorFlow -Neuron dan Kompiler AWS Neuron

Tutorial ini menunjukkan cara menggunakan kompiler AWS Neuron untuk mengkompilasi model Keras ResNet -50 dan mengekspornya sebagai model yang disimpan dalam format. SavedModel Format ini adalah format TensorFlow model yang dapat dipertukarkan khas. Anda juga belajar cara menjalankan inferensi pada instance Inf1 dengan input contoh.

Untuk informasi lebih lanjut tentang Neuron SDK, lihat dokumentasi [AWS Neuron SDK](#).

Daftar Isi

- [Prasyarat](#)
- [Aktifkan lingkungan Conda](#)
- [Resnet50 Kompilasi](#)
- [ResNet50 Inferensi](#)

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan di [Meluncurkan Instance DLAMI dengan Neuron AWS](#). Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan DLAMI.

Aktifkan lingkungan Conda

Aktifkan lingkungan conda TensorFlow -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_tensorflow_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan perintah berikut:

```
source deactivate
```

Resnet50 Kompilasi

Buat skrip Python yang disebut **tensorflow_compile_resnet50.py** yang memiliki konten berikut. Skrip Python ini mengkompilasi model Keras ResNet 50 dan mengekspornya sebagai model yang disimpan.

```
import os
import time
import shutil
import tensorflow as tf
import tensorflow.neuron as tfn
import tensorflow.compat.v1.keras as keras
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input

# Create a workspace
WORKSPACE = './ws_resnet50'
os.makedirs(WORKSPACE, exist_ok=True)

# Prepare export directory (old one removed)
model_dir = os.path.join(WORKSPACE, 'resnet50')
compiled_model_dir = os.path.join(WORKSPACE, 'resnet50_neuron')
shutil.rmtree(model_dir, ignore_errors=True)
```

```
shutil.rmtree(compiled_model_dir, ignore_errors=True)

# Instantiate Keras ResNet50 model
keras.backend.set_learning_phase(0)
model = ResNet50(weights='imagenet')

# Export SavedModel
tf.saved_model.simple_save(
    session          = keras.backend.get_session(),
    export_dir       = model_dir,
    inputs           = {'input': model.inputs[0]},
    outputs          = {'output': model.outputs[0]})

# Compile using Neuron
tfn.saved_model.compile(model_dir, compiled_model_dir)

# Prepare SavedModel for uploading to Inf1 instance
shutil.make_archive(compiled_model_dir, 'zip', WORKSPACE, 'resnet50_neuron')
```

Kompilasi model menggunakan perintah berikut:

```
python tensorflow_compile_resnet50.py
```

Proses kompilasi akan memakan waktu beberapa menit. Ketika selesai, output Anda akan terlihat seperti berikut:

```
...
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
...
```

Setelah kompilasi, model yang disimpan di-zip di **ws_resnet50/resnet50_neuron.zip**. Buka zip model dan unduh gambar sampel untuk inferensi menggunakan perintah berikut:

```
unzip ws_resnet50/resnet50_neuron.zip -d .
curl -O https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg
```

ResNet50 Inferensi

Buat skrip Python yang disebut **tensorflow_infer_resnet50.py** yang memiliki konten berikut. Skrip ini menjalankan inferensi pada model yang diunduh menggunakan model inferensi yang dikompilasi sebelumnya.

```
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])
```

Jalankan inferensi pada model menggunakan perintah berikut:

```
python tensorflow_infer_resnet50.py
```

Output Anda akan terlihat seperti berikut:

...


```
[('n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159', 'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757', 'snow_leopard', 0.009290541)]
```

Langkah Selanjutnya

[Menggunakan TensorFlow Penyajian AWS Neuron](#)

Menggunakan TensorFlow Penyajian AWS Neuron

Tutorial ini menunjukkan cara membuat grafik dan menambahkan langkah kompilasi AWS Neuron sebelum mengekspor model yang disimpan untuk digunakan dengan TensorFlow Serving. TensorFlow Melayani adalah sistem penyajian yang memungkinkan Anda meningkatkan inferensi di seluruh jaringan. Neuron TensorFlow Serving menggunakan API yang sama dengan TensorFlow Serving normal. Satu-satunya perbedaan adalah bahwa model yang disimpan harus dikompilasi untuk AWS Inferentia dan titik masuknya adalah biner yang berbeda bernama `tensorflow_model_server_neuron`. Biner ditemukan di `/usr/local/bin/tensorflow_model_server_neuron` dan sudah diinstal sebelumnya di DLAMI.

Untuk informasi lebih lanjut tentang Neuron SDK, lihat dokumentasi [AWS Neuron SDK](#).

Daftar Isi

- [Prasyarat](#)
- [Aktifkan lingkungan Conda](#)
- [Kompilasi dan Ekspor Model Tersimpan](#)
- [Melayani Model Tersimpan](#)
- [Hasilkan permintaan inferensi ke server model](#)

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan di [Meluncurkan Instance DLAMI dengan Neuron AWS](#). Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan DLAMI.

Aktifkan lingkungan Conda

Aktifkan lingkungan conda TensorFlow -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_tensorflow_p36
```

Jika Anda perlu keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Kompilasi dan Ekspor Model Tersimpan

Buat skrip Python yang disebut `tensorflow-model-server-compile.py` dengan konten berikut. Skrip ini membuat grafik dan mengkompilasinya menggunakan Neuron. Kemudian mengekspor grafik yang dikompilasi sebagai model yang disimpan.

```
import tensorflow as tf
import tensorflow.neuron
import os

tf.keras.backend.set_learning_phase(0)
model = tf.keras.applications.ResNet50(weights='imagenet')
sess = tf.keras.backend.get_session()
inputs = {'input': model.inputs[0]}
outputs = {'output': model.outputs[0]}

# save the model using tf.saved_model.simple_save
modeldir = "./resnet50/1"
tf.saved_model.simple_save(sess, modeldir, inputs, outputs)

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

Kompilasi model menggunakan perintah berikut:

```
python tensorflow-model-server-compile.py
```

Output Anda akan terlihat seperti berikut:

```
...
```

```
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
```

Melayani Model Tersimpan

Setelah model dikompilasi, Anda dapat menggunakan perintah berikut untuk menyajikan model yang disimpan dengan biner tensorflow_model_server_neuron:

```
tensorflow_model_server_neuron --model_name=resnet50_inf1 \
  --model_base_path=$HOME/resnet50_inf1/ --port=8500 &
```

Output Anda akan terlihat seperti berikut ini. Model yang dikompilasi dipentaskan di DRAM perangkat Inferentia oleh server untuk mempersiapkan inferensi.

```
...
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 ...
```

Hasilkan permintaan inferensi ke server model

Membuat skrip Python yang disebut tensorflow-model-server-infer.py dengan konten berikut. Skrip ini menjalankan inferensi melalui gRPC, yang merupakan kerangka kerja layanan.

```
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/aws-labs/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))
```

Jalankan inferensi pada model dengan menggunakan gRPC dengan perintah berikut:

```
python tensorflow-model-server-infer.py
```

Output Anda akan terlihat seperti berikut:

```
[(['n02123045', 'tabby', 0.6918919), ('n02127052', 'lynx', 0.12770271), ('n02123159',
'tiger_cat', 0.08277027), ('n02124075', 'Egyptian_cat', 0.06418919), ('n02128757',
'snow_leopard', 0.009290541)]]
```

Menggunakan MXNet -Neuron dan Kompiler AWS Neuron

API kompilasi MXNet -Neuron menyediakan metode untuk mengkompilasi grafik model yang dapat Anda jalankan pada perangkat AWS Inferentia.

Dalam contoh ini, Anda menggunakan API untuk mengkompilasi model ResNet -50 dan menggunakannya untuk menjalankan inferensi.

Untuk informasi lebih lanjut tentang Neuron SDK, lihat dokumentasi [AWS Neuron SDK](#).

Daftar Isi

- [Prasyarat](#)
- [Aktifkan Lingkungan Conda](#)
- [Resnet50 Kompilasi](#)
- [ResNet50 Inferensi](#)

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan di [Meluncurkan Instance DLAMI dengan Neuron AWS](#). Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan DLAMI.

Aktifkan Lingkungan Conda

Aktifkan lingkungan conda MXNet -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_mxnet_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Resnet50 Kompilasi

Membuat skrip Python yang disebut **mxnet_compile_resnet50.py** dengan konten berikut. Skrip ini menggunakan kompilasi MXNet -Neuron Python API untuk mengkompilasi ResNet model -50.

```
import mxnet as mx
import numpy as np

print("downloading...")
path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
print("download finished.")

sym, args, aux = mx.model.load_checkpoint('resnet-50', 0)
```

```
print("compile for inferentia using neuron... this will take a few minutes...")
inputs = { "data" : mx.nd.ones([1,3,224,224], name='data', dtype='float32') }

sym, args, aux = mx.contrib.neuron.compile(sym, args, aux, inputs)

print("save compiled model...")
mx.model.save_checkpoint("compiled_resnet50", 0, sym, args, aux)
```

Kompilasi model menggunakan perintah berikut:

```
python mxnet_compile_resnet50.py
```

Kompilasi akan memakan waktu beberapa menit. Ketika kompilasi telah selesai, file-file berikut akan berada di direktori Anda saat ini:

```
resnet-50-0000.params
resnet-50-symbol.json
compiled_resnet50-0000.params
compiled_resnet50-symbol.json
```

ResNet50 Inferensi

Membuat skrip Python yang disebut **mxnet_infer_resnet50.py** dengan konten berikut. Skrip ini mengunduh gambar sampel dan menggunakannya untuk menjalankan inferensi dengan model yang dikompilasi.

```
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]))
```

Jalankan inferensi dengan model yang dikompilasi menggunakan perintah berikut:

```
python mxnet_infer_resnet50.py
```

Output Anda akan terlihat seperti berikut:

```
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
```

Langkah Selanjutnya

[Menggunakan Penyajian Model MXNet -Neuron](#)

Menggunakan Penyajian Model MXNet -Neuron

Dalam tutorial ini, Anda belajar menggunakan MXNet model pra-terlatih untuk melakukan klasifikasi gambar real-time dengan Multi Model Server (MMS). MMS adalah easy-to-use alat yang fleksibel dan untuk melayani model pembelajaran mendalam yang dilatih menggunakan pembelajaran mesin atau kerangka pembelajaran mendalam. Tutorial ini mencakup langkah kompilasi menggunakan AWS Neuron dan implementasi MMS menggunakan MXNet.

Untuk informasi lebih lanjut tentang Neuron SDK, lihat dokumentasi [AWS Neuron SDK](#).

Daftar Isi

- [Prasyarat](#)
- [Aktifkan Lingkungan Conda](#)
- [Unduh Kode Contoh](#)
- [Kompilasi Model](#)
- [Jalankan Inferensi](#)

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan di [Meluncurkan Instance DLAMI dengan Neuron AWS](#). Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan DLAMI.

Aktifkan Lingkungan Conda

Aktifkan lingkungan conda MXNet -Neuron dengan menggunakan perintah berikut:

```
source activate aws_neuron_mxnet_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Unduh Kode Contoh

Untuk menjalankan contoh ini, unduh kode contoh menggunakan perintah berikut:

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


Kompilasi Model

Membuat skrip Python yang disebut `multi-model-server-compile.py` dengan konten berikut. Skrip ini mengkompilasi model ResNet 50 ke target perangkat Inferentia.

```
import mxnet as mx
from mxnet.contrib import neuron
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'resnet/50-layers/resnet-50-0000.params')
mx.test_utils.download(path+'resnet/50-layers/resnet-50-symbol.json')
mx.test_utils.download(path+'synset.txt')

nn_name = "resnet-50"

#Load a model
sym, args, auxs = mx.model.load_checkpoint(nn_name, 0)

#Define compilation parameters# - input shape and dtype
inputs = {'data' : mx.nd.zeros([1,3,224,224], dtype='float32') }

# compile graph to inferentia target
csym, cargs, cauxs = neuron.compile(sym, args, auxs, inputs)

# save compiled model
mx.model.save_checkpoint(nn_name + "_compiled", 0, csym, cargs, cauxs)
```

Untuk mengkompilasi model, gunakan perintah berikut:

```
python multi-model-server-compile.py
```

Output Anda akan terlihat seperti berikut:

```
...
[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!
```

Buat file bernama `signature.json` dengan konten berikut untuk mengkonfigurasi nama input dan bentuk:

```
{
  "inputs": [
    {
      "data_name": "data",
      "data_shape": [
        1,
        3,
        224,
        224
      ]
    }
  ]
}
```

Download `synset.txt` file dengan menggunakan perintah berikut. File ini adalah daftar nama untuk kelas ImageNet prediksi.

```
curl -O https://s3.amazonaws.com/model-server/model_archive_1.0/examples/
squeeze_net_v1.1/synset.txt
```

Buat kelas layanan kustom mengikuti template di `model_server_template` folder. Salin template ke direktori kerja Anda saat ini dengan menggunakan perintah berikut:

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

Edit `mxnet_model_service.py` modul untuk mengganti `mx.cpu()` konteks dengan `mx.neuron()` konteks sebagai berikut. Anda juga perlu mengomentari salinan data yang tidak perlu `model_input` karena MXNet-Neuron tidak mendukung NDAarray dan APIs Gluon.

```
...
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]
```

Package model dengan `model-archiver` menggunakan perintah berikut:

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

Jalankan Inferensi

Mulai Multi Model Server dan muat model yang menggunakan RESTful API dengan menggunakan perintah berikut. Pastikan neuron-rtd itu berjalan dengan pengaturan default.

```
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
```

Jalankan inferensi menggunakan contoh gambar dengan perintah berikut:

```
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
```

Output Anda akan terlihat seperti berikut:

```
[
  {
    "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"  
}  
]
```

Untuk membersihkan setelah pengujian, keluarkan perintah delete melalui RESTful API dan hentikan server model menggunakan perintah berikut:

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

Anda akan melihat output berikut:

```
{  
  "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
```

Menggunakan PyTorch -Neuron dan Kompiler AWS Neuron

API kompilasi PyTorch -Neuron menyediakan metode untuk mengkompilasi grafik model yang dapat Anda jalankan pada perangkat AWS Inferentia.

Model terlatih harus dikompilasi ke target Inferentia sebelum dapat digunakan pada instance Inf1. Tutorial berikut mengkompilasi model torchvision ResNet 50 dan mengekspornya sebagai modul yang disimpan. TorchScript Model ini kemudian digunakan untuk menjalankan inferensi.

Untuk kenyamanan, tutorial ini menggunakan instance Inf1 untuk kompilasi dan inferensi. Dalam praktiknya, Anda dapat mengkompilasi model Anda menggunakan tipe instance lain, seperti keluarga instance c5. Anda kemudian harus menerapkan model yang dikompilasi ke server inferensi Inf1. Untuk informasi lebih lanjut, lihat [Dokumentasi AWS Neuron PyTorch SDK](#).

Daftar Isi

- [Prasyarat](#)
- [Aktifkan Lingkungan Conda](#)
- [Resnet50 Kompilasi](#)

- [ResNet50 Inferensi](#)

Prasyarat

Sebelum menggunakan tutorial ini, Anda seharusnya telah menyelesaikan langkah-langkah pengaturan di [Meluncurkan Instance DLAMI dengan Neuron AWS](#). Anda juga harus memiliki keakraban dengan pembelajaran mendalam dan menggunakan DLAMI.

Aktifkan Lingkungan Conda

Aktifkan lingkungan conda PyTorch -Neuron menggunakan perintah berikut:

```
source activate aws_neuron_pytorch_p36
```

Untuk keluar dari lingkungan conda saat ini, jalankan:

```
source deactivate
```

Resnet50 Kompilasi

Membuat skrip Python yang disebut **pytorch_trace_resnet50.py** dengan konten berikut. Skrip ini menggunakan kompilasi PyTorch -Neuron Python API untuk mengkompilasi ResNet model -50.

Note

Ada ketergantungan antara versi torchvision dan paket obor yang harus Anda ketahui saat mengkompilasi model torchvision. Aturan ketergantungan ini dapat dikelola melalui pip. Torchvision==0.6.1 cocok dengan rilis torch==1.5.1, sedangkan torchvision==0.8.2 cocok dengan rilis torch==1.7.1.

```
import torch
import numpy as np
import os
import torch_neuron
from torchvision import models

image = torch.zeros([1, 3, 224, 224], dtype=torch.float32)
```

```
## Load a pretrained ResNet50 model
model = models.resnet50(pretrained=True)

## Tell the model we are using it for evaluation (not training)
model.eval()
model_neuron = torch.neuron.trace(model, example_inputs=[image])

## Export to saved model
model_neuron.save("resnet50_neuron.pt")
```

Jalankan skrip kompilasi.

```
python pytorch_trace_resnet50.py
```

Kompilasi akan memakan waktu beberapa menit. Ketika kompilasi selesai, model yang dikompilasi disimpan seperti `resnet50_neuron.pt` di direktori lokal.

ResNet50 Inferensi

Membuat skrip Python yang disebut **pytorch_infer_resnet50.py** dengan konten berikut. Skrip ini mengunduh gambar sampel dan menggunakannya untuk menjalankan inferensi dengan model yang dikompilasi.

```
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) )
```

Jalankan inferensi dengan model yang dikompilasi menggunakan perintah berikut:

```
python pytorch_infer_resnet50.py
```

Output Anda akan terlihat seperti berikut:

```
Top 5 labels:  
['tiger', 'lynx', 'tiger_cat', 'Egyptian_cat', 'tabby']
```

ARM64 DLAMI

AWS ARM64 GPU DLAMIs dirancang untuk memberikan kinerja tinggi dan efisiensi biaya untuk beban kerja pembelajaran mendalam. Secara khusus, tipe instans G5G menampilkan [prosesor AWS Graviton2](#) berbasis ARM64, yang dibangun dari bawah ke atas AWS dan dioptimalkan untuk bagaimana pelanggan menjalankan beban kerja mereka di cloud. AWS ARM64 GPU sudah DLAMIs dikonfigurasi sebelumnya dengan Docker, NVIDIA Docker, NVIDIA Driver, CUDA, cuDNN, NCCL, serta kerangka kerja pembelajaran mesin populer seperti dan. TensorFlow PyTorch

Dengan tipe instans G5G, Anda dapat memanfaatkan manfaat harga dan kinerja Graviton2 untuk menerapkan model pembelajaran mendalam yang dipercepat GPU dengan biaya yang jauh lebih rendah jika dibandingkan dengan instans berbasis x86 dengan akselerasi GPU.

Pilih ARM64 DLAMI

Luncurkan [instans G5G](#) dengan ARM64 DLAMI pilihan Anda.

Untuk step-by-step petunjuk tentang meluncurkan DLAMI, [lihat Meluncurkan dan Mengonfigurasi DLAMI](#).

Untuk daftar yang terbaru ARM64 DLAMIs, lihat [Catatan Rilis untuk DLAMI](#).

Mulai

Topik berikut menunjukkan cara memulai menggunakan ARM64 DLAMI.

Daftar Isi

- [Menggunakan ARM64 DLAMI GPU PyTorch](#)

Menggunakan ARM64 DLAMI GPU PyTorch

AWS Deep Learning AMIs Ini siap digunakan dengan berbasis prosesor Arm64 GPUs, dan dioptimalkan untuk. PyTorch PyTorch DLAMI ARM64 GPU mencakup lingkungan Python yang

[PyTorch](#) telah dikonfigurasi sebelumnya [TorchVision](#) dengan, [TorchServe](#), dan untuk pelatihan pembelajaran mendalam dan kasus penggunaan inferensi.

Daftar Isi

- [Verifikasi PyTorch Lingkungan Python](#)
- [Jalankan Sampel Pelatihan dengan PyTorch](#)
- [Jalankan Sampel Inferensi dengan PyTorch](#)

Verifikasi PyTorch Lingkungan Python

Hubungkan ke instans G5G Anda dan aktifkan lingkungan dasar Conda dengan perintah berikut:

```
source activate base
```

Prompt perintah Anda harus menunjukkan bahwa Anda bekerja di lingkungan dasar Conda, yang berisi PyTorch TorchVision, dan pustaka lainnya.

```
(base) $
```

Verifikasi jalur alat default PyTorch lingkungan:

```
(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)
```

Jalankan Sampel Pelatihan dengan PyTorch

Jalankan contoh pekerjaan pelatihan MNIST:

```
git clone https://github.com/pytorch/examples.git
cd examples/mnist
python main.py
```

Output-nya semestinya mirip dengan yang berikut:

```
...
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%)
```

Jalankan Sampel Inferensi dengan PyTorch

Gunakan perintah berikut untuk mengunduh model densenet161 yang telah dilatih sebelumnya dan jalankan inferensi menggunakan: 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
```

Output-nya semestinya mirip dengan yang berikut:

```
{
  "tiger_cat": 0.4693363308906555,
  "tabby": 0.4633873701095581,
  "Egyptian_cat": 0.06456123292446136,
  "lynx": 0.0012828150065615773,
  "plastic_bag": 0.00023322898778133094
}
```

Gunakan perintah berikut untuk membatalkan pendaftaran model densenet161 dan menghentikan server:

```
curl -X DELETE http://localhost:8081/models/densenet161/1.0
torchserve --stop
```

Output-nya semestinya mirip dengan yang berikut:

```
{
  "status": "Model \"densenet161\" unregistered"
}
TorchServe has stopped.
```

Inferensi

Bagian ini menyediakan tutorial tentang cara menjalankan inferensi menggunakan kerangka kerja dan alat DLAMI.

Alat Inferensi

- [TensorFlow Melayani](#)

Penyajian Model

Berikut ini adalah opsi penyajian model yang diinstal pada AMI Pembelajaran Mendalam dengan Conda. Klik salah satu opsi untuk mempelajari cara menggunakannya.

Topik

- [TensorFlow Melayani](#)
- [TorchServe](#)

TensorFlow Melayani

[TensorFlow Melayani](#) adalah sistem penyajian yang fleksibel dan berkinerja tinggi untuk model pembelajaran mesin.

`tensorflow-serving-api` ini sudah diinstal sebelumnya dengan DLAMI framework tunggal. Untuk menggunakan penyajian tensorflow, aktifkan lingkungan terlebih dahulu. TensorFlow

```
$ source /opt/tensorflow/bin/activate
```

Kemudian gunakan editor teks pilihan Anda untuk membuat skrip yang memiliki konten berikut. Nama `itutest_train_mnist.py`. Skrip ini direferensikan dari [TensorFlow Tutorial](#) yang akan melatih dan mengevaluasi model pembelajaran mesin jaringan saraf yang mengklasifikasikan gambar.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Sekarang jalankan skrip melewati lokasi server dan port dan nama file foto husky sebagai parameter.

```
$ /opt/tensorflow/bin/python3 test_train_mnist.py
```

Bersabarlah, karena skrip ini mungkin memakan waktu beberapa saat sebelum memberikan output apa pun. Ketika pelatihan selesai, Anda harus melihat yang berikut:

```
I0000 00:00:1739482012.389276    4284 device_compiler.h:188] Compiled cluster using
XLA! This line is logged at most once for the lifetime of the process.
1875/1875 [=====] - 24s 2ms/step - loss: 0.2973 - accuracy:
0.9134
Epoch 2/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.1422 - accuracy:
0.9582
Epoch 3/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.1076 - accuracy:
0.9687
Epoch 4/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.0872 - accuracy:
0.9731
Epoch 5/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0731 - accuracy:
0.9771
313/313 [=====] - 0s 1ms/step - loss: 0.0749 - accuracy:
0.9780
```

Lebih Banyak Fitur dan Contoh

Jika Anda tertarik untuk mempelajari lebih lanjut tentang TensorFlow Melayani, lihat [TensorFlow situs webnya](#).

TorchServe

TorchServe adalah alat yang fleksibel untuk melayani model pembelajaran mendalam yang telah diekspor dari PyTorch. TorchServe datang pra-instal dengan AMI Pembelajaran Mendalam dengan Conda.

Untuk informasi selengkapnya tentang penggunaan TorchServe, lihat [Server Model untuk PyTorch Dokumentasi](#).

Topik

Sajikan Model Klasifikasi Gambar pada TorchServe

Tutorial ini menunjukkan cara menyajikan model klasifikasi gambar dengan TorchServe. Ini menggunakan model DenseNet -161 yang disediakan oleh PyTorch. Setelah server berjalan, ia mendengarkan permintaan prediksi. Saat Anda mengunggah gambar, dalam hal ini, gambar anak kucing, server mengembalikan prediksi 5 kelas pencocokan teratas dari kelas tempat model dilatih.

Untuk menyajikan contoh model klasifikasi gambar pada TorchServe

1. Connect ke instans Amazon Elastic Compute Cloud (Amazon EC2) dengan Deep Learning AMI dengan Conda v34 atau versi lebih baru.
2. Aktifkan `pytorch_p310` lingkungan.

```
source activate pytorch_p310
```

3. Kloning TorchServe repositori, lalu buat direktori untuk menyimpan model Anda.

```
git clone https://github.com/pytorch/serve.git
mkdir model_store
```

4. Arsipkan model menggunakan pengarsipan model. `extra-files`Param menggunakan file dari TorchServe repo, jadi perbarui jalur jika perlu. Untuk informasi selengkapnya tentang pengarsipan model, lihat [Pengarsip Model Obor](#) untuk 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. Jalankan TorchServe untuk memulai titik akhir. Menambahkan `> /dev/null` menenangkan output log.

```
torchserve --start --ncs --model-store model_store --models densenet161.mar > /dev/
null
```

6. Unduh gambar anak kucing dan kirimkan ke titik akhir TorchServe prediksi:

```
curl -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg
curl http://127.0.0.1:8080/predictions/densenet161 -T kitten.jpg
```

Titik akhir prediksi mengembalikan prediksi di JSON yang mirip dengan lima prediksi teratas berikut, di mana gambar memiliki probabilitas 47% mengandung kucing Mesir, diikuti oleh kemungkinan 46% memiliki kucing kucing kucing.

```
{
  "tiger_cat": 0.46933576464653015,
  "tabby": 0.463387668132782,
  "Egyptian_cat": 0.0645613968372345,
  "lynx": 0.0012828196631744504,
  "plastic_bag": 0.00023323058849200606
}
```

7. Setelah Anda selesai menguji, hentikan server:

```
torchserve --stop
```

Contoh Lain

TorchServe memiliki berbagai contoh yang dapat Anda jalankan pada instance DLAMI Anda. Anda dapat melihatnya di halaman [contoh repositori TorchServe proyek](#).

Info Lebih Lanjut

Untuk TorchServe dokumentasi lebih lanjut, termasuk cara mengatur TorchServe dengan Docker dan TorchServe fitur terbaru, lihat [halaman TorchServe proyek](#) di GitHub.

Meningkatkan DLAMI Anda

Di sini Anda akan menemukan informasi tentang meningkatkan DLAMI Anda dan tips memperbarui perangkat lunak pada DLAMI Anda.

Selalu perbarui sistem operasi Anda dan perangkat lunak terinstal lainnya dengan menerapkan tambalan dan pembaruan segera setelah tersedia.

Jika Anda menggunakan Amazon Linux atau Ubuntu, ketika Anda masuk ke DLAMI Anda, Anda akan diberi tahu jika pembaruan tersedia dan melihat instruksi untuk memperbarui. Untuk informasi lebih lanjut tentang pemeliharaan Amazon Linux, lihat [Memperbarui Perangkat Lunak Instans](#). Untuk contoh Ubuntu, lihat [dokumentasi resmi Ubuntu](#).

Di Windows, periksa Pembaruan Windows secara teratur untuk pembaruan perangkat lunak dan keamanan. Jika Anda mau, sediakan pembaruan secara otomatis.

Important

[Untuk informasi tentang kerentanan Meltdown dan Spectre dan cara menambal sistem operasi Anda untuk mengatasinya, lihat Buletin Keamanan -2018-013. AWS](#)

Topik

- [Upgrade ke Versi DLAMI Baru](#)
- [Kiat untuk Pembaruan Perangkat Lunak](#)
- [Menerima Pemberitahuan tentang Pembaruan Baru](#)

Upgrade ke Versi DLAMI Baru

Gambar sistem DLAMI diperbarui secara teratur untuk memanfaatkan rilis kerangka pembelajaran mendalam baru, CUDA dan pembaruan perangkat lunak lainnya, dan penyetelan kinerja. Jika Anda telah menggunakan DLAMI untuk beberapa waktu dan ingin memanfaatkan pembaruan, Anda perlu meluncurkan instance baru. Anda juga harus mentransfer kumpulan data, pos pemeriksaan, atau data berharga lainnya secara manual. Sebagai gantinya, Anda dapat menggunakan Amazon EBS untuk menyimpan data Anda dan melampirkannya ke DLAMI baru. Dengan cara ini, Anda dapat sering meningkatkan, sambil meminimalkan waktu yang diperlukan untuk mentransisikan data Anda.

Note

Saat melampirkan dan memindahkan volume Amazon EBS di antaranya DLAMIs, Anda harus memiliki volume DLAMIs dan volume baru di Availability Zone yang sama.

1. Gunakan Amazon EC2console untuk membuat volume Amazon EBS baru. Untuk petunjuk mendetail, lihat [Membuat Volume Amazon EBS](#).
2. Lampirkan volume Amazon EBS yang baru dibuat ke DLAMI yang sudah ada. Untuk petunjuk mendetail, lihat [Melampirkan Volume Amazon EBS](#).
3. Transfer data Anda, seperti dataset, pos pemeriksaan, dan file konfigurasi.
4. Luncurkan DLAMI. Untuk petunjuk terperinci, lihat [Menyiapkan instance DLAMI](#).
5. Lepaskan volume Amazon EBS dari DLAMI lama Anda. Untuk petunjuk mendetail, lihat [Melepaskan Volume Amazon EBS](#).
6. Lampirkan volume Amazon EBS ke DLAMI baru Anda. Ikuti instruksi dari Langkah 2 untuk melampirkan volume.
7. Setelah Anda memverifikasi bahwa data Anda tersedia di DLAMI baru Anda, hentikan dan hentikan DLAMI lama Anda. Untuk instruksi pembersihan terperinci, lihat. [Membersihkan contoh DLAMI](#)

Kiat untuk Pembaruan Perangkat Lunak

Dari waktu ke waktu, Anda mungkin ingin memperbarui perangkat lunak secara manual pada DLAMI Anda. Hal ini umumnya dianjurkan bahwa Anda menggunakan pip untuk memperbarui paket Python. Anda juga harus menggunakan pip untuk memperbarui paket dalam lingkungan Conda pada AMI Pembelajaran Mendalam dengan Conda. Lihat situs web kerangka kerja atau perangkat lunak tertentu untuk instruksi peningkatan dan penginstalan.

Note

Kami tidak dapat menjamin bahwa pembaruan paket akan berhasil. Mencoba memperbarui paket di lingkungan dengan dependensi yang tidak kompatibel dapat mengakibatkan kegagalan. Dalam kasus seperti itu, Anda harus menghubungi pengelola perpustakaan untuk melihat apakah mungkin untuk memperbarui dependensi paket. Atau, Anda dapat mencoba memodifikasi lingkungan sedemikian rupa sehingga memungkinkan pembaruan. Namun,

modifikasi ini kemungkinan berarti menghapus atau memperbarui paket yang ada, yang berarti bahwa kami tidak dapat lagi menjamin stabilitas lingkungan ini.

AWS Deep Learning AMIs Muncul dengan banyak lingkungan Conda dan banyak paket yang sudah diinstal sebelumnya. Karena jumlah paket yang sudah diinstal sebelumnya, sulit menemukan satu set paket yang dijamin kompatibel. Anda mungkin melihat peringatan “Lingkungan tidak konsisten, silakan periksa paket paket dengan cermat”. DLAMI memastikan bahwa semua lingkungan yang disediakan DLAMI sudah benar, tetapi tidak dapat menjamin bahwa setiap paket yang diinstal pengguna akan berfungsi dengan benar.

Menerima Pemberitahuan tentang Pembaruan Baru

Note

AWS Deep Learning AMIs memiliki irama rilis mingguan untuk patch keamanan. Pemberitahuan rilis akan dikirim untuk tambalan keamanan tambahan ini meskipun mungkin tidak disertakan dalam catatan rilis resmi.

Anda dapat menerima pemberitahuan setiap kali DLAMI baru dirilis. Pemberitahuan diterbitkan dengan [Amazon SNS](#) menggunakan topik berikut.

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

Pesan diposting di sini ketika DLAMI baru diterbitkan. Versi, metadata, dan ID AMI regional dari AMI akan disertakan dalam pesan.

Pesan-pesan ini dapat diterima menggunakan beberapa metode berbeda. Kami menyarankan Anda menggunakan metode berikut.

1. Buka [konsol Amazon SNS](#).
2. Di bilah navigasi, ubah AWS Wilayah ke AS Barat (Oregon), jika perlu. Anda harus memilih wilayah tempat notifikasi SNS yang Anda berlangganan dibuat.
3. Di panel navigasi, pilih Langganan, Buat langganan.
4. Untuk kotak dialog Buat langganan, lakukan hal berikut:

- a. Untuk Topik ARN, salin dan tempel Nama Sumber Daya Amazon (ARN) berikut:
arn:aws:sns:us-west-2:767397762724:diami-updates
 - b. Untuk Protokol, pilih salah satu dari [Amazon SQS, AWS Lambda, Email, Email-JSON]
 - c. Untuk Endpoint, masukkan alamat email atau Nama Sumber Daya Amazon (ARN) sumber daya yang akan Anda gunakan untuk menerima notifikasi.
 - d. Pilih Buat langganan.
5. Anda menerima email konfirmasi dengan baris subjek AWS Pemberitahuan - Konfirmasi Langganan. Buka email dan pilih Konfirmasi berlangganan untuk menyelesaikan langganan Anda.

Keamanan di AWS Deep Learning AMIs

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model tanggung jawab bersama](#) menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari [Program AWS Kepatuhan Program AWS Kepatuhan](#) . Untuk mempelajari tentang program kepatuhan yang berlaku AWS Deep Learning AMIs, lihat [AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan](#) .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan DLAMI. Topik berikut menunjukkan cara mengonfigurasi DLAMI untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan sumber daya DLAMI Anda.

Untuk informasi selengkapnya, lihat [Keamanan EC2 di Amazon](#) di Panduan EC2 Pengguna Amazon.

Topik

- [Perlindungan data di AWS Deep Learning AMIs](#)
- [Identitas dan manajemen akses untuk AWS Deep Learning AMIs](#)
- [Validasi kepatuhan untuk AWS Deep Learning AMIs](#)
- [Ketahanan di AWS Deep Learning AMIs](#)
- [Keamanan infrastruktur di AWS Deep Learning AMIs](#)
- [AWS Deep Learning AMIs Contoh pemantauan](#)

Perlindungan data di AWS Deep Learning AMIs

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di AWS Deep Learning AMIs. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensial dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat [Bekerja dengan CloudTrail jejak](#) di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-3](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan DLAMI atau Layanan AWS lainnya menggunakan konsol, API AWS CLI, atau AWS SDKs Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan

atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Identitas dan manajemen akses untuk AWS Deep Learning AMIs

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya DLAMI. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Untuk informasi selengkapnya tentang identitas dan manajemen akses, lihat [Identitas dan manajemen akses untuk Amazon EC2](#).

Topik

- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [IAM dengan Amazon EMR](#)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensial Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS](#).

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara

kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Guna mengetahui informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [AWS Signature Version 4 untuk permintaan API](#) dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Autentikasi multi-faktor AWS di IAM](#) dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat meminta kelompok untuk menyebutkan IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat [Kasus penggunaan untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Untuk mengambil peran IAM sementara AWS Management Console, Anda dapat [beralih dari pengguna ke peran IAM \(konsol\)](#). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat [Metode untuk mengambil peran](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Buat peran untuk penyedia identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM](#) dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin

melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.

- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).
- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS](#) dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS permintaan API. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di Panduan Pengguna IAM](#).

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan

diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam Akun AWS. Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat [Pilih antara kebijakan yang dikelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya,

administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus [menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnya ACLs, lihat [Ringkasan daftar kontrol akses \(ACL\)](#) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS

Untuk informasi selengkapnya tentang Organizations dan SCPs, lihat [Kebijakan kontrol layanan](#) di Panduan AWS Organizations Pengguna.

- Kebijakan kontrol sumber daya (RCPs) — RCPs adalah kebijakan JSON yang dapat Anda gunakan untuk menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda tanpa memperbarui kebijakan IAM yang dilampirkan ke setiap sumber daya yang Anda miliki. RCP membatasi izin untuk sumber daya di akun anggota dan dapat memengaruhi izin efektif untuk identitas, termasuk Pengguna root akun AWS, terlepas dari apakah itu milik organisasi Anda. Untuk informasi selengkapnya tentang Organizations dan RCPs, termasuk daftar dukungan Layanan AWS tersebut RCPs, lihat [Kebijakan kontrol sumber daya \(RCPs\)](#) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

IAM dengan Amazon EMR

Anda dapat menggunakan IAM dengan Amazon EMR untuk menentukan pengguna AWS, sumber daya, grup, peran, dan kebijakan. Anda juga dapat mengontrol pengguna dan peran mana Layanan AWS yang dapat diakses.

Untuk informasi selengkapnya tentang penggunaan IAM dengan Amazon EMR, lihat [Amazon AWS Identity and Access Management EMR](#).

Validasi kepatuhan untuk AWS Deep Learning AMLs

Auditor pihak ketiga menilai keamanan dan kepatuhan AWS Deep Learning AMLs sebagai bagian dari beberapa program AWS kepatuhan. Untuk informasi tentang program kepatuhan yang didukung, lihat [Validasi kepatuhan untuk Amazon EC2](#).

Untuk daftar cakupan program kepatuhan tertentu, lihat [AWS Layanan dalam Lingkup menurut AWS Layanan Program Kepatuhan](#) . Layanan AWS Untuk informasi umum, lihat [Program AWS Kepatuhan Program AWS](#) .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di Laporan Artefak](#) di. AWS Artifact

Tanggung jawab kepatuhan Anda saat menggunakan DLAMI ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- [Panduan Quick Start Keamanan dan Kepatuhan](#) – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah-langkah untuk melakukan deployment terhadap lingkungan dasar di AWS yang menjadi fokus keamanan dan kepatuhan.
- [AWS Sumber Daya AWS](#) — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- [Mengevaluasi Sumber Daya dengan AWS Config Aturan](#) dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- [AWS Security Hub](#)— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk mengevaluasi AWS sumber daya Anda dan untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di AWS Deep Learning AMIs

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat [Infrastruktur AWS Global](#).

Untuk informasi tentang EC2 fitur Amazon untuk membantu mendukung ketahanan data dan kebutuhan pencadangan, lihat [Ketahanan di Amazon EC2 di Panduan Pengguna Amazon EC2](#) .

Keamanan infrastruktur di AWS Deep Learning AMIs

Keamanan infrastruktur AWS Deep Learning AMIs didukung oleh Amazon EC2. Untuk informasi selengkapnya, lihat [Keamanan infrastruktur EC2 di Amazon](#) di Panduan EC2 Pengguna Amazon.

AWS Deep Learning AMIs Contoh pemantauan

Pemantauan adalah bagian penting untuk menjaga keandalan, ketersediaan, dan kinerja AWS Deep Learning AMIs instans Anda dan AWS solusi Anda yang lain. Instans DLAMI Anda dilengkapi dengan beberapa alat pemantauan GPU, termasuk utilitas yang melaporkan statistik penggunaan GPU ke Amazon. CloudWatch Untuk informasi selengkapnya, lihat [Pemantauan dan Optimasi GPU](#), dan lihat [Memantau EC2 sumber daya Amazon](#) di Panduan EC2 Pengguna Amazon.

Memilih keluar dari pelacakan penggunaan untuk instans DLAMI

Distribusi sistem AWS Deep Learning AMIs operasi berikut mencakup kode yang memungkinkan AWS untuk mengumpulkan jenis instance, ID instance, tipe DLAMI, dan informasi OS.

Note

AWS tidak mengumpulkan atau menyimpan informasi lain tentang DLAMI, seperti perintah yang Anda gunakan dalam DLAMI.

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

Untuk memilih keluar dari pelacakan penggunaan

Jika Anda memilih, Anda dapat memilih keluar dari pelacakan penggunaan untuk instans DLAMI baru. Untuk memilih keluar, Anda harus menambahkan tag ke EC2 instans Amazon Anda selama peluncuran. Tag harus menggunakan kunci OPT_OUT_TRACKING dengan nilai terkait yang disetel

`ketrue`. Untuk informasi selengkapnya, lihat [Menandai EC2 sumber daya Amazon Anda](#) di Panduan EC2 Pengguna Amazon.

Kebijakan dukungan kerangka kerja DLAMI

Di sini Anda dapat menemukan detail kebijakan dukungan untuk kerangka kerja AWS Deep Learning AMIs (DLAMI).

[Untuk daftar kerangka kerja DLAMI yang saat ini mendukung, lihat halaman DLAMI Framework AWS Support Policy.](#) Dalam tabel di halaman itu, ingatlah hal-hal berikut:

- Versi saat ini menentukan versi kerangka kerja dalam format x.y.z. Dalam format ini, x mengacu pada versi utama, y mengacu pada versi minor, dan z mengacu pada versi patch. Misalnya, untuk TensorFlow 2.10.1, versi utama adalah 2, versi minor adalah 10, dan versi patch adalah 1.
- Akhir tambalan menentukan berapa lama AWS mendukung versi kerangka kerja itu.

Untuk informasi rinci tentang spesifik DLAMIs, lihat [Catatan rilis untuk DLAMIs](#).

Dukungan kerangka kerja DLAMI FAQs

- [Versi kerangka kerja apa yang mendapatkan tambalan keamanan?](#)
- [Gambar apa yang AWS dipublikasikan saat versi kerangka kerja baru dirilis?](#)
- [Gambar apa yang mendapatkan AWS fitur SageMaker AI/baru?](#)
- [Bagaimana versi saat ini didefinisikan dalam tabel Kerangka Kerja yang Didukung?](#)
- [Bagaimana jika saya menjalankan versi yang tidak ada dalam tabel Kerangka Kerja yang Didukung?](#)
- [Apakah DLAMIs mendukung versi sebelumnya TensorFlow?](#)
- [Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?](#)
- [Seberapa sering gambar baru dirilis?](#)
- [Apakah instance saya akan ditambah di tempat saat beban kerja saya berjalan?](#)
- [Apa yang terjadi ketika versi kerangka kerja baru yang ditambah atau diperbarui tersedia?](#)
- [Apakah dependensi diperbarui tanpa mengubah versi kerangka kerja?](#)
- [Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?](#)
- [Akankah gambar dengan versi kerangka kerja yang tidak lagi dipelihara secara aktif ditambah?](#)
- [Bagaimana cara menggunakan versi kerangka kerja yang lebih lama?](#)

- [Bagaimana cara saya tetap up-to-date dengan perubahan dukungan dalam kerangka kerja dan versinya?](#)
- [Apakah saya memerlukan lisensi komersial untuk menggunakan Repositori Anaconda?](#)

Versi kerangka kerja apa yang mendapatkan tambalan keamanan?

Jika versi framework diberi label Supported dalam [tabel AWS Deep Learning AMIs Framework Support Policy](#), maka akan mendapat patch keamanan.

Gambar apa yang AWS dipublikasikan saat versi kerangka kerja baru dirilis?

Kami menerbitkan baru DLAMIs segera setelah versi baru TensorFlow dan PyTorch dirilis. Ini termasuk versi mayor, versi mayor-minor, dan major-minor-patch versi kerangka kerja. Kami juga memperbarui gambar saat versi driver dan pustaka baru tersedia. Untuk informasi lebih lanjut tentang pemeliharaan gambar, lihat [Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?](#)

Gambar apa yang mendapatkan AWS fitur SageMaker AI/baru?

Fitur baru biasanya dirilis dalam versi terbaru DLAMIs untuk PyTorch dan TensorFlow. Lihat catatan rilis untuk gambar tertentu untuk detail tentang SageMaker AI atau AWS fitur baru. Untuk daftar yang tersedia DLAMIs, lihat [Catatan Rilis untuk DLAMI](#). Untuk informasi lebih lanjut tentang pemeliharaan gambar, lihat [Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?](#)

Bagaimana versi saat ini didefinisikan dalam tabel Kerangka Kerja yang Didukung?

Versi saat ini dalam [tabel AWS Deep Learning AMIs Framework Support Policy](#) mengacu pada versi framework terbaru yang AWS tersedia di GitHub. Setiap rilis terbaru mencakup pembaruan untuk driver, perpustakaan, dan paket yang relevan di DLAMI. Untuk informasi tentang pemeliharaan gambar, lihat [Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?](#)

Bagaimana jika saya menjalankan versi yang tidak ada dalam tabel Kerangka Kerja yang Didukung?

Jika Anda menjalankan versi yang tidak ada dalam [tabel Kebijakan Dukungan AWS Deep Learning AMIs Kerangka Kerja](#), Anda mungkin tidak memiliki driver, pustaka, dan paket yang relevan yang

paling diperbarui. Untuk up-to-date versi lainnya, kami sarankan Anda meningkatkan ke salah satu kerangka kerja yang didukung yang tersedia menggunakan DLAMI terbaru pilihan Anda. Untuk daftar yang tersedia DLAMIs, lihat [Catatan Rilis untuk DLAMI](#).

Apakah DLAMIs mendukung versi sebelumnya TensorFlow?

Tidak. Kami mendukung versi patch terbaru dari setiap versi utama terbaru framework yang dirilis 365 hari dari GitHub rilis awal seperti yang dinyatakan dalam [tabel AWS Deep Learning AMIs Framework Support Policy](#). Untuk informasi selengkapnya, silakan lihat [Bagaimana jika saya menjalankan versi yang tidak ada dalam tabel Kerangka Kerja yang Didukung?](#)

Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?

[Untuk menggunakan DLAMI dengan versi kerangka kerja terbaru, ambil ID DLAMI dan gunakan untuk meluncurkan DLAMI menggunakan Konsol. EC2. Untuk contoh perintah AWS CLI untuk mengambil AWS Deep Learning AMIs ID, lihat halaman catatan rilis DLAMI catatan rilis kerangka tunggal DLAMI catatan rilis.](#) Versi framework yang Anda pilih harus diberi label Supported dalam [tabel AWS Deep Learning AMIs Framework Support Policy](#).

Seberapa sering gambar baru dirilis?

Menyediakan versi tambalan yang diperbarui adalah prioritas tertinggi kami. Kami secara rutin membuat gambar yang ditambal pada kesempatan paling awal. Kami memantau versi kerangka kerja yang baru ditambal (mis. TensorFlow 2.9 hingga TensorFlow 2.9.1) dan versi rilis minor baru (mis. TensorFlow 2.9 hingga TensorFlow 2.10) dan membuatnya tersedia pada kesempatan paling awal. Ketika versi yang TensorFlow ada dirilis dengan versi baru CUDA, kami merilis DLAMI baru untuk versi tersebut dengan dukungan untuk versi TensorFlow CUDA baru.

Apakah instance saya akan ditambal di tempat saat beban kerja saya berjalan?

Tidak. Pembaruan tambalan untuk DLAMI bukan pembaruan “di tempat”.

Anda harus mengaktifkan EC2 instance baru, memigrasikan beban kerja dan skrip, lalu mematikan instance sebelumnya.

Apa yang terjadi ketika versi kerangka kerja baru yang ditambal atau diperbarui tersedia?

Periksa halaman catatan rilis secara teratur untuk gambar Anda. Kami mendorong Anda untuk meningkatkan ke kerangka kerja baru yang ditambal atau diperbarui saat tersedia. Untuk daftar yang tersedia DLAMIs, lihat [Catatan Rilis untuk DLAMI](#).

Apakah dependensi diperbarui tanpa mengubah versi kerangka kerja?

Kami memperbarui dependensi tanpa mengubah versi kerangka kerja. Namun, jika pembaruan ketergantungan menyebabkan ketidakcocokan, kami membuat gambar dengan versi yang berbeda. Pastikan untuk memeriksa [Catatan Rilis untuk DLAMI untuk](#) informasi ketergantungan yang diperbarui.

Kapan dukungan aktif untuk versi kerangka kerja saya berakhir?

Gambar DLAMI tidak dapat diubah. Begitu mereka diciptakan, mereka tidak berubah. Ada empat alasan utama mengapa dukungan aktif untuk versi kerangka kerja berakhir:

- [Versi kerangka kerja \(patch\) upgrade](#)
- [AWS patch keamanan](#)
- [Akhir tanggal tambalan \(Aging out\)](#)
- [Ketergantungan end-of-support](#)

Note

Karena frekuensi upgrade versi patch dan patch keamanan, kami sarankan memeriksa halaman catatan rilis untuk DLAMI Anda sering, dan upgrade ketika perubahan dilakukan.

Versi kerangka kerja (patch) upgrade

Jika Anda memiliki beban kerja DLAMI TensorFlow berdasarkan 2.7.0 TensorFlow dan merilis versi 2.7.1, kemudian merilis DLAMI baru dengan 2.7.1 GitHub. AWS TensorFlow Gambar sebelumnya dengan 2.7.0 tidak lagi aktif dipertahankan setelah gambar baru dengan TensorFlow 2.7.1 dirilis. DLAMI TensorFlow dengan 2.7.0 tidak menerima tambalan lebih lanjut. Halaman catatan rilis DLAMI

TensorFlow untuk 2.7 kemudian diperbarui dengan informasi terbaru. Tidak ada halaman catatan rilis individual untuk setiap tambalan kecil.

Baru DLAMIs dibuat karena upgrade patch ditetapkan dengan [ID AMI](#) baru.

AWS patch keamanan

Jika Anda memiliki beban kerja berdasarkan gambar dengan TensorFlow 2.7.0 dan AWS membuat patch keamanan, maka versi baru DLAMI dirilis untuk 2.7.0. TensorFlow Versi gambar sebelumnya dengan TensorFlow 2.7.0 tidak lagi dipertahankan secara aktif. Untuk informasi selengkapnya, lihat [Apakah instance saya akan ditambal di tempat saat beban kerja saya berjalan?](#) Untuk langkah-langkah menemukan DLAMI terbaru, lihat [Bagaimana cara menemukan gambar tambalan terbaru untuk versi kerangka kerja yang didukung?](#)

Baru DLAMIs dibuat karena upgrade patch ditetapkan dengan [ID AMI](#) baru.

Akhir tanggal tambalan (Aging out)

DLAMIs mencapai akhir tanggal patch mereka 365 hari setelah tanggal GitHub rilis.

Untuk [multi-kerangka kerja DLAMIs](#), ketika salah satu versi kerangka diperbarui, DLAMI baru dengan versi yang diperbarui diperlukan. DLAMI dengan versi kerangka kerja lama tidak lagi dipertahankan secara aktif.

Important

Kami membuat pengecualian ketika ada pembaruan kerangka kerja utama. Sebagai contoh, jika TensorFlow 1.15 memperbarui ke TensorFlow 2.0, maka kami terus mendukung versi terbaru TensorFlow 1.15 untuk jangka waktu dua tahun sejak tanggal GitHub rilis atau enam bulan setelah tim pemeliharaan kerangka kerja asal menjatuhkan dukungan, tanggal mana pun yang lebih awal.

Ketergantungan end-of-support

Jika Anda menjalankan beban kerja pada gambar DLAMI TensorFlow 2.7.0 dengan Python 3.6 dan versi Python ditandai, end-of-support maka semua gambar DLAMI berdasarkan Python 3.6 tidak akan lagi dipertahankan secara aktif. Demikian pula, jika versi OS seperti Ubuntu 16.04 ditandai untuk end-of-support, maka semua gambar DLAMI yang bergantung pada Ubuntu 16.04 tidak akan lagi dipertahankan secara aktif.

Akankah gambar dengan versi kerangka kerja yang tidak lagi dipelihara secara aktif ditambah?

Tidak. Gambar yang tidak lagi dipelihara secara aktif tidak akan memiliki rilis baru.

Bagaimana cara menggunakan versi kerangka kerja yang lebih lama?

[Untuk menggunakan DLAMI dengan versi kerangka kerja yang lebih lama, ambil ID DLAMI dan gunakan untuk meluncurkan DLAMI menggunakan Konsol. EC2](#) Untuk perintah AWS CLI untuk mengambil ID AMI, lihat halaman catatan rilis dalam catatan rilis [DLAMI](#) kerangka tunggal.

Bagaimana cara saya tetap up-to-date dengan perubahan dukungan dalam kerangka kerja dan versinya?

[Tetap up-to-date dengan kerangka kerja dan versi DLAMI menggunakan tabel Kebijakan Dukungan AWS Deep Learning AMIs Kerangka Kerja, catatan rilis DLAMI.](#)

Apakah saya memerlukan lisensi komersial untuk menggunakan Repositori Anaconda?

Anaconda beralih ke model lisensi komersial untuk pengguna tertentu. Dipelihara secara aktif DLAMIs telah dimigrasikan ke versi open-source Conda ([conda-forge](#)) yang tersedia untuk umum dari saluran Anaconda.

Driver NVIDIA penting berubah menjadi DLAMIs

Pada 15 November 2023, AWS membuat perubahan penting pada AWS Deep Learning AMIs (DLAMI) terkait dengan driver NVIDIA yang menggunakan. DLAMIs Untuk informasi tentang apa yang berubah dan apakah itu memengaruhi penggunaan Anda DLAMIs, lihat [Perubahan driver DLAMI NVIDIA FAQs](#).

Perubahan driver DLAMI NVIDIA FAQs

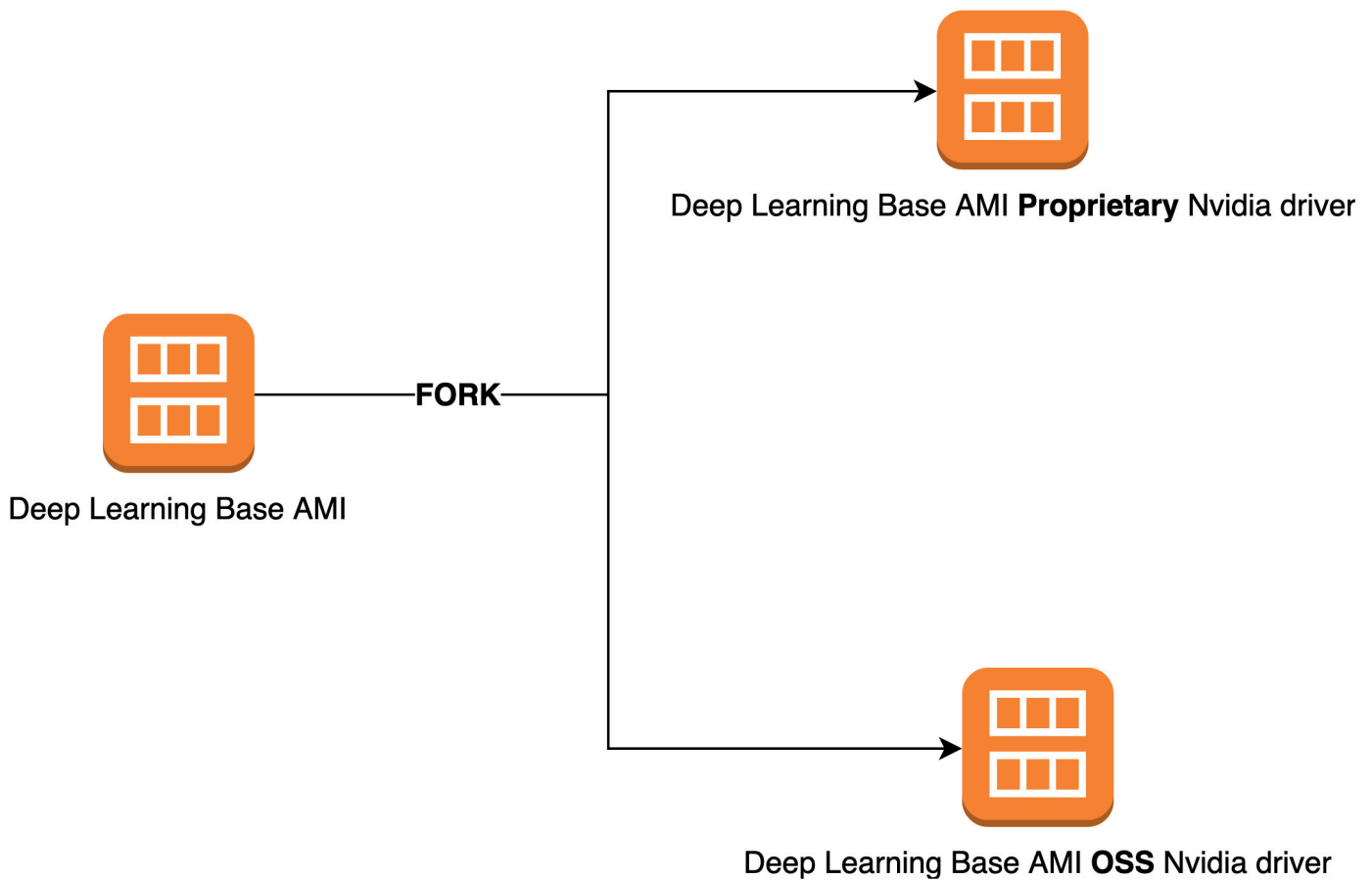
- [Apa yang berubah?](#)
- [Mengapa perubahan ini diperlukan?](#)
- [DLAMIs Apa yang mempengaruhi perubahan ini?](#)
- [Apa artinya ini bagi Anda?](#)
- [Apakah ada kehilangan fungsionalitas dengan yang lebih baru? DLAMIs](#)
- [Apakah perubahan ini memengaruhi Deep Learning Containers?](#)

Apa yang berubah?

Kami membagi DLAMIs menjadi dua kelompok terpisah:

- DLAMIs yang menggunakan driver berpemilik NVIDIA (untuk mendukung P3, P3dn, G3)
- DLAMIs yang menggunakan driver NVIDIA OSS (untuk mendukung G4dn, G5, P4, P5)

Hasilnya, kami membuat baru DLAMIs untuk masing-masing dari dua kategori dengan nama baru dan AMI baru IDs. Ini DLAMIs tidak bisa dipertukarkan. Artinya, DLAMIs dari satu grup tidak mendukung contoh yang didukung grup lain. Misalnya, DLAMI yang mendukung P5 tidak mendukung G3, dan DLAMI yang mendukung G3 tidak mendukung P5.



Mengapa perubahan ini diperlukan?

Sebelumnya, DLAMIs untuk NVIDIA GPUs termasuk driver kernel berpemilik dari NVIDIA. Namun, komunitas kernel Linux hulu menerima perubahan yang mengisolasi driver kernel berpemilik, seperti driver GPU NVIDIA, dari berkomunikasi dengan driver kernel lainnya. Perubahan ini menonaktifkan GPUDirect RDMA pada instance seri P4 dan P5, yang merupakan mekanisme yang memungkinkan penggunaan EFA secara efisien GPUs untuk pelatihan terdistribusi. Akibatnya, DLAMIs sekarang gunakan driver OpenRM (driver open source NVIDIA), yang ditautkan dengan driver EFA open source untuk mendukung G4dn, G5, P4, dan P5. Namun, driver OpenRM ini tidak mendukung instance lama (seperti P3 dan G3). Oleh karena itu, untuk memastikan bahwa kami terus menyediakan arus, berkinerja, dan aman DLAMIs yang mendukung kedua jenis instans, kami membagi DLAMIs menjadi dua kelompok: satu dengan driver OpenRM (yang mendukung G4dn, G5, P4, dan P5), dan satu dengan driver berpemilik yang lebih lama (yang mendukung P3, P3dn, dan G3).

DLAMIs Apa yang mempengaruhi perubahan ini?

Perubahan ini mempengaruhi semua DLAMIs.

Apa artinya ini bagi Anda?

Semua DLAMIs akan terus menyediakan fungsionalitas, kinerja, dan keamanan selama Anda menjalankannya pada jenis instans Amazon Elastic Compute Cloud (Amazon EC2) yang didukung. Untuk menentukan jenis EC2 instance yang didukung DLAMI, periksa catatan rilis untuk DLAMI tersebut, lalu cari Instans yang Didukung. EC2 Untuk daftar opsi DLAMI yang saat ini didukung dan tautan ke catatan rilis mereka, lihat. [Catatan rilis untuk DLAMIs](#)

Selain itu, Anda harus menggunakan perintah AWS Command Line Interface (AWS CLI) yang benar untuk memanggil arus DLAMIs.

Untuk basis DLAMIs yang mendukung P3, P3dn, dan G3, gunakan perintah ini:

```
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
```

Untuk basis DLAMIs yang mendukung G4dn, G5, P4, dan P5, gunakan perintah ini:

```
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
```

Apakah ada kehilangan fungsionalitas dengan yang lebih baru? DLAMIs

Tidak, tidak ada kehilangan fungsionalitas. Saat ini DLAMIs menyediakan semua fungsionalitas, kinerja, dan keamanan yang sebelumnya DLAMIs, asalkan Anda menjalankannya pada jenis EC2 instans yang didukung.

Apakah perubahan ini memengaruhi Deep Learning Containers?

Tidak, perubahan ini tidak memengaruhi AWS Deep Learning Containers, karena tidak menyertakan driver NVIDIA. Namun, pastikan untuk menjalankan Deep Learning Containers AMIs yang kompatibel dengan instance yang mendasarinya.

Informasi terkait DLAMI

Anda dapat menemukan sumber daya lain dengan informasi terkait tentang DLAMI di luar Panduan Pengembang AWS Deep Learning AMIs . Pada AWS re:Post, lihat pertanyaan tentang DLAMI dari pelanggan lain, atau ajukan pertanyaan Anda sendiri. Di Blog AWS Machine Learning dan AWS blog lainnya, baca posting resmi tentang DLAMI.

AWS re:Post

[Tag: AWS Deep Learning AMIs](#)

AWS Blog

- [AWS Machine Learning Blog | Kategori: AWS Deep Learning AMIs](#)
- [AWS Machine Learning Blog | Pelatihan lebih cepat dengan TensorFlow 1.6 yang dioptimalkan di instans Amazon EC2 C5 dan P3](#)
- [AWS Machine Learning Blog | Baru AWS Deep Learning AMIs untuk Praktisi Machine Learning](#)
- [AWS Partner Network \(APN\) Blog | Kursus Pelatihan Baru Tersedia: Pengantar Machine Learning & Deep Learning on AWS](#)
- [AWS Blog Berita | Perjalanan ke Pembelajaran Mendalam dengan AWS](#)

Fitur DLAMI yang tidak digunakan lagi

Tabel berikut mencantumkan fitur yang tidak digunakan lagi dari (AWS Deep Learning AMIs DLAMI), tanggal yang kita tidak digunakan lagi, dan detail tentang mengapa kita tidak menggunakannya lagi.

Fitur	Tanggal	Detail
Ubuntu 16.04	10/07/2021	Ubuntu Linux 16.04 LTS mencapai akhir jendela LTS lima tahun pada 30 April 2021 dan tidak lagi didukung oleh vendornya. Tidak ada lagi pembaruan untuk Deep Learning Base AMI (Ubuntu 16.04) dalam rilis baru per Oktober 2021. Rilis sebelumnya akan terus tersedia.
Amazon Linux	10/07/2021	Amazon Linux adalah end-of-life pada Desember 2020. Tidak ada lagi pembaruan untuk Deep Learning AMI (Amazon Linux) dalam rilis baru per Oktober 2021. Rilis Deep Learning AMI (Amazon Linux) sebelumnya akan terus tersedia.
Chainer	07/01/2020	Chainer telah mengumumkan akhir rilis utama pada Desember 2019. Akibatnya, kami tidak akan lagi memasukkan lingkungan Chainer Conda di DLAMI mulai Juli 2020. Rilis DLAMI sebelumnya yang

Fitur	Tanggal	Detail
		<p>berisi lingkungan ini akan terus tersedia. Kami akan memberikan pembaruan untuk lingkungan ini hanya jika ada perbaikan keamanan yang diterbitkan oleh komunitas open source untuk kerangka kerja ini.</p>
Python 3.6	06/15/2020	<p>Karena permintaan pelanggan, kami pindah ke Python 3.7 untuk rilis baru. TF/MX/PT</p>
Python 2	01/01/2020	<p>Komunitas open source Python telah secara resmi mengakhiri dukungan untuk Python 2.</p> <p>The TensorFlow, PyTorch, and MXNet community juga telah mengumumkan bahwa rilis TensorFlow 1.15, TensorFlow 2.1, PyTorch 1.4, dan MXNet 1.6.0 akan menjadi yang terakhir mendukung Python 2.</p>

Riwayat dokumen untuk DLAMI

Tabel berikut memberikan riwayat rilis DLAMI terbaru dan perubahan terkait dengan Panduan Pengembang AWS Deep Learning AMIs .

Perubahan terbaru

Perubahan	Deskripsi	Tanggal
Menggunakan TensorFlow Melayani untuk Melatih Model MNIST	Contoh untuk menggunakan Tensorflow yang berfungsi untuk melatih model MNIST.	Februari 14, 2025
ARM64 DLAMI	AWS Deep Learning AMIs Sekarang mendukung gambar berbasis prosesor GPUs Arm64.	29 November 2021
TensorFlow 2	AMI Pembelajaran Mendalam dengan Conda sekarang hadir dengan TensorFlow 2 dengan CUDA 10.	3 Desember 2019
AWS Inferensia	Deep Learning AMI sekarang mendukung perangkat keras AWS Inferentia dan AWS Neuron SDK.	3 Desember 2019
Menginstal PyTorch dari Nightly Build	Sebuah tutorial telah ditambahkan yang mencakup bagaimana Anda dapat menghapus instalasi PyTorch, lalu menginstal build malam PyTorch pada AMI Pembelajaran Mendalam Anda dengan Conda.	25 September 2018

[Conda Tutorial](#)

Contoh MOTD diperbarui untuk mencerminkan rilis yang lebih baru.

Juli 23, 2018

Perubahan sebelumnya

Tabel berikut memberikan riwayat rilis DLAMI sebelumnya dan perubahan terkait sebelum Juli 2018.

Perubahan	Deskripsi	Tanggal
TensorFlow dengan Horovod	Ditambahkan tutorial untuk pelatihan ImageNet dengan TensorFlow dan Horovod.	Selasa, 06 Juni 2018
Panduan peningkatan	Menambahkan panduan peningkatan.	15 Mei 2018
Daerah baru dan tutorial 10 menit baru	Wilayah baru ditambahkan: AS Barat (California N.), Amerika Selatan, Kanada (Tengah), UE (London), dan UE (Paris). Juga, rilis pertama dari tutorial 10 menit berjudul: "Memulai dengan Deep Learning AMI".	26 April 2018
Tutorial rantai	Tutorial untuk menggunakan Chainer dalam mode multi-GPU, GPU tunggal, dan CPU ditambahkan. Integrasi CUDA ditingkatkan dari CUDA 8 ke CUDA 9 untuk beberapa kerangka kerja.	28 Februari 2018
Linux AMIs v3.0, ditambah pengenalan MXNet Model	Menambahkan tutorial untuk Conda AMIs dengan model baru dan kemampuan	25 Januari 2018

Perubahan	Deskripsi	Tanggal
Server, TensorFlow Serving, dan TensorBoard	penyajian visualisasi menggunakan MXNet Model Server v0.1.5, TensorFlow Melayani v1.4.0, dan v0.4.0. TensorBoard AMI dan kerangka kerja kemampuan CUDA dijelaskan dalam ikhtisar Conda dan CUDA. Catatan rilis terbaru dipindahkan ke https://aws.amazon.com/releasenotes/	
Linux AMIs v2.0	Base, Source, dan Conda AMIs diperbarui dengan NCCL 2.1. Sumber dan Conda AMIs diperbarui dengan MXNet v1.0, PyTorch 0.3.0, dan Keras 2.0.9.	11 Desember 2017
Dua opsi AMI Windows ditambahkan	Windows 2012 R2 dan 2016 AMIs dirilis: ditambahkan ke panduan pemilihan AMI dan ditambahkan ke catatan rilis.	30 November 2017
Rilis dokumentasi awal	Penjelasan rinci tentang perubahan dengan tautan ke topik/bagian yang diubah.	15 November 2017

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.