aws

Guía para desarrolladores

AWS Deep Learning AMIs

AWS Deep Learning AMIs: Guía para desarrolladores

Table of Contents

Qué es AWS Deep Learning AMIs?

AWS Deep Learning AMIs (DLAMI) proporciona imágenes de máquina personalizadas que puede utilizar para el aprendizaje profundo en la nube. DLAMIsEstán disponibles en la mayoría Regiones de AWS para una variedad de tipos de instancias de Amazon Elastic Compute Cloud (AmazonEC2), desde una instancia CPU pequeña hasta las últimas instancias múltiples de alta potencia. GPU DLAMIsVienen preconfiguradas con NVIDIACUDAlas NVIDIAversiones DNN más recientes de los marcos de aprendizaje profundo más populares.

Acerca de esta guía

El contenido de puede ayudarle a lanzar y utilizar elDLAMIs. La guía cubre varios casos de uso comunes del aprendizaje profundo, tanto para el entrenamiento como para la inferencia. También explica cómo elegir la instancia adecuada AMI para su propósito y el tipo de instancias que podría preferir.

Además, DLAMIs incluyen varios tutoriales que ofrecen sus marcos compatibles. Esta guía puede mostrarle cómo activar cada marco y encontrar los tutoriales adecuados para empezar. También tiene tutoriales sobre el entrenamiento distribuido, la depuración y el uso AWS Inferencias y AWS Trainium y otros conceptos clave. Para obtener instrucciones sobre cómo configurar un servidor de portátiles Jupyter para ejecutar los tutoriales en su navegador, consulte. Configuración de un servidor de cuadernos de Jupyter

Requisitos previos

Para ejecutarlo correctamenteDLAMIs, le recomendamos que se familiarice con las herramientas de línea de comandos y con Python básico.

Ejemplos de casos de uso de DLAMI

A continuación se muestran ejemplos de algunos casos de uso habituales de AWS Deep Learning AMIs (DLAMI).

Aprender sobre el aprendizaje profundo: DLAMI es una excelente opción para aprender o enseñar marcos de aprendizaje automático y aprendizaje profundo. Esto DLAMIs elimina el dolor de cabeza que supone solucionar problemas en las instalaciones de cada marco y hacer que funcionen en el

mismo ordenador. DLAMIsIncluyen un cuaderno de Jupyter y facilitan la ejecución de los tutoriales que los marcos ofrecen a las personas que no conocen el aprendizaje automático y el aprendizaje profundo.

Desarrollo de aplicaciones: si eres un desarrollador de aplicaciones y estás interesado en utilizar el aprendizaje profundo para que tus aplicaciones utilicen los últimos avances de la IA, entonces DLAMI es el banco de pruebas perfecto para ti. Cada marco de trabajo incluye tutoriales sobre cómo empezar a utilizar el aprendizaje profundo, y muchos de ellos tienen colecciones de modelos que permiten probarlo sin necesidad de crear redes neuronales ni de llevar a cabo el entrenamiento de modelos. Algunos ejemplos le muestran cómo crear una aplicación de detección de imágenes en tan solo unos minutos, o cómo crear una aplicación de reconocimiento de voz para su propio chatbot.

Aprendizaje automático y análisis de datos: si es un científico de datos o está interesado en procesar sus datos con aprendizaje profundo, descubrirá que muchos de los marcos son compatibles con R y Spark. Encontrará tutoriales sobre cómo crear desde regresiones sencillas hasta sistemas escalables de procesamiento de datos para sistemas de predicción y personalización.

Investigación: si eres un investigador que quiere probar un nuevo marco, probar un nuevo modelo o entrenar nuevos modelos, entonces DLAMI y AWS las capacidades de escalabilidad pueden aliviar las tediosas instalaciones y la administración de varios nodos de entrenamiento.

> **Note**
>
> Si bien su primera opción podría ser actualizar su tipo de instancia a una instancia más grande con más GPUs (hasta 8), también puede escalar horizontalmente mediante la creación de un clúster de DLAMI instancias. Consulte Información relacionada sobre DLAMI para obtener más información sobre las compilaciones de clústeres.

Características de DLAMI

Las características de AWS Deep Learning AMIs (DLAMI) incluyen marcos de aprendizaje profundo, GPU software, servidores de modelos y herramientas de visualización de modelos preinstalados.

Marcos preinstalados

Actualmente, existen dos variantes principales, DLAMI con otras variaciones relacionadas con el sistema operativo (SO) y las versiones de software:

- Aprendizaje profundo AMI con Conda— Los marcos se instalan por separado mediante `conda` paquetes y entornos Python separados.
- AMI base de aprendizaje profundo— No hay marcos instalados; solo NVIDIACUDAy otras dependencias.

El aprendizaje profundo AMI con Conda utiliza `conda` entornos para aislar cada marco, de modo que pueda cambiar de uno a otro a su antojo sin preocuparse de que sus dependencias entren en conflicto. El aprendizaje profundo AMI con Conda es compatible con los siguientes marcos:

- PyTorch
- TensorFlow 2.

> **Note**
>
> DLAMIya no es compatible con los siguientes marcos de aprendizaje profundo: ApacheMXNet, Microsoft Cognitive Toolkit (CNTK), Caffe, Caffe2, Theano, Chainer y Keras.

Software preinstalado GPU

Incluso si usa una instancia CPU única, tendrán NVIDIACUDAy DLAMIs NVIDIA cortarán. DNN El software instalado es el mismo, independientemente del tipo de instancia. Ten en cuenta que las herramientas GPU específicas solo funcionan en una instancia que tenga al menos una. GPU Para obtener más información sobre los tipos de instancias, consulteSelección del tipo de instancia para DLAMI.

Para obtener más información sobreCUDA, consulteInstalaciones de CUDA y enlaces de marco de trabajo.

Presentación y visualización de modelos

El aprendizaje profundo AMI con Conda viene preinstalado con servidores de modelos para TensorFlow, así como TensorBoard para la visualización de modelos. Para obtener más información, consulte TensorFlow Sirviendo.

Introducción

Cómo empezar con el DLAMI

Esta guía incluye consejos sobre cómo elegir la DLAMI instancia adecuada para usted, seleccionar un tipo de instancia que se adapte a su caso de uso y presupuesto, y Información relacionada sobre DLAMI describe las configuraciones personalizadas que pueden resultarle interesantes.

Si es la primera vez que usas AWS o usando AmazonEC2, comience conAprendizaje profundo AMI con Conda. Si estás familiarizado con Amazon EC2 y otros AWS servicios como Amazon EMREFS, Amazon o Amazon S3, y si está interesado en integrar esos servicios para proyectos que necesitan formación o inferencia distribuida, compruebe si alguno se adapta Información relacionada sobre DLAMI a su caso de uso.

Le recomendamos que consulte ¿Cómo elegir el tuyo DLAMI para que se haga una idea del tipo de instancia que mejor se adapta a su aplicación.

Paso siguiente

¿Cómo elegir el tuyo DLAMI

¿Cómo elegir el tuyo DLAMI

Ofrecemos una gama de DLAMI opciones. Para ayudarlo a seleccionar la que mejor se DLAMI adapte a su caso de uso, agrupamos las imágenes por el tipo de hardware o la funcionalidad para la que se desarrollaron. Nuestras agrupaciones principales son:

- DLAMITipo: CUDA versus base versus marco único versus marco múltiple (Conda) DLAMI
- Arquitectura de cómputo: basada en x86 frente a basada en ARM64 AWS Gravitón
- Tipo de procesador: GPUhttps://docs.aws.amazon.com/dlami/latest/devguide/gpu versus versus CPUInferentia versus Trainium
- SDKCUDA: contra AWS Neurona
- SO: Amazon Linux versus Ubuntu

En el resto de los temas de esta guía encontrará más detalles.

Temas

Tema siguiente

Aprendizaje profundo AMI con Conda

Instalaciones de CUDA y enlaces de marco de trabajo

Mientras que el aprendizaje profundo es algo bastante novedoso, todos los marcos de trabajo ofrecen versiones "estables". Es posible que estas versiones estables no funcionen con las implementaciones y características más recientes de CUDA o cuDNN. Su caso de uso y las características que necesita pueden ayudarle a elegir un marco. Si no está seguro, utilice la última AMI de aprendizaje profundo con Conda. Tiene `pip` binarios oficiales para todos los marcos con CUDA, utilizando la versión más reciente compatible con cada marco. Si desea obtener las versiones más recientes y personalizar su entorno de aprendizaje profundo, utilice la AMI base de aprendizaje profundo.

Eche un vistazo a nuestra guía sobre Comparación de Stable y Release Candidates para obtener más información.

Elija una DLAMI con CUDA

AMI base de aprendizaje profundoTiene todas las series de versiones CUDA disponibles

Aprendizaje profundo AMI con CondaTiene todas las series de versiones CUDA disponibles

> **Note**
>
> Ya no incluimos los entornos MXNet, CNTK, Caffe, Caffe2, Theano, Chainer o Keras Conda en el. AWS Deep Learning AMIs

Para obtener números de versión específicos, consulte las Notas de la versión de DLAMIs.

Elija este tipo de DLAMI u obtenga más información sobre las distintas DLAMI mediante la opción Tema siguiente.

Elija una de las versiones de CUDA y consulte en el Apéndice la lista completa de las DLAMI que tengan esa versión, u obtenga más información sobre las distintas DLAMI en el tema indicado en la sección Tema siguiente.

Tema siguiente

AMI base de aprendizaje profundo

Temas relacionados

- Para obtener instrucciones sobre cómo cambiar de versión de CUDA, consulte el tutorial Uso de la base de aprendizaje profundo AMI.

AMI base de aprendizaje profundo

La AMI base de aprendizaje profundo es como un lienzo vacío para el aprendizaje profundo. Incluye todo lo que se necesita hasta el momento de la instalación de un marco de trabajo determinado, e incluye las versiones de CUDA que haya elegido.

Por qué elegir la DLAMI base

Este grupo de AMI es útil para los colaboradores de proyectos que desean adaptar un proyecto de aprendizaje profundo y compilar la versión más reciente. Está pensado para quienes desean actualizar su propio entorno con la confianza de que tienen instalado y en funcionamiento el software más reciente de NVIDIA, y desean centrarse en seleccionar los marcos de trabajo y las versiones que quieren instalar.

Elija este tipo de DLAMI u obtenga más información sobre las distintas DLAMI mediante la opción Tema siguiente.

Tema siguiente

DLAMI con Conda

Temas relacionados

- Uso de la AMI base de aprendizaje profundo

Aprendizaje profundo AMI con Conda

El Conda DLAMI utiliza entornos `conda` virtuales, están presentes en varios marcos o en un solo marco. DLAMIs Estos entornos están configurados para mantener separadas las instalaciones de los distintos marcos de trabajo y agilizar el paso de un marco a otro. Esto es ideal para aprender y experimentar con todos los marcos que DLAMI ofrece. La mayoría de los usuarios consideran que el nuevo aprendizaje profundo AMI con Conda es perfecto para ellos.

Se actualizan a menudo con las últimas versiones de los marcos y tienen los GPU controladores y el software más recientes. Por lo general, se denominan AWS Aprendizaje profundo AMIs en la mayoría de los documentos. Son DLAMIs compatibles con los sistemas operativos Ubuntu 20.04 y Amazon Linux 2. El soporte de los sistemas operativos depende del soporte del sistema operativo anterior.

Comparación de Stable y Release Candidates

Los Conda AMIs utilizan binarios optimizados de las versiones formales más recientes de cada marco. No se esperan "release candidates" ni funciones experimentales. Las optimizaciones dependen de la compatibilidad del marco con tecnologías de aceleración, como las de Intel MKLDNN, que aceleran el entrenamiento y la inferencia en los tipos de instancias C5 y C4. CPU Los binarios también se compilan para admitir conjuntos de instrucciones Intel avanzados, que incluyen, entre otrosAVX, AVX -2, SSE4 .1 y .2. SSE4 Estos aceleran las operaciones vectoriales y de punto flotante en las CPU arquitecturas Intel. Además, por GPU ejemplo, los tipos, the CUDA y cu DNN se actualizan con cualquier versión compatible con la última versión oficial.

El aprendizaje profundo AMI con Conda instala automáticamente la versión más optimizada del marco para su EC2 instancia de Amazon tras la primera activación del marco. Para obtener más información, consulte Uso del aprendizaje profundo con Conda AMI.

Si desea instalar desde el código fuente mediante opciones de compilación personalizadas u optimizadas, las AMI base de aprendizaje profundo podrían ser una opción más adecuada para usted.

Retirada de Python 2

La comunidad de código abierto de Python finalizó oficialmente la compatibilidad con Python 2 el 1 de enero de 2020. La TensorFlow PyTorch comunidad ha anunciado que las versiones TensorFlow 2.1 y PyTorch 1.4 son las últimas compatibles con Python 2. Las versiones anteriores DLAMI (v26,

v25, etc.) que contienen entornos Conda de Python 2 siguen estando disponibles. Sin embargo, proporcionamos actualizaciones para los entornos Conda de Python 2 en DLAMI las versiones publicadas anteriormente solo si hay correcciones de seguridad publicadas por la comunidad de código abierto para esas versiones. DLAMIlas versiones con las últimas versiones de los PyTorch marcos TensorFlow y no contienen los entornos Conda de Python 2.

CUDASupport

Los números de CUDA versión específicos se encuentran en las notas GPU DLAMI de la versión.

Tema siguiente

Opciones de arquitectura DLAMI

Temas relacionados

- Para ver un tutorial sobre el uso del aprendizaje profundo AMI con Conda, consulte el Uso del aprendizaje profundo con Conda AMI tutorial.

Opciones de arquitectura DLAMI

AWS Deep Learning AMIsLos s se ofrecen con arquitecturas Graviton2 basadas en x86 o ARM64.AWS

Para obtener información sobre cómo empezar a utilizar la DLAMI de GPU ARM64, consulte. La ARM64 DLAMI Para obtener más información sobre los tipos de instancias disponibles, consulte Selección del tipo de instancia para DLAMI.

Tema siguiente

Opciones de sistema operativo para la DLAMI

Opciones de sistema operativo para la DLAMI

Las DLAMI se ofrecen en los siguientes sistemas operativos.

- Amazon Linux 2
- Ubuntu 20.04
- Ubuntu 22.04

Las versiones anteriores de los sistemas operativos están disponibles en las DLAMI obsoletas. Para obtener más información sobre la obsolescencia de DLAMI, consulte Deprecations for DLAMI

Antes de elegir una DLAMI, evalúe qué tipo de instancia necesita e identifique su región de AWS .

Tema siguiente

Selección del tipo de instancia para DLAMI

Selección del tipo de instancia para DLAMI

De manera más general, tenga en cuenta lo siguiente al seleccionar un tipo de instancia para unDLAMI.

- Si eres nuevo en el mundo del aprendizaje profundo, una instancia con una sola GPU podría adaptarse a tus necesidades.
- Si te preocupas por tu presupuesto, puedes usar instancias CPU exclusivas.
- Si busca optimizar el alto rendimiento y la rentabilidad para la inferencia de modelos de aprendizaje profundo, puede utilizar instancias con AWS Chips de inferencia.
- Si busca una GPU instancia de alto rendimiento con una CPU arquitectura basada en ARM64, puede utilizar el tipo de instancia G5g.
- Si está interesado en ejecutar un modelo previamente entrenado para inferencias y predicciones, puede adjuntar una Amazon Elastic Inference a su instancia de Amazon. EC2 Amazon Elastic Inference le da acceso a un acelerador con una fracción de a. GPU
- Para los servicios de inferencia de gran volumen, una única CPU instancia con mucha memoria o un clúster de dichas instancias podría ser una mejor solución.
- Si está utilizando un modelo de gran tamaño con muchos datos o un tamaño de lote elevado, necesitará una instancia más grande con más memoria. También puede distribuir el modelo en un clúster de. GPUs El uso de una instancia con menos memoria puede ser una solución más adecuada para usted si disminuye el tamaño del lote. Sin embargo, puede afectar a la precisión y a la velocidad de entrenamiento.
- Si está interesado en ejecutar aplicaciones de aprendizaje automático mediante la biblioteca de comunicaciones NVIDIA colectivas (NCCL) que requieren altos niveles de comunicación entre nodos a gran escala, puede utilizar Elastic Fabric Adapter (EFA).

Para obtener más información sobre las instancias, consulte Tipos de instancias.

Los siguientes temas proporcionan información acerca de las consideraciones del tipo de instancia.

> **Important**
>
> El aprendizaje profundo AMIs incluye controladores, software o kits de herramientas desarrollados, propiedad o proporcionados por la NVIDIA empresa. Aceptas usar estos NVIDIA controladores, software o kits de herramientas únicamente en las EC2 instancias de Amazon que incluyan NVIDIA hardware.

Temas

Precios de la DLAMI

Los marcos de aprendizaje profundo incluidos DLAMI son gratuitos y cada uno tiene sus propias licencias de código abierto. Aunque el software incluido en el DLAMI es gratuito, tienes que pagar por el hardware de la EC2 instancia de Amazon subyacente.

Algunos tipos de EC2 instancias de Amazon están etiquetados como gratuitos. Es posible ejecutarlo DLAMI en una de estas instancias gratuitas. Esto significa que el uso de DLAMI es totalmente gratuito si solo se utiliza la capacidad de esa instancia. Si necesitas una instancia más potente con más CPU núcleos, más espacio en discoRAM, más o una o másGPUs, entonces necesitas una instancia que no pertenezca a la clase de instancias de nivel libre.

Para obtener más información sobre la selección de instancias y los precios, consulta EC2los precios de Amazon.

Disponibilidad por región de DLAMI

Cada región admite tipos de instancias distintos y, a menudo, un tipo de instancia tiene un costo ligeramente distinto en diferentes regiones. DLAMIsno están disponibles en todas las regiones,

pero es posible copiarlos DLAMIs a la región que prefieras. Consulte Copiar un AMI ventilador para obtener más información. Fíjese en la lista de selección de regiones y asegúrese de que elige una región que esté cerca de usted o de sus clientes. Si planea usar más de uno DLAMI y, posiblemente, crear un clúster, asegúrese de usar la misma región para todos los nodos del clúster.

Para obtener más información sobre las regiones, visita Regions.

Tema siguiente

GPUInstancias recomendadas

GPUInstancias recomendadas

Recomendamos una GPU instancia para la mayoría de los fines del aprendizaje profundo. Entrenar nuevos modelos es más rápido en una GPU instancia que en una CPU instancia. Puede escalar de forma sublineal si tiene varias GPU instancias o si utiliza la formación distribuida en muchas instancias con ellas. GPUs

Los siguientes tipos de instancias son compatibles con. DLAMI Para obtener información sobre GPU las opciones de tipos de instancias y seleccione Computación acelerada.

> **Note**
>
> El tamaño del modelo debe ser un factor a tener en cuenta para la selección de una instancia. Si tu modelo supera el de una instancia disponibleRAM, selecciona un tipo de instancia diferente con memoria suficiente para tu aplicación.

- Las instancias Amazon EC2 P5e tienen hasta 8 NVIDIA Tesla H200. GPUs
- Las instancias Amazon EC2 P5 tienen hasta 8 NVIDIA Tesla GPUs H100.
- Las instancias Amazon EC2 P4 tienen hasta 8 NVIDIA Tesla GPUs A100.
- Las instancias Amazon EC2 P3 tienen hasta 8 NVIDIA Tesla GPUs V100.
- Las instancias Amazon EC2 G3 tienen hasta 4 NVIDIA Tesla GPUs M60.
- Las instancias Amazon EC2 G4 tienen hasta 4 NVIDIA GPUs T4.
- Las instancias Amazon EC2 G5 tienen hasta 8 NVIDIA GPUs A10G.
- Las instancias Amazon EC2 G6 tienen hasta 8 NVIDIA GPUs L4.
- Las instancias Amazon EC2 G6e tienen hasta 8 núcleos Tensor NVIDIA L40S. GPUs
- Las instancias EC2 G5g de Amazon están basadas en ARM64 AWS Procesadores Graviton 2.

DLAMIlas instancias proporcionan herramientas para monitorear y optimizar sus procesos. GPU Para obtener más información sobre la supervisión de sus GPU procesos, consulteGPUSupervisión y optimización.

Para ver tutoriales específicos sobre cómo trabajar con instancias G5G, consulte La ARM64 DLAMI.

Tema siguiente

CPUInstancias recomendadas

CPUInstancias recomendadas

Ya sea que tengas un presupuesto limitado, estés aprendiendo sobre el aprendizaje profundo o simplemente quieras utilizar un servicio de predicción, tienes muchas opciones asequibles en CPU esta categoría. Algunos marcos aprovechan las ventajas de Intel MKLDNN, lo que agiliza la formación y las inferencias sobre los tipos de CPU instancias C5 (no disponibles en todas las regiones). Para obtener información sobre los tipos de CPU instancias, consulte y seleccione EC2Optimizado para cómputo.

> **Note**
>
> El tamaño del modelo debe ser un factor a tener en cuenta para la selección de una instancia. Si tu modelo supera el de una instancia disponibleRAM, selecciona un tipo de instancia diferente con suficiente memoria para tu aplicación.

- Las instancias Amazon EC2 C5 tienen hasta 72 procesadores IntelvCPUs. Las instancias C5 se destacan en el modelado científico, el procesamiento por lotes, el análisis distribuido, la computación de alto rendimiento (HPC) y la inferencia de aprendizaje profundo y automático.

Tema siguiente

Instancias de Inferentia recomendadas

Instancias de Inferentia recomendadas

AWS Las instancias de Inferentia están diseñadas para proporcionar un alto rendimiento y rentabilidad para las cargas de trabajo de inferencia de modelos de aprendizaje profundo. En concreto, los tipos de instancias de Inf2 utilizan AWS Los chips Inferentia y el AWS Neuron SDK, que está integrado con marcos populares de aprendizaje automático como y. TensorFlow PyTorch

Los clientes pueden usar las instancias de Inf2 para ejecutar aplicaciones de inferencia de machine learning a gran escala, como búsquedas, motores de recomendación, visión artificial, reconocimiento de voz, procesamiento del lenguaje natural, personalización y detección de fraudes, al menor costo en la nube.

> **Note**
>
> El tamaño del modelo debe ser un factor a tener en cuenta para la selección de una instancia. Si tu modelo supera el de una instancia disponibleRAM, selecciona un tipo de instancia diferente con suficiente memoria para tu aplicación.

- Las instancias de Amazon EC2 Inf2 tienen hasta 16 AWS Chips inferentia y un rendimiento de red de 100 Gbps.

Para obtener más información sobre cómo empezar con AWS InferenciaDLAMIs, consulte. La AWS Chip Inferentia con DLAMI

Tema siguiente

Instancias de Trainium recomendadas

Instancias de Trainium recomendadas

AWS Las instancias de Trainium están diseñadas para proporcionar un alto rendimiento y rentabilidad para las cargas de trabajo de inferencia de modelos de aprendizaje profundo. En concreto, los tipos de instancias Trn1 utilizan AWS Los chips Trainium y el AWS Neuron SDK, que está integrado con marcos populares de aprendizaje automático como y. TensorFlow PyTorch

Los clientes pueden usar las instancias de Trn1 para ejecutar aplicaciones de inferencia de machine learning a gran escala, como búsquedas, motores de recomendación, visión artificial, reconocimiento de voz, procesamiento del lenguaje natural, personalización y detección de fraudes, al menor costo en la nube.

> **Note**
>
> El tamaño del modelo debe ser un factor a tener en cuenta para la selección de una instancia. Si tu modelo supera el de una instancia disponibleRAM, selecciona un tipo de instancia diferente con suficiente memoria para tu aplicación.

- Las instancias Amazon EC2 Trn1 tienen hasta 16 AWS Chips Trainium y 100 Gbps de rendimiento de red.

Lanzamiento y configuración de una DLAMI

Si estás aquí, ya deberías tener una buena idea de lo que AMI quieres lanzar. Si no es así, busca el hardware, los marcos y la recuperación de ID relacionados en. DLAMI Notas de la versión de DLAMIs

También debe saber el tipo de instancia y la región que va a elegir. En caso contrario, consulte Selección del tipo de instancia para DLAMI.

> **Note**
>
> Usaremos p3.16xlarge como el tipo de instancia predeterminado en los ejemplos. Sustitúyalo por el tipo de instancia que tenga pensado.

> **Important**
>
> Si planea usar Elastic Inference, tiene una configuración de Elastic Inference que debe completarse antes de lanzar su. DLAMI

Temas

Paso 1: Lanzamiento de una DLAMI

> **Note**
>
> Para este tutorial, podríamos hacer referencias específicas al aprendizaje profundo AMI (Ubuntu 18.04). Incluso si seleccionas una diferenteDLAMI, deberías poder seguir esta guía.

1. Encuentra el ID de tu DLAMI
2. Lanza una EC2 instancia de Amazon desde tu DLAMI

   Utilizarás Amazon EC2 Console. Siga las instrucciones detalladas en Lanzamiento desde Amazon EC2 Console

   > **Tip**
   >
   > CLIOpción: si eliges hacer girar una DLAMI con la AWS CLI, necesitarás el identificador, AMI la región y el tipo de instancia, y la información de tu token de seguridad. Asegúrate de tener tu instancia AMI y IDs lista. Si no ha configurado la AWS CLIsin embargo, hágalo primero utilizando la guía para instalar el AWS Interfaz de línea de comandos.

3. Una vez que haya finalizado los pasos de una de estas opciones, espere a que la instancia esté lista. Este proceso suele tardar unos minutos. Puede verificar el estado de la instancia en la EC2consola.

Recupera el DLAMI ID

Cada uno AMI posee un identificador (ID) único. Puede consultar el ID DLAMI de su elección con el AWS Interfaz de línea de comandos (AWS CLI). Si aún no tiene el AWS CLI instalado, consulte Primeros pasos con el AWS CLI.

> **Note**
>
> Consulte las notas de la DLAMI versión Notas de la versión de DLAMIs para ver las configuraciones de software adicionales (controladores, versiones de Python, EBS tipo Amazon).

1. Asegúrese de que su AWS las credenciales están configuradas.

   ```
   aws configure
   ```

2. Utilice el siguiente comando para recuperar su ID DLAMI o buscar la consulta proporcionada en el AWS Deep Learning AMIs notas de publicación.

   ```
   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
```

> **Note**
>
> Puede especificar una versión de lanzamiento para un marco determinado u obtener la última versión sustituyendo el número de versión por un signo de interrogación.

3. El resultado debería tener un aspecto similar al siguiente:

```
ami-09ee1a996ac214ce7
```

Copie este DLAMI identificador y pulse q para salir del mensaje.

Paso siguiente

Lanzamiento desde Amazon EC2 Console

Lanzamiento desde Amazon EC2 Console

> **Note**
>
> Para lanzar una instancia con Elastic Fabric Adapter (EFA), sigue estos pasos.

1. Abre la EC2consola.
2. Tome nota de la región actual en la parte superior del panel de navegación. Si esto no es lo que deseas Región de AWS, cambie esta opción antes de continuar. Para obtener más información, consulte y EC2regiones.
3. Elija Iniciar instancia.
4. Introduzca un nombre para la instancia y seleccione el DLAMI que mejor se adapte a sus necesidades.
   a. Busca uno existente DLAMI en Mi AMIs o selecciona Inicio rápido.
   b. Busca por DLAMI ID. Examine las opciones y seleccione la que desee.

5. Elija un tipo de instancia. Puedes encontrar las familias de instancias recomendadas para ti DLAMI en el AWS Deep Learning AMIs notas de la versión. Para obtener recomendaciones generales sobre los tipos de DLAMI instancias, consulta Instance Selection.

   > **Note**
   >
   > Si desea utilizar Elastic Inference (EI), haga clic en Configurar los detalles de la instancia, seleccione Add an Amazon EI accelerator y, a continuación, seleccione el tamaño del acelerador de Amazon EI.

6. Elija Iniciar instancia.

> **Tip**
>
> Consulta Cómo empezar con el aprendizaje profundo con el AWS Aprendizaje profundo AMI para ver un recorrido con capturas de pantalla.

Paso siguiente

Paso 2: Conexión a la DLAMI

Paso 2: Conexión a la DLAMI

Conéctese al DLAMI que ha iniciado desde un cliente (Windows, macOS o Linux). Para obtener más información, consulte Connect to Your Linux Instance en la Guía del EC2 usuario de Amazon.

Tenga a mano una copia del comando SSH login si quiere realizar la configuración de Jupyter después de iniciar sesión. Utilizará una variante del comando para conectarse a la página web de Jupyter.

Paso siguiente

Paso 3: Comprobación de la DLAMI

Paso 3: Comprobación de la DLAMI

Dependiendo de tu DLAMI versión, tienes diferentes opciones de prueba:

- Aprendizaje profundo AMI con Conda : vaya a Uso del aprendizaje profundo con Conda AMI.
- AMI base de aprendizaje profundo : consulte la documentación de instalación del marco que desee.

También puede crear un bloc de notas de Jupyter, probar los tutoriales o comenzar a escribir código en Python. Para obtener más información, consulte Configuración de un servidor de cuadernos de Jupyter.

Paso 4: Administre su DLAMI instancia

Mantenga siempre actualizado su sistema operativo y otro software instalado y aplique los parches y actualizaciones en cuanto estén disponibles.

Si utilizas Amazon Linux o Ubuntu, cuando inicies sesión en tuDLAMI, se te notificará si hay actualizaciones disponibles y consultarás las instrucciones de actualización. Para obtener más información sobre el mantenimiento de Amazon Linux, consulte Actualizar software en la instancia de Amazon Linux. Para las instancias de Ubuntu, consulte la documentación oficial de Ubuntu.

En Windows, compruebe Windows Update con regularidad para ver si hay actualizaciones de seguridad y de software. Si lo prefiere, aplique las actualizaciones automáticamente.

> **Important**
>
> Para obtener información sobre las vulnerabilidades de Meltdown y Spectre y sobre cómo parchear su sistema operativo para solucionarlas, consulte el boletín de seguridad AWS-2018-013.

Eliminación

Cuando ya no lo necesiteDLAMI, puede detenerlo o cancelarlo para evitar incurrir en cargos continuos. Si se detiene una instancia, se conservará para que pueda reanudarla más adelante. Las configuraciones, los archivos y demás información no volátil se almacenan en un volumen en Amazon S3. Se le cobrará una pequeña cuota de S3 por conservar el volumen mientras la instancia esté detenida, pero no se le cobrará por los recursos informáticos mientras se encuentre en ese estado. Cuando inicie la instancia de nuevo, se montará ese volumen y sus datos estarán disponibles. Si termina una instancia, se borrará y no podrá volver a iniciarla. En realidad, los datos

todavía se encuentran en S3, por lo que, para evitar nuevos cargos, debe eliminar también el volumen. Para obtener más instrucciones, consulte Finalizar su instancia en la Guía del EC2 usuario de Amazon.

Configuración de un servidor de cuadernos de Jupyter

Un servidor de cuadernos de Jupyter permite crear y ejecutar cuadernos de Jupyter desde la instancia de DLAMI. Con los cuadernos de Jupyter, puede realizar experimentos de machine learning (ML) para entrenamiento e inferencia mientras utiliza la infraestructura de AWS y accede a los paquetes integrados en la DLAMI. Para obtener más información sobre los cuadernos de Jupyter, consulte la documentación del cuaderno de Jupyter.

Para configurar un cuaderno de Jupyter:

- Configure el servidor de cuadernos de Jupyter en la instancia de la DLAMI de Amazon EC2.
- Configure el cliente para poder conectarse al servidor de cuadernos de Jupyter. Dispone de instrucciones de configuración para clientes Windows, macOS y Linux.
- Pruebe la configuración iniciando sesión en el servidor de cuadernos de Jupyter.

Para completar los pasos necesarios para configurar un Jupyter, siga las instrucciones de los siguientes temas. Una vez que haya configurado un servidor de cuadernos de Jupyter, consulte Ejecución de los tutoriales del cuaderno de Jupyter para obtener información sobre cómo ejecutar los cuadernos de ejemplo que se incluyen en la DLAMI.

Temas

Protección del servidor de Jupyter

Aquí configuraremos Jupyter con SSL y una contraseña personalizada.

Conéctese a la instancia de Amazon EC2 y, a continuación, complete el siguiente procedimiento.

Configuración del servidor de Jupyter

1. Jupyter proporciona una utilidad de contraseñas. Ejecute el siguiente comando y escriba la contraseña que desee en el símbolo del sistema.

   ```
   $ jupyter notebook password
   ```

   El resultado tendrá un aspecto similar a este:

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

2. Cree un certificado SSL autofirmado. Siga las instrucciones para especificar la configuración regional más adecuada para sus necesidades. Debe introducir `.` si desea dejar un mensaje en blanco. Sus respuestas no afectarán a la funcionalidad del certificado.

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

> **Note**
>
> Tal vez le convenga crear un certificado normal que esté firmado por un tercero y que no haga que el navegador muestre una advertencia de seguridad. Este proceso es mucho más complejo. Consulte la documentación de Jupyter para obtener más información.

Paso siguiente

Inicio del servidor de cuadernos de Jupyter

Inicio del servidor de cuadernos de Jupyter

Ahora puede iniciar el servidor de Jupyter iniciando sesión en la instancia y ejecutando el siguiente comando que utiliza el certificado SSL que ha creado en el paso anterior.

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

Con el servidor iniciado, puede conectarse a él desde el equipo cliente través de un túnel SSH. Cuando se ejecute el servidor, verá un resultado de Jupyter que confirma que el servidor está en ejecución. En este punto, no tenga en cuenta el aviso de que puede obtener acceso al servidor a través de una dirección URL de localhost, ya que eso no funcionará hasta que cree el túnel.

> **Note**
>
> Jupyter se encargará de cambiar de entorno por usted cuando cambie de plataforma con la interfaz web de Jupyter. Puede encontrar más información al respecto en Cambio de entorno con Jupyter.

Paso siguiente

Configuración del cliente para conectarse con el servidor de Jupyter

Configuración del cliente para conectarse con el servidor de Jupyter

Después de configurar el cliente para que se conecte con el servidor de cuadernos de Jupyter, puede crear cuadernos en el servidor y obtener acceso a ellos desde su espacio de trabajo, así como ejecutar código de aprendizaje profundo en el servidor.

Para obtener información sobre la configuración, elija uno de los siguientes enlaces.

Temas

- Configuración de un cliente Windows
- Configurar un cliente Linux o macOS

Configuración de un cliente Windows

Prepare

Asegúrese de tener la siguiente información, ya que la necesitará para configurar el túnel de SSH:

- El nombre de DNS público de la instancia de Amazon EC2. Puede encontrar el nombre DNS público en la consola de EC2.

- El par de claves del archivo de clave privada. Para obtener más información sobre el acceso al par de claves, consulte Pares de claves de Amazon EC2 en la Guía del usuario de Amazon EC2 para instancias de Linux.

Uso de cuadernos de Jupyter desde un cliente Windows

Consulte estas guías sobre cómo conectarse a la instancia de Amazon EC2 desde un cliente de Windows.

1. Solución de problemas con la conexión a la instancia
2. Conexión a la instancia Linux desde Windows utilizando PuTTY

Para crear un túnel con un servidor de Jupyter en ejecución, el enfoque recomendado es instalar Git Bash en su cliente Windows y seguir después las instrucciones del cliente Linux/macOS. Sin embargo, puede utilizar el enfoque que desee para abrir un túnel SSH con mapeo de puertos. Consulte la documentación de Jupyter para obtener más información.

Paso siguiente

Configurar un cliente Linux o macOS

Configurar un cliente Linux o macOS

1. Abra un terminal .
2. Ejecute el siguiente comando para reenviar todas las solicitudes del puerto local 8888 al puerto 8888 de la instancia remota de Amazzon EC2. Actualice el comando sustituyendo la ubicación de la clave para obtener acceso a la instancia de Amazon EC2 y al nombre DNS público de esa instancia. Tenga en cuenta que para una AMI de Amazon Linux el nombre de usuario es `ec2-user` en lugar de `ubuntu`.

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

   Este comando abre un túnel entre el cliente y la instancia remota de EC2 que está ejecutando el servidor de cuadernos de Jupyter.

Paso siguiente

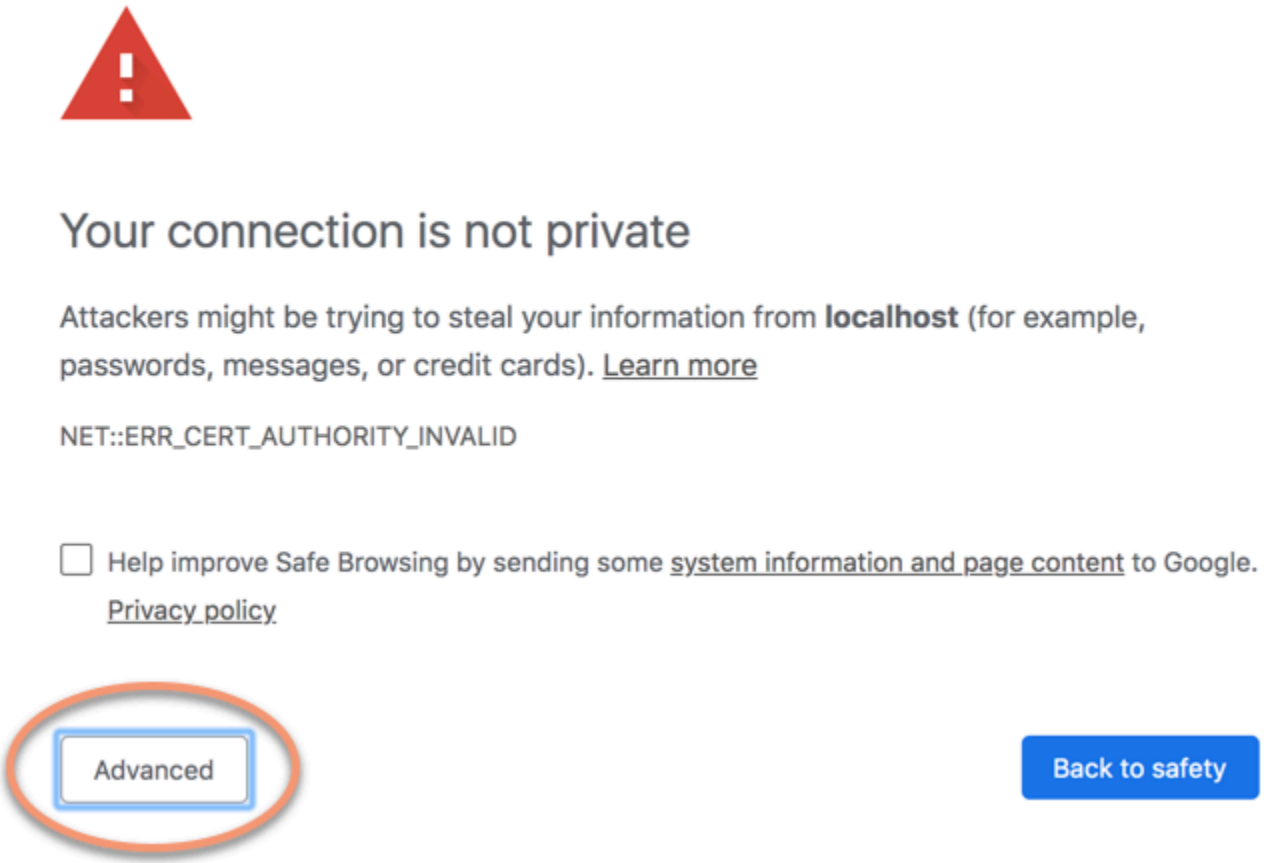
Prueba de la conexión iniciando sesión en el servidor de cuadernos de Jupyter

Prueba de la conexión iniciando sesión en el servidor de cuadernos de Jupyter

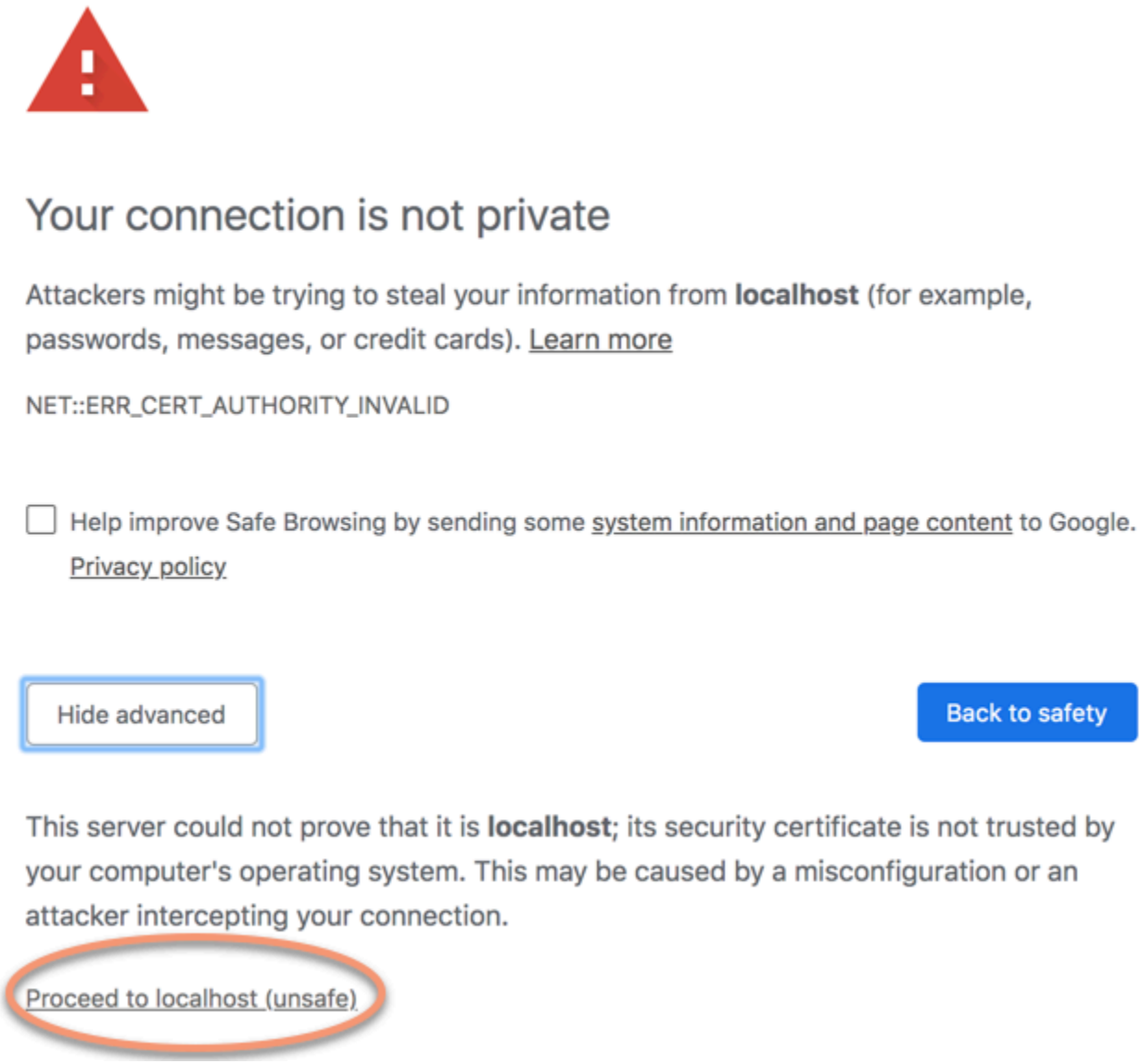
Ahora ya puede iniciar sesión en el servidor de cuadernos de Jupyter.

El siguiente paso consiste en probar la conexión con el servidor a través del navegador.

1. Escriba la siguiente dirección URL en la barra de direcciones del navegador o haga clic en este enlace: https://localhost:8888
2. Con un certificado SSL autofirmado, el navegador emitirá un aviso y le recomendará que abandone el sitio web.



Como lo ha configurado usted, puede continuar sin problema. Dependiendo del navegador, aparece un botón "avanzadas", "mostrar detalles" o similar.



Haga clic en este botón y, a continuación, haga clic en el enlace "ir a localhost". Si la conexión se realiza correctamente, aparecerá la página web del servidor de cuadernos de Jupyter. En este momento, se le pedirá la contraseña que configuró anteriormente.

Ahora ya dispone de acceso al servidor de cuadernos de Jupyter que se ejecuta en la DLAMI. Puede crear cuadernos nuevos o ejecutar los Tutoriales proporcionados.

Usando un DLAMI

Temas

En las siguientes secciones se describe cómo se puede utilizar el aprendizaje profundo AMI con Conda para cambiar de entorno, ejecutar código de ejemplo desde cada uno de los marcos y ejecutar Jupyter para que pueda probar diferentes tutoriales de notebook.

Uso del aprendizaje profundo con Conda AMI

Temas

Introducción al aprendizaje profundo AMI con Conda

Conda es un sistema de código abierto para la administración de paquetes y del entorno que se ejecuta en Windows, macOS y Linux. Conda instala, ejecuta y actualiza rápidamente paquetes y sus dependencias. Conda le permite crear, guardar y cargar entornos en el equipo local, así como alternar entre ellos, con suma facilidad.

El aprendizaje profundo AMI con Conda se ha configurado para que pueda cambiar fácilmente de un entorno de aprendizaje profundo a otro. Las siguientes instrucciones le orientan acerca de algunos comandos básicos con `conda`. También le ayudan a verificar que la importación básica del marco de trabajo funciona correctamente, y que puede ejecutar un par de operaciones sencillas con este. A continuación, puede continuar con los tutoriales más exhaustivos que se proporcionan con los ejemplos de marcos DLAMI o los que se encuentran en el sitio del proyecto de cada marco.

Inicie sesión en Your DLAMI

Tras iniciar sesión en el servidor, verá el «mensaje del día» (MOTD) en el que se describen varios comandos de Conda que puede utilizar para cambiar entre los distintos marcos de aprendizaje profundo. A continuación se muestra un ejemploMOTD. Su información específica MOTD puede variar a medida que DLAMI se publiquen nuevas versiones del.

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

Inicie el TensorFlow entorno

> **Note**
>
> Cuando lance su primer entorno Conda, tenga paciencia mientras se carga. El aprendizaje profundo AMI con Conda instala automáticamente la versión más optimizada del marco para su EC2 instancia tras la primera activación del marco. No debe esperar retrasos posteriores.

1. Active el entorno TensorFlow virtual para Python 3.

```
$ source activate tensorflow2_p310
```

2. Inicie la iPython terminal.

```
(tensorflow2_p310)$ ipython
```

3. Ejecute un TensorFlow programa rápido.

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

Debería aparecer "Hello, Tensorflow!"

Tema siguiente

Ejecución de los tutoriales del cuaderno de Jupyter

Cambie al entorno PyTorch Python 3

Si todavía estás en la iPython consola`quit()`, úsala y prepárate para cambiar de entorno.

- Active el entorno PyTorch virtual para Python 3.

```
$ source activate pytorch_p310
```

Pruebe algún PyTorch código

Para probar la instalación, utilice Python para escribir PyTorch código que cree e imprima una matriz.

1. Inicie la iPython terminal.

```
(pytorch_p310)$ ipython
```

2. Importar PyTorch.

```
import torch
```

Es posible que vea un mensaje de advertencia sobre un paquete de terceros. Puede omitirlo.

3. Cree una matriz de 5x3 con los elementos inicializados de forma aleatoria. Imprima la matriz.

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

Verifique el resultado.

```
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]])
```

Eliminación de entornos

Si se queda sin espacio en elDLAMI, puede optar por desinstalar los paquetes de Conda que no esté utilizando:

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

Uso de la base de aprendizaje profundo AMI

Uso de la base de aprendizaje profundo AMI

La Base AMI incluye una plataforma básica de GPU controladores y bibliotecas de aceleración para implementar su propio entorno de aprendizaje profundo personalizado. De forma predeterminada, AMI se configura con cualquier NVIDIA CUDA versión del entorno. También puede cambiar entre diferentes versiones deCUDA. Consulte las siguientes instrucciones para saber cómo hacerlo.

Configuración de CUDA versiones

Puede verificar la CUDA versión ejecutando NVIDIA el `nvcc` programa.

```
nvcc --version
```

Puede seleccionar y verificar una CUDA versión en particular con el siguiente comando bash:

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

Para obtener más información, consulte las notas de la DLAMI versión base.

Ejecución de los tutoriales del cuaderno de Jupyter

Los tutoriales y los ejemplos vienen con el código fuente de cada uno de los proyectos de aprendizaje profundo y, en la mayoría de los casos, se ejecutarán en cualquiera DLAMI de ellos. Si eligió la Aprendizaje profundo AMI con Conda, obtendrá la ventaja añadida de unos tutoriales seleccionados específicamente, preconfigurados y listos para probarlos.

> **Important**
>
> Para ejecutar los tutoriales del cuaderno de Jupyter instalados en elDLAMI, tendrás que hacerlo. Configuración de un servidor de cuadernos de Jupyter

Cuando el servidor de Jupyter esté en funcionamiento, puede ejecutar los tutoriales a través de un navegador web. Si está ejecutando el aprendizaje profundo AMI con Conda o si ha configurado entornos de Python, puede cambiar los núcleos de Python desde la interfaz del bloc de notas de Jupyter. Seleccione el kernel adecuado antes de ejecutar un tutorial específico para un marco de trabajo. Se proporcionan más ejemplos de esto para los usuarios del aprendizaje profundo con Conda. AMI

> **Note**
>
> Muchos tutoriales requieren módulos de Python adicionales que pueden no estar configurados en su ordenadorDLAMI. Si recibes un error como iniciar sesión `"xyz module not found"`DLAMI, activar el entorno tal y como se ha descrito anteriormente e instalar los módulos necesarios.

> **Tip**
>
> Los tutoriales y ejemplos de aprendizaje profundo suelen basarse en uno o másGPUs. Si tu tipo de instancia no tiene unaGPU, es posible que tengas que cambiar parte del código del ejemplo para que se ejecute.

Navegación por los tutoriales instalados

Cuando hayas iniciado sesión en el servidor de Jupyter y puedas ver el directorio de tutoriales (solo en Deep Learning AMI con Conda), verás carpetas de tutoriales con cada nombre de framework. Si no ves ningún framework en la lista, significa que los tutoriales para ese framework no están disponibles en el tuyo actual. DLAMI Haga clic en el nombre del marco de trabajo para ver los tutoriales de la lista y, a continuación, haga clic en un tutorial para lanzarlo.

La primera vez que ejecute un cuaderno en el Deep Learning AMI con Conda, querrá saber qué entorno le gustaría usar. Se le pedirá que lo seleccione en una lista. El nombre de cada entorno sigue este patrón:

`Environment (conda_framework_python-version)`

Por ejemplo, puede ver`Environment (conda_mxnet_p36)`, lo que significa que el entorno tiene MXNet Python 3. La otra variación de esto sería`Environment (conda_mxnet_p27)`, lo que significa que el entorno tiene MXNet Python 2.

> **Tip**
>
> Si te preocupa qué versión de CUDA está activa, una forma de verla es MOTD cuando inicies sesión por primera vez en. DLAMI

Cambio de entorno con Jupyter

Si decide probar un tutorial para otro marco de trabajo, asegúrese de comprobar cuál es el kernel que se está ejecutando actualmente. Esta información se puede ver en la esquina superior derecha de la interfaz de Jupyter, justo debajo del botón de cerrar sesión. Puede cambiar el kernel en cualquier bloc de notas abierto haciendo clic en la opción Kernel del menú de Jupyter, seguido de Change Kernel y, a continuación, haciendo clic en el entorno correspondiente al bloc de notas que esté ejecutando.

En este punto, tendrá que volver a ejecutar todas las celdas, debido a que un cambio en el kernel borrará el estado de cualquier elemento que se haya ejecutado anteriormente.

> **Tip**
>
> Cambiar de marco de trabajo puede ser divertido y educativo, pero puede hacer que se agote la memoria. Si comienzan a aparecer errores, examine la ventana de terminal en la que se está ejecutando el servidor de Jupyter. Aquí hay mensajes útiles y un registro de errores, y es posible que veas un out-of-memory error. Para solucionar este problema, vaya a la página de inicio del servidor de Jupyter, haga clic en la pestaña Running y, a continuación, haga clic en Shutdown para cada uno de los tutoriales que probablemente sigan ejecutándose en segundo plano y estén consumiendo toda la memoria.

Tutoriales

Los siguientes son tutoriales sobre cómo utilizar el aprendizaje profundo AMI con el software de Conda.

Temas

Activación de los marcos de trabajo

Los siguientes son los marcos de aprendizaje profundo instalados en el Deep Learning AMI with Conda. Haga clic en un marco de trabajo para obtener información acerca de cómo activarlo.

Temas

PyTorch

¿Activando PyTorch

Cuando se lanza un paquete Conda estable de un framework, se prueba y se preinstala en el. DLAMI Si desea ejecutar la última compilación nocturna sin probar, puede PyTorchInstall's Nightly Build (experimental) manualmente.

Para activar el marco actualmente instalado, siga estas instrucciones en su Deep Learning AMI con Conda.

Para PyTorch Python 3 con CUDA y MKL -DNN, ejecute este comando:

```
$ source activate pytorch_p310
```

Inicie la iPython terminal.

```
(pytorch_p310)$ ipython
```

Ejecute un PyTorch programa rápido.

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

Debería ver la matriz aleatoria inicial, a continuación su tamaño y, por último, la adición de otra matriz aleatoria.

PyTorchInstall's Nightly Build (experimental)

¿Cómo realizar una instalación a PyTorch partir de una compilación nocturna

Puede instalar la versión más reciente en PyTorch uno de los entornos de PyTorch Conda o en ambos de su Deep Learning AMI con Conda.

1. • (Opción para Python 3): active el PyTorch entorno Python 3:

   ```
   $ source activate pytorch_p310
   ```

2. En los pasos restantes se da por hecho que está utilizando el entorno `pytorch_p310`. Elimine lo que está instalado actualmente PyTorch:

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

3.
   - (Opción para GPU instancias) - Instala la última versión nocturna PyTorch con CUDA .0:

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

   - (Opción para CPU instancias): instala la última versión nocturna de las instancias que no PyTorch tengan: GPUs

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

4. Para comprobar que has instalado correctamente la última versión nocturna, inicia el IPython terminal y comprueba la versión de. PyTorch

```
(pytorch_p310)$ ipython
```

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

   El resultado debería ser similar a `1.0.0.dev20180922`

5. Para comprobar que la compilación PyTorch nocturna funciona bien con el MNIST ejemplo, puedes ejecutar un script de prueba desde el repositorio PyTorch de ejemplos:

```
(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
```

Más tutoriales

Para obtener más tutoriales y ejemplos, consulta los documentos oficiales, la PyTorch documentación y el PyTorchsitio web del framework.

TensorFlow 2.

Este tutorial muestra cómo activar TensorFlow 2 en una instancia que ejecuta el Deep Learning AMI con Conda (DLAMIen Conda) y cómo ejecutar un TensorFlow programa de 2.

Cuando se publica un paquete Conda estable de un framework, se prueba y se preinstala en el. DLAMI

Activando 2 TensorFlow

¿Para correr TensorFlow DLAMI con Conda

1. Para activar TensorFlow 2, abra una instancia de Amazon Elastic Compute Cloud (AmazonEC2) DLAMI con Conda.
2. Para TensorFlow 2 y Keras 2 en Python 3 con CUDA 10.1 y MKL -DNN, ejecute este comando:

   ```
   $ source activate tensorflow2_p310
   ```
3. Inicie la iPython terminal:

   ```
   (tensorflow2_p310)$ ipython
   ```
4. Ejecute un programa TensorFlow 2 para comprobar que funciona correctamente:

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

   `Hello, TensorFlow!` debe aparecer en la pantalla.

Más tutoriales

Para obtener más tutoriales y ejemplos, consulte la TensorFlow documentación de TensorFlow Python API o visite el TensorFlowsitio web.

Capacitación distribuida con el adaptador Elastic Fabric

Un adaptador Elastic Fabric (EFA) es un dispositivo de red que se puede conectar a la DLAMI instancia para acelerar las aplicaciones de computación de alto rendimiento (HPC). EFAle permite

alcanzar el rendimiento de las aplicaciones de un HPC clúster local, con la escalabilidad, la flexibilidad y la elasticidad que proporciona el AWS Nube.

En los siguientes temas se muestra cómo empezar a utilizar EFA elDLAMI.

> **Note**
>
> Elija el suyo DLAMI de esta GPUDLAMIlista base

Temas

- Lanzando un AWS Deep Learning AMIs Instancia con EFA
- EFAUtilizándolo en el DLAMI

Lanzando un AWS Deep Learning AMIs Instancia con EFA

La última versión de Base DLAMI está lista para usarse EFA e incluye los controladores necesarios, los módulos del núcleo, libfabric, openmpi y el NCCLOFIcomplemento para las instancias. GPU

Encontrará las CUDA versiones compatibles de una Base DLAMI en las notas de la versión.

Nota:

- Al ejecutar una NCCL aplicación utilizando `mpirun` onEFA, tendrá que especificar la ruta completa a la instalación EFA compatible de la siguiente manera:

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

- Para permitir el uso de la aplicaciónEFA, añada `FI_PROVIDER="efa"` al `mpirun` comando como se muestra enEFAUtilizándolo en el DLAMI.

Temas

- Preparación de un grupo de seguridad habilitado para EFA
- Lanzar la instancia
- Verificación de una asociación de EFA

Preparación de un grupo de seguridad habilitado para EFA

EFArequiere un grupo de seguridad que permita todo el tráfico entrante y saliente hacia y desde el propio grupo de seguridad. Para obtener más información, consulte la EFA documentación.

1. Abre la EC2 consola de Amazon en https://console.aws.amazon.com/ec2/.
2. En el panel de navegación, elija Security Groups (Grupos de seguridad) y, a continuación, elija Create Security Group (Crear grupo de seguridad).
3. En la ventana Create Security Group, haga lo siguiente:
   - En Nombre del grupo de seguridad, ingrese un nombre descriptivo para el grupo de seguridad, como, por ejemplo, `EFA-enabled security group`.
   - (Opcional) En Descripción, ingrese una breve descripción del grupo de seguridad.
   - Para ello VPC, seleccione la instancia VPC en la que desea lanzar sus instancias EFA habilitadas para dispositivos móviles.
   - Seleccione Crear.
4. Seleccione el grupo de seguridad que ha creado y, en la pestaña Description (Descripción), copie el Group ID (ID de grupo).
5. En las pestañas Entrante y Saliente, haga lo siguiente:
   - Elija Edit (Editar).
   - En Type (Tipo), seleccione All traffic (Todo el tráfico).
   - En Source (Origen), seleccione Custom (Personalizado).
   - Pegue el ID del grupo de seguridad que copió en el campo.
   - Seleccione Guardar.
6. Habilite el tráfico de entrada que hace referencia a Autorización del tráfico de entrada para sus instancias de Linux. Si omite este paso, no podrá comunicarse con su DLAMI instancia.

Lanzar la instancia

EFAen el AWS Deep Learning AMIs actualmente es compatible con los siguientes tipos de instancias y sistemas operativos:

- P3DN.24xlarge: Amazon Linux 2, Ubuntu 20.04
- P4D.24xlarge: Amazon Linux 2, Ubuntu 20.04
- p5.48xlarge: Amazon Linux 2, Ubuntu 20.04

En la siguiente sección, se muestra cómo lanzar una instancia habilitada. EFA DLAMI Para obtener más información sobre el lanzamiento de una instancia EFA habilitada, consulte Lanzar instancias EFA habilitadas en un grupo de ubicación en clústeres.

1. Abre la EC2 consola de Amazon en https://console.aws.amazon.com/ec2/.
2. Elija Iniciar instancia.
3. En la AMI página Elige una, selecciona una compatible que DLAMI se encuentra en la página de notas DLAMI de la versión
4. En la página Elegir un tipo de instancia , seleccione uno de los tipos de instancias admitidos y, a continuación, elija Next: Configure Instance Details. Consulte este enlace para ver la lista de instancias compatibles: Comience con EFA y MPI
5. En la página Configurar detalles de instancia, haga lo siguiente:
   - En Número de instancias, introduce el número de instancias EFA habilitadas que deseas lanzar.
   - En Red y subred, selecciona la subred VPC y en la que quieres lanzar las instancias.
   - [Opcional] Para el grupo de ubicación, seleccione Agregar instancia al grupo de ubicación. Para lograr el mejor rendimiento, lance las instancias dentro de un grupo de ubicación.
   - [Opcional] En el nombre del grupo de ubicación, seleccione Añadir a un nuevo grupo de ubicación, introduzca un nombre descriptivo para el grupo de ubicación y, a continuación, en Estrategia de grupo de ubicación, seleccione clúster.
   - Asegúrese de activar el “Elastic Fabric Adapter” en esta página. Si esta opción está deshabilitada, cambie la subred por una que admita el tipo de instancia seleccionado.
   - En la sección Interfaces de red, para el dispositivo eth0, elija Nueva interfaz de red. Si lo desea, puede especificar una IPv4 dirección principal y una o más IPv4 direcciones secundarias. Si vas a lanzar la instancia en una subred que tiene un IPv6 CIDR bloque asociado, si lo deseas, puedes especificar una IPv6 dirección principal y una o más IPv6 direcciones secundarias.
   - Elija Siguiente: Añadir almacenamiento.
6. En la página Añadir almacenamiento, especifica los volúmenes que deseas adjuntar a las instancias además de los volúmenes especificados en ella AMI (como el volumen del dispositivo raíz) y, a continuación, selecciona Siguiente: Añadir etiquetas.
7. En la página Añadir etiquetas, especifique etiquetas para las instancias, por ejemplo, un nombre fácil de recordar, y, a continuación, elija Siguiente: Configurar grupo de seguridad.

8. En la página Configurar un grupo de seguridad, en Asignar un grupo de seguridad, seleccione Seleccionar un grupo de seguridad existente y, a continuación, seleccione el grupo de seguridad que creó anteriormente.
9. Elija Review and Launch (Revisar y lanzar).
10. En la página Revisar inicialización de instancia, revise la configuración y, a continuación, elija Iniciar para elegir un par de claves e iniciar las instancias.

Verificación de una asociación de EFA

En la consola

Tras lanzar la instancia, compruebe los detalles de la instancia en la AWS Consola. Para ello, selecciona la instancia en la EC2 consola y consulta la pestaña de descripción en el panel inferior de la página. Busque el parámetro “Network Interfaces: eth0” y haga clic en eth0 para que aparezca una ventana emergente. Asegúrese de que la opción “Elastic Fabric Adapter” esté habilitada.

Si no EFA está habilitada, puedes solucionar este problema de una de las siguientes maneras:

- Finalizar la EC2 instancia y lanzar una nueva siguiendo los mismos pasos. Asegúrese de que EFA esté asociado.
- Asocie EFA a una instancia existente.

  1. En la EC2 consola, vaya a Interfaces de red.
  2. Haga clic en Create a Network Interface (Crear una interfaz de red).
  3. Seleccione la misma subred en la que se encuentra la instancia.
  4. Asegúrese de habilitar “Elastic Fabric Adapter” y haga clic en Crear.
  5. Vuelva a la pestaña EC2 Instancias y seleccione su instancia.
  6. Ve a Acciones: estado de la instancia y detiene la instancia antes de adjuntarlaEFA.
  7. En Actions (Acciones), seleccione Networking: Attach Network Interface (Redes: Asociar interfaz de red).
  8. Seleccione la interfaz que acaba de crear y haga clic en Attach (Asociar).
  9. Reinicie la instancia.

En la instancia

El siguiente script de prueba ya está presente enDLAMI. Ejecútelo para asegurarse de que los módulos de kernel estén cargados correctamente.

```
$ fi_info -p efa
```

El resultado debería tener un aspecto similar al siguiente.

```
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
```

Verificación de la configuración del grupo de seguridad

El siguiente script de prueba ya está presente en elDLAMI. Ejecútelo para asegurarse de que el grupo de seguridad que creó esté configurado correctamente.

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

El resultado debería tener un aspecto similar al siguiente.

```
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
```

Si deja de responder o no se completa, asegúrese de que el grupo de seguridad tenga las reglas de entrada/salida correctas.

EFAUtilizándolo en el DLAMI

En la siguiente sección se describe cómo EFA ejecutar aplicaciones de varios nodos en AWS Deep Learning AMIs.

Ejecución de aplicaciones de varios nodos con EFA

Para ejecutar una aplicación en un clúster de nodos, se requiere la siguiente configuración

Temas

- Habilite la opción sin contraseña SSH
- Creación de archivo de hosts
- NCCLPruebas

Habilite la opción sin contraseña SSH

Seleccione un nodo del clúster como nodo principal. Los nodos restantes se denominan nodos miembro.

1. En el nodo líder, genere el par de claves. RSA

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

2. Cambie los permisos de la clave privada en el nodo principal.

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

3. Copie la clave `~/.ssh/id_rsa.pub` pública y añádala a `~/.ssh/authorized_keys` los nodos miembros del clúster.

4. Ahora debe poder iniciar sesión directamente en los nodos miembro desde el nodo principal mediante la ip privada.

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

5. Desactive la strictHostKey comprobación y habilite el reenvío de agentes en el nodo líder añadiendo lo siguiente al archivo ~/.ssh/config del nodo líder:

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

6. En las instancias de Amazon Linux 2, ejecute el siguiente comando en el nodo principal para proporcionar los permisos correctos al archivo de configuración:

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

Creación de archivo de hosts

En el nodo principal, cree un archivo de hosts para identificar los nodos del clúster. El archivo de hosts debe tener una entrada para cada nodo del clúster. Cree un archivo ~/hosts y añada cada nodo mediante la ip privada de la siguiente manera:

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

NCCLPruebas

> **Note**
>
> Estas pruebas se han realizado con la EFA versión 1.30.0 y el OFI NCCL complemento 1.7.4.

A continuación se muestra un subconjunto de NCCL pruebas realizadas por Nvidia para probar tanto la funcionalidad como el rendimiento en varios nodos de procesamiento

Instancias compatibles: P3dn, P4, P5

Pruebas de funcionalidad

NCCLPrueba de transferencia de mensajes con varios nodos

El nccl_message_transfer es una prueba sencilla para garantizar que el NCCL OFI complemento funcione según lo esperado. La prueba valida la funcionalidad del establecimiento de la conexión y la transferencia de datosNCCL. APIs Asegúrese de utilizar la ruta completa para ejecutar mpirun, como se muestra en el ejemplo, mientras ejecuta NCCL aplicaciones con. EFA Cambie los parámetros np en N función del número de instancias y GPUs del clúster. Para obtener más información, consulte la .AWS OFINCCLdocumentación.

La siguiente prueba nccl_message_transfer es para una versión xx.x genérica. CUDA Puedes ejecutar los comandos para cualquier CUDA versión disponible en tu EC2 instancia de Amazon sustituyendo la CUDA versión en el script.

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

El resultado debería tener el siguiente aspecto. Puedes comprobar el resultado para ver si se EFA está utilizando como OFI proveedor.

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

```
INFO: Function: main Line: 145: NET/OFI Send 8 requests to rank 1
INFO: Function: main Line: 179: NET/OFI Server: Start accepting requests
INFO: Function: main Line: 183: NET/OFI Successfully accepted connection from rank 0
INFO: Function: main Line: 187: NET/OFI Rank 1 posting 8 receive buffers
INFO: Function: main Line: 161: NET/OFI Successfully sent 8 requests to rank 1
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 0
INFO: Function: main Line: 251: NET/OFI Got completions for 8 requests for rank 1
```

Pruebas de rendimiento

Prueba de NCCL rendimiento de varios nodos en P4D.24xlarge

Para comprobar NCCL el rendimientoEFA, ejecute la prueba de NCCL rendimiento estándar que está disponible en el repositorio oficial de NCCL-Tests. DLAMIViene con esta prueba ya creada para CUDA XX.X. También puedes ejecutar tu propio script con ella. EFA

Cuando cree su propio script, siga estas directrices:

- Utilice la ruta completa para ejecutar mpirun, como se muestra en el ejemplo, mientras ejecuta aplicaciones con. NCCL EFA
- Cambie los parámetros np y N en función del número de instancias y GPUs del clúster.
- Agrega el INFO indicador NCCL _ DEBUG = y asegúrate de que los registros indiquen el EFA uso como «El proveedor seleccionado esEFA».
- Defina la ubicación del registro de entrenamiento para analizarla y validarla

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

Use el comando `watch nvidia-smi` en cualquiera de los nodos miembros para monitorear el GPU uso. Los siguientes `watch nvidia-smi` comandos son para una versión CUDA xx.x genérica y dependen del sistema operativo de la instancia. Puedes ejecutar los comandos para cualquier CUDA versión disponible en tu EC2 instancia de Amazon sustituyendo la CUDA versión en el script.

- Amazon Linux 2:

```
 $ /opt/amazon/openmpi/bin/mpirun -n 16 -N 8 \
-x NCCL_DEBUG=INFO --mca pml ^cm \
-x LD_LIBRARY_PATH=/usr/local/cuda-xx.x/efa/lib:/usr/local/cuda-xx.x/lib:/usr/
local/cuda-xx.x/lib64:/usr/local/cuda-xx.x:/opt/amazon/efa/lib64:/opt/amazon/openmpi/
lib64:$LD_LIBRARY_PATH \
```

```
--hostfile hosts --mca btl tcp,self --mca btl_tcp_if_exclude lo,docker0 --bind-to
 none \
/usr/local/cuda-xx.x/efa/test-cuda-xx.x/all_reduce_perf -b 8 -e 1G -f 2 -g 1 -c 1 -n
 100 | tee ${TRAINING_LOG}
```

- Ubuntu 20.04:

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

El resultado debería tener el siguiente aspecto:

```
# nThread 1 nGpus 1 minBytes 8 maxBytes 1073741824 step: 2(factor) warmup iters: 5
 iters: 100 agg iters: 1 validation: 1 graph: 0
#
# Using devices
#  Rank  0 Group  0 Pid   9591 on ip-172-31-4-37 device  0 [0x10] NVIDIA A100-SXM4-40GB
#  Rank  1 Group  0 Pid   9592 on ip-172-31-4-37 device  1 [0x10] NVIDIA A100-SXM4-40GB
#  Rank  2 Group  0 Pid   9593 on ip-172-31-4-37 device  2 [0x20] NVIDIA A100-SXM4-40GB
#  Rank  3 Group  0 Pid   9594 on ip-172-31-4-37 device  3 [0x20] NVIDIA A100-SXM4-40GB
#  Rank  4 Group  0 Pid   9595 on ip-172-31-4-37 device  4 [0x90] NVIDIA A100-SXM4-40GB
#  Rank  5 Group  0 Pid   9596 on ip-172-31-4-37 device  5 [0x90] NVIDIA A100-SXM4-40GB
#  Rank  6 Group  0 Pid   9597 on ip-172-31-4-37 device  6 [0xa0] NVIDIA A100-SXM4-40GB
#  Rank  7 Group  0 Pid   9598 on ip-172-31-4-37 device  7 [0xa0] NVIDIA A100-SXM4-40GB
#  Rank  8 Group  0 Pid  10216 on ip-172-31-13-179 device  0 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank  9 Group  0 Pid  10217 on ip-172-31-13-179 device  1 [0x10] NVIDIA A100-
SXM4-40GB
#  Rank 10 Group  0 Pid  10218 on ip-172-31-13-179 device  2 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank 11 Group  0 Pid  10219 on ip-172-31-13-179 device  3 [0x20] NVIDIA A100-
SXM4-40GB
#  Rank 12 Group  0 Pid  10220 on ip-172-31-13-179 device  4 [0x90] NVIDIA A100-
SXM4-40GB
```

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

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

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

Pruebas de validación

Para validar que las EFA pruebas arrojaron un resultado válido, utilice las siguientes pruebas para confirmarlo:

- Obtenga el tipo de instancia mediante los metadatos de la EC2 instancia:

```
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)
```

- Ejecute la Pruebas de rendimiento
- Establezca los siguientes parámetros

```
CUDA_VERSION
CUDA_RUNTIME_VERSION
NCCL_VERSION
```

- Valide los resultados como se muestra:

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

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

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

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

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

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

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

- Para acceder a los datos de referencia, podemos analizar la última fila del resultado de la tabla de la prueba all_reduce de varios nodos:

```
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}"
```

GPUSupervisión y optimización

La siguiente sección lo guiará a través de las opciones de GPU optimización y monitoreo. Esta sección está organizada como un flujo de trabajo típico en la que se monitoriza el procesamiento previo y el entrenamiento.

- Monitorización
  - GPUsSupervise con CloudWatch
- Optimización
  - Procesamiento previo
  - Formación

Monitorización

DLAMIViene preinstalado con varias herramientas GPU de monitoreo. Esta guía también menciona las herramientas que están disponibles para su descarga e instalación.

- GPUsSupervise con CloudWatch- una utilidad preinstalada que informa a Amazon GPU CloudWatch de las estadísticas de uso.
- nvidia-smi CLI: una utilidad para monitorear el uso general de cómputo y memoria. GPU Está preinstalado en su AWS Deep Learning AMIs (DLAMI).
- NVMLBiblioteca C: basada en C para acceder directamente API a las funciones de GPU supervisión y administración. Lo utiliza la nvidia-smi CLI bajo el capó y viene preinstalado en su. DLAMI También tiene enlaces a Python y Perl para facilitar el desarrollo en dichos lenguajes. La utilidad gpumon.py preinstalada utiliza el paquete pynvml de. DLAMI nvidia-ml-py
- NVIDIADCGM- Una herramienta de administración de clústeres. Visite la página del desarrollador para obtener información sobre cómo instalar y configurar esta herramienta.

> **Tip**
>
> Consulta NVIDIA el blog para desarrolladores para obtener la información más reciente sobre el uso de las CUDA herramientas instaladas en tuDLAMI:
>
> - Supervisión TensorCore del uso mediante Nsight IDE y nvprof.

GPUsSupervise con CloudWatch

Cuando usas tu DLAMI con un, GPU es posible que descubras que estás buscando formas de rastrear su uso durante el entrenamiento o la inferencia. Esto puede resultar útil para optimizar la canalización de datos y ajustar la red de aprendizaje profundo.

Hay dos formas de configurar GPU las métricas con CloudWatch:

- Configure las métricas con AWS CloudWatch agente (recomendado)
- Configure las métricas con el script preinstalado `gpumon.py`

Configure las métricas con AWS CloudWatch agente (recomendado)

Intégrelo DLAMI con el CloudWatch agente unificado para configurar GPU las métricas y supervisar la utilización de GPU los coprocesos en las instancias EC2 aceleradas de Amazon.

Hay cuatro formas de configurar GPUlas métricas con susDLAMI:

- Configure GPU métricas mínimas
- Configure métricas parciales GPU
- Configura todas las métricas disponibles GPU
- Configura métricas personalizadas GPU

Para obtener más información sobre actualizaciones y parches de seguridad, consulte Parches de seguridad para el AWS CloudWatch agente.

Requisitos previos

Para empezar, debes configurar IAM los permisos de EC2 instancia de Amazon que permitan a tu instancia enviar métricas a CloudWatch. Para ver los pasos detallados, consulta Crear IAM roles y usuarios para usarlos con el CloudWatch agente.

Configure GPU métricas mínimas

Configure GPU las métricas mínimas mediante el `dlami-cloudwatch-agent@minimal` `systemd` servicio. En este servicio se configuran las siguientes métricas:

- `utilization_gpu`
- `utilization_memory`

Puede encontrar el `systemd` servicio de GPU métricas mínimas preconfiguradas en la siguiente ubicación:

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

Active e inicie el servicio `systemd` con los siguientes comandos:

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

Configure métricas parciales GPU

Configure GPU métricas parciales mediante el `dlami-cloudwatch-agent@partial` `systemd` servicio. En este servicio se configuran las siguientes métricas:

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

Puede encontrar el `systemd` servicio de GPU métricas preconfiguradas parciales en la siguiente ubicación:

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

Active e inicie el servicio `systemd` con los siguientes comandos:

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

Configura todas las métricas disponibles GPU

Configure todas GPU las métricas disponibles mediante el `dlami-cloudwatch-agent@all` `systemd` servicio. En este servicio se configuran las siguientes métricas:

- `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

Puede encontrar el systemd servicio para todas las GPU métricas preconfiguradas disponibles en la siguiente ubicación:

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

Active e inicie el servicio systemd con los siguientes comandos:

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

Configura métricas personalizadas GPU

Si las métricas preconfiguradas no cumplen sus requisitos, puede crear un archivo de configuración de CloudWatch agente personalizado.

Para crear un archivo de configuración personalizado

Para crear un archivo de configuración personalizado, consulte los pasos detallados en Crear o editar manualmente el archivo de configuración del CloudWatch agente.

Para este ejemplo, suponga que la definición del esquema se encuentra en /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json.

Configure las métricas con su archivo personalizado

Ejecute el siguiente comando para configurar el CloudWatch agente según su archivo personalizado:

```
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
```

Parches de seguridad para el AWS CloudWatch agente

Los recién lanzados DLAMIs están configurados con la última versión disponible AWS CloudWatch parches de seguridad para agentes. Consulte las siguientes secciones para actualizar los parches de seguridad actuales DLAMI con los más recientes, en función del sistema operativo que elija.

Amazon Linux 2

yumÚselo para obtener las últimas AWS CloudWatch parches de seguridad del agente para Amazon Linux 2DLAMI.

```
sudo yum update
```

Ubuntu

Para obtener las últimas AWS CloudWatch parches de seguridad para DLAMI un Ubuntu, es necesario volver a instalar el AWS CloudWatch agente mediante un enlace de descarga de Amazon S3.

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

Para obtener más información sobre la instalación del AWS CloudWatch utilice los enlaces de descarga de Amazon S3, consulte Instalación y ejecución del CloudWatch agente en sus servidores.

Configure las métricas con el script preinstalado **`gpumon.py`**

Una utilidad llamada gpumon.py viene preinstalada en su. DLAMI Se integra CloudWatch y admite la supervisión de cada GPU uso: GPU memoria, GPU temperatura y GPU alimentación. El script envía periódicamente los datos monitoreados a CloudWatch. Puede configurar el nivel de granularidad de los datos a los que se envían CloudWatch cambiando algunos ajustes del script. Sin embargo, antes de iniciar el script, tendrá que configurarlo CloudWatch para recibir las métricas.

Cómo configurar y ejecutar la GPU supervisión con CloudWatch

1. Cree un IAM usuario o modifique uno existente para tener una política en la que publicar la métrica CloudWatch. Si crea un usuario nuevo, anote las credenciales, ya que las necesitará en el siguiente paso.

La IAM política que se debe buscar es «cloudwatch:PutMetricData». La política que se añade es la siguiente:

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

> **Tip**
>
> Para obtener más información sobre cómo crear un IAM usuario y añadir políticas CloudWatch, consulte la CloudWatch documentación.

2. A tu ladoDLAMI, ejecuta AWS configure y especifique las credenciales IAM de usuario.

```
$ aws configure
```

3. Es posible que tenga que realizar algunas modificaciones en la utilidad gpumon antes de ejecutarla. Puede encontrar la utilidad gpumon y README en la ubicación definida en el siguiente bloque de códigos. Para obtener más información sobre el script `gpumon.py`, consulte la ubicación del script en Amazon S3.

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

Opciones:

- Cambia la región en gpumon.py si tu instancia está NOT en us-east-1.
- Cambie otros parámetros, como el CloudWatch `namespace` o el período del informe, con. `store_reso`

4. Actualmente, el script solo es compatible con Python 3. Active el entorno Python 3 de su framework preferido o active el entorno Python 3 DLAMI general.

```
$ source activate python3
```

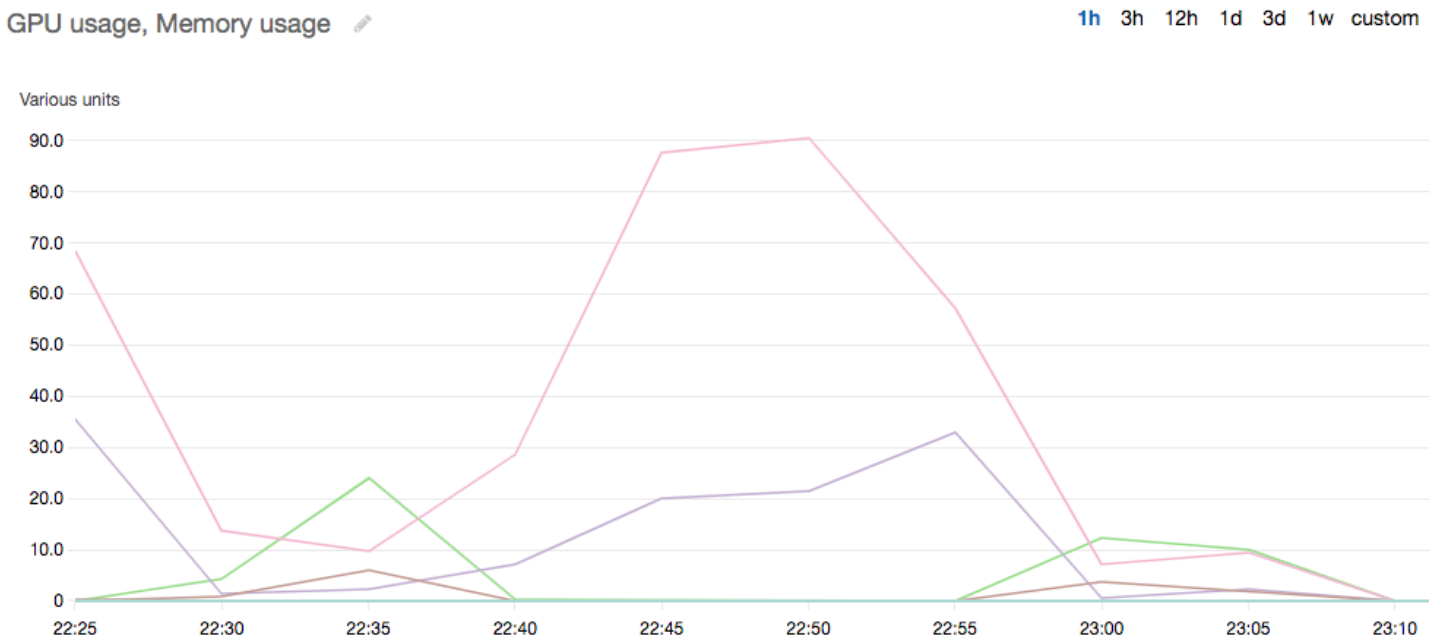
5. Ejecute la utilidad gpumon en segundo plano.

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

6. Abra su navegador para ver la métrica y, a https://console.aws.amazon.com/cloudwatch/continuación, seleccione. Tendrá un espacio de nombres ". DeepLearningTrain

> **Tip**
>
> Puede cambiar el espacio de nombres modificando gpumon.py. También puede modificar el intervalo de notificación ajustando `store_reso`.

El siguiente es un ejemplo de CloudWatch gráfico que informa sobre una ejecución de gpumon.py supervisando un trabajo de entrenamiento en una instancia p2.8xlarge.



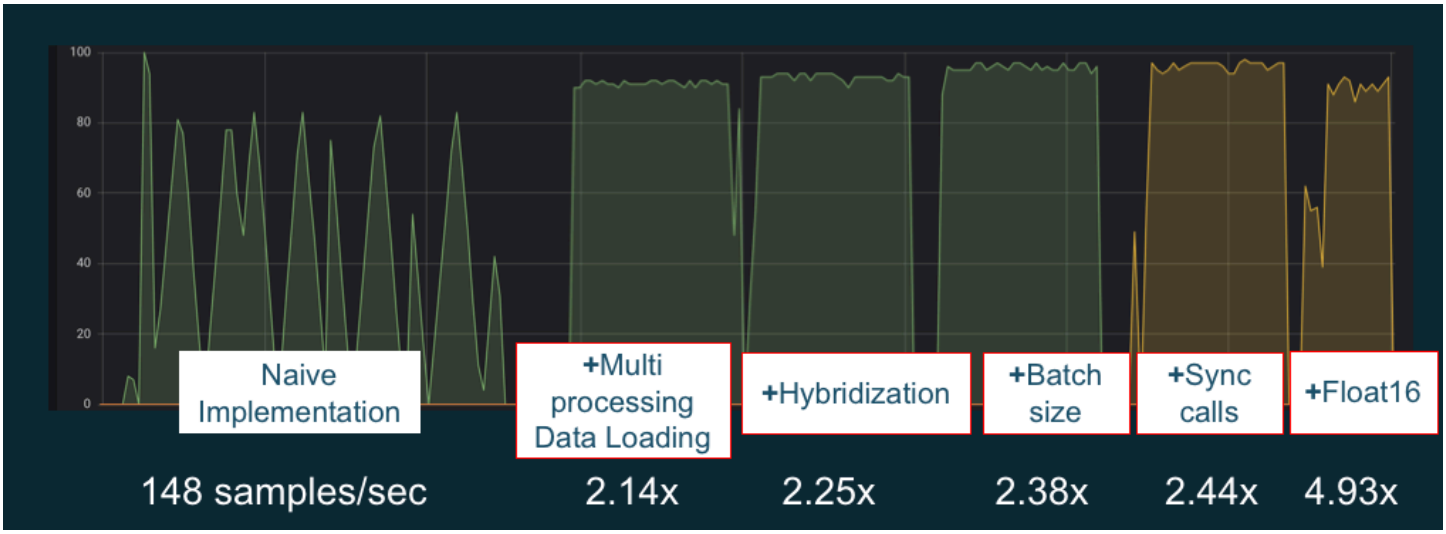
Puede que le interesen estos otros temas sobre la GPU supervisión y la optimización:

- Monitorización

Optimización

Para sacarle el máximo partidoGPUs, puede optimizar su canalización de datos y ajustar su red de aprendizaje profundo. Como se describe en el siguiente gráfico, una implementación básica o ingenua de una red neuronal podría utilizar su potencial de GPU manera inconsistente y no aprovechada al máximo. Cuando optimiza el preprocesamiento y la carga de datos, puede reducir el cuello de botella que va del suyo al suyo. CPU GPU Puede ajustar la propia red neuronal mediante la hibridación (si el marco de trabajo la admite), ajustando el tamaño del lote y sincronizando las llamadas. También puede utilizar entrenamiento de precisión múltiple (float16 o int8) en la mayoría de los marcos de trabajo, lo que puede tener un efecto drástico en la mejora del rendimiento.

El gráfico siguiente muestra la mejora acumulativa del rendimiento que se obtiene cuando se aplican distintas optimizaciones. Los resultados dependerán de los datos que esté procesando y de la red que esté optimizando.



Ejemplos de optimizaciones de GPU rendimiento. Fuente del gráfico: Performance Tricks with Gluon MXNet

Las siguientes guías presentan opciones que se adaptarán a sus necesidades DLAMI y le ayudarán a mejorar su GPU rendimiento.

Temas

Procesamiento previo

El preprocesamiento de los datos mediante transformaciones o ampliaciones suele ser un proceso CPU limitado, lo que puede suponer un obstáculo en su proceso general. Los marcos tienen operadores integrados para el procesamiento de imágenes, pero DALI (Biblioteca de aumento de datos) demuestra un rendimiento mejorado en comparación con las opciones integradas de los marcos.

- NVIDIABiblioteca de aumento de datos (DALI): transfiere el aumento DALI de datos al. GPU No está preinstalado en elDLAMI, pero puede acceder a él instalándolo o cargando un contenedor de marco compatible en su instancia DLAMI o en otra instancia de Amazon Elastic Compute Cloud. Consulte la página del DALI proyecto en el NVIDIA sitio web para obtener más información. Para ver un ejemplo de caso de uso y descargar ejemplos de código, consulte el ejemplo sobre el rendimiento de la formación SageMaker previa al procesamiento.
- nvJPEG: una biblioteca de JPEG decodificadores GPU acelerados para programadores de C. Admite la decodificación de imágenes individuales o por lotes, así como las operaciones de transformación posteriores que son comunes en el aprendizaje profundo. El nv JPEG viene integradoDALI, o puede descargarlo desde la página nvjpeg del NVIDIAsitio web y usarlo por separado.

Es posible que le interesen estos otros temas sobre la supervisión y la optimización: GPU

Formación

El entrenamiento de precisión mixta permite implementar redes de mayor tamaño con la misma cantidad de memoria, o reducir el uso de esta en comparación con las redes de precisión única o

doble, lo que se traduce en un aumento del rendimiento informático. También ofrece el beneficio de transferencias de datos más pequeñas y rápidas, un factor importante en el entrenamiento distribuido con varios nodos. Para utilizar el entrenamiento de precisión mixta es necesario ajustar el envío de datos y el escalado de pérdidas. Las siguientes guías describen cómo realizar esta operación en los marcos de trabajo compatibles con la precisión mixta.

- NVIDIAAprendizaje profundo SDK: documentos en el NVIDIA sitio web que describen la implementación de precisión mixta para MXNet PyTorch, y. TensorFlow

> **Tip**
>
> Asegúrese de consultar el sitio web de su marco de trabajo preferido y busque "mixed precision" o "fp16" para conocer las técnicas de optimización más recientes. A continuación se muestran algunas guías de precisión mixta que pueden resultarle de utilidad:
>
> - Capacitación de precisión mixta con TensorFlow (vídeo): en el sitio del blog. NVIDIA
> - Entrenamiento de precisión mixta con float16 con MXNet: un artículo en el sitio web. FAQ MXNet
> - NVIDIAApex: una herramienta para un entrenamiento sencillo de precisión mixta con PyTorch un artículo de blog en el sitio web. NVIDIA

Es posible que le interesen estos otros temas sobre GPU supervisión y optimización:

- Monitorización
  - GPUsSupervise con CloudWatch
- Optimización
  - Procesamiento previo
  - Formación

La AWS Chip Inferentia con DLAMI

AWS Inferentia es un chip de aprendizaje automático personalizado diseñado por AWS que puede utilizar para predicciones de inferencias de alto rendimiento. Para usar el chip, configure una instancia de Amazon Elastic Compute Cloud y use el AWS Kit de desarrollo de software Neuron

(SDK) para invocar el chip Inferentia. Para ofrecer a los clientes la mejor experiencia de Inferentia, Neuron se ha incorporado al AWS Deep Learning AMIs (DLAMI).

Los siguientes temas le muestran cómo empezar a utilizar Inferentia con. DLAMI

Contenido

Lanzar una instancia con DLAMI AWS Neuron

La última DLAMI está lista para usarse con AWS Inferentia y viene con el AWS Paquete NeuronAPI. Para lanzar una DLAMI instancia, consulte Lanzamiento y configuración de un DLAMI. Cuando tenga unaDLAMI, siga los pasos que se indican a continuación para asegurarse de que su AWS Chip de inferencia y AWS Los recursos neuronales están activos.

Contenido

Comprobación de la instancia

Antes de usar la instancia, compruebe que esté correctamente instalada y configurada con Neuron.

Identificando AWS Dispositivos de inferencia

Para identificar el número de dispositivos de Inferencia de la instancia, utilice el siguiente comando:

```
neuron-ls
```

Si su instancia tiene dispositivos de Inferentia asociados a ella, la salida tendrá un aspecto similar al siguiente:

```
+--------+--------+--------+-----------+--------------+
```

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

El resultado suministrado se toma de una instancia INF1.6xLarge e incluye las siguientes columnas:

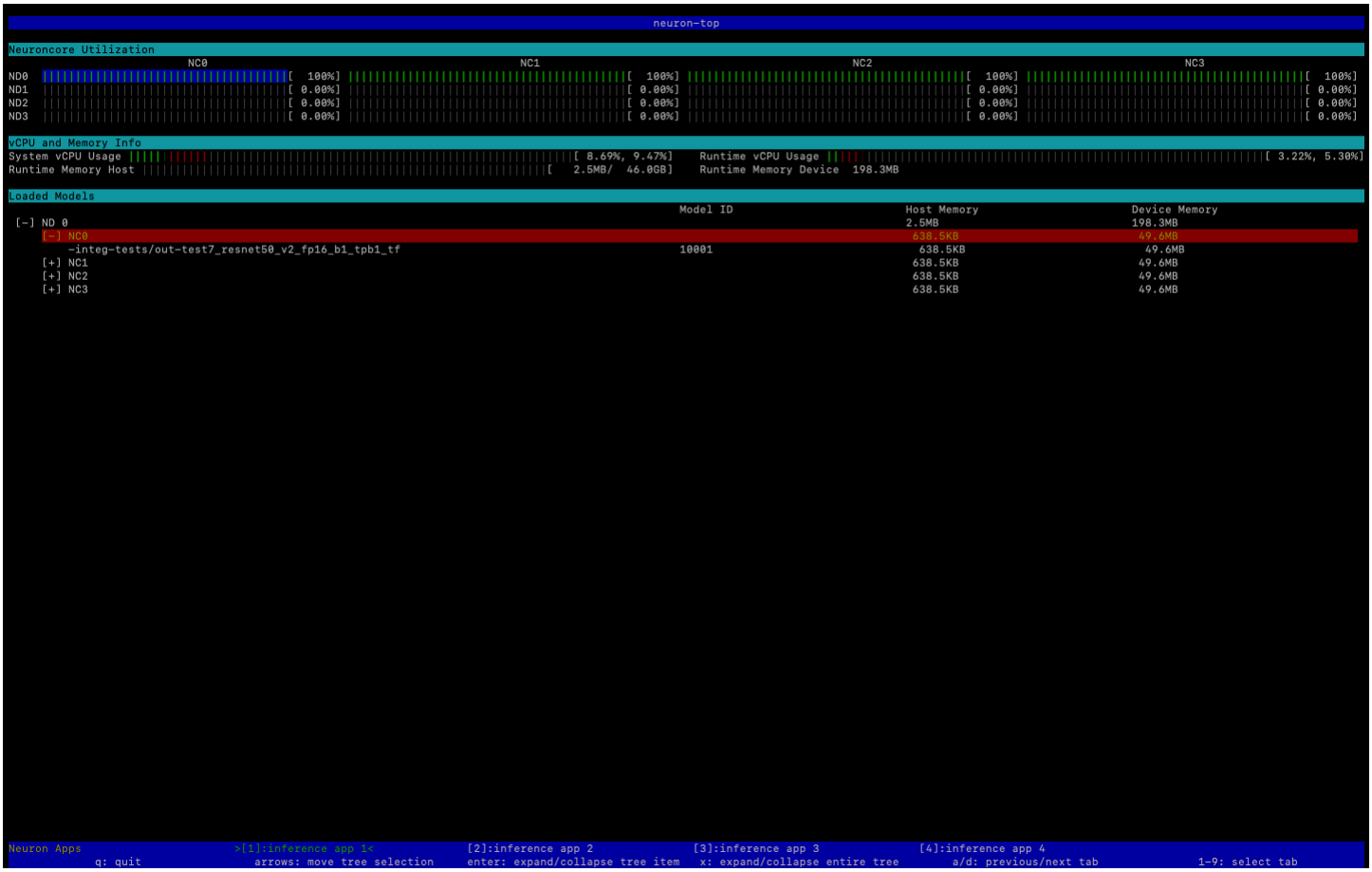
- NEURONDEVICE: El identificador lógico asignado al. NeuronDevice Este ID se usa al configurar varios tiempos de ejecución para usar diferentes NeuronDevices.
- NEURONCORES: El número de NeuronCores presentes en. NeuronDevice
- NEURONMEMORY: La cantidad de DRAM memoria del NeuronDevice.
- CONNECTEDDEVICES: Otro NeuronDevices conectado al NeuronDevice.
- PCIBDF: El identificador de función del dispositivo de PCI bus (BDF) del NeuronDevice.

Visualización del uso de recursos

Consulte información útil sobre la CPU utilización de NeuronCore y v, el uso de la memoria, los modelos cargados y las aplicaciones de Neuron con el `neuron-top` comando. Si se inicia `neuron-top` sin argumentos, se mostrarán los datos de todas las aplicaciones de aprendizaje automático que se utilicen NeuronCores.

```
neuron-top
```

Cuando una aplicación utiliza cuatro NeuronCores, el resultado debe tener un aspecto similar al de la imagen siguiente:



Para obtener más información sobre los recursos para supervisar y optimizar las aplicaciones de inferencia basadas en Neuron, consulte Neuron Tools.

Uso de Neuron Monitor (monitor de neuronas)

Neuron Monitor recopila las métricas de los tiempos de ejecución de Neuron que se ejecutan en el sistema y transmite los datos recopilados a una salida estándar. JSON Estas métricas se organizan en grupos de métricas que se configuran proporcionando un archivo de configuración. Para obtener más información sobre Neuron Monitor, consulte la User Guide for Neuron Monitor.

Actualización del software Neuron

Para obtener información sobre cómo actualizar el software SDK Neuron desde dentro, consulte la DLAMI AWS Guía de configuración de Neuron.

Paso siguiente

Usando el con DLAMI AWS Neuron

Usando el con DLAMI AWS Neuron

Un flujo de trabajo típico con AWS Neuron SDK consiste en compilar un modelo de aprendizaje automático previamente entrenado en un servidor de compilación. Después de esto, distribuya los artefactos a las instancias de Inf1 para su ejecución. AWS Deep Learning AMIs (DLAMI) viene preinstalado con todo lo que necesita para compilar y ejecutar inferencias en una instancia de Inf1 que utilice Inferentia.

En las siguientes secciones, se describe cómo usarlo con Inferentia. DLAMI

Contenido

Uso de TensorFlow -Neuron y el AWS Compilador de neuronas

En este tutorial se muestra cómo utilizar el AWS Compilador Neuron para compilar el modelo Keras ResNet -50 y exportarlo como un modelo guardado en formato. SavedModel Este formato es un formato típico TensorFlow de modelos intercambiables. También aprenderá a ejecutar la inferencia en una instancia Inf1 con entrada de ejemplo.

Para obtener más información sobre la neuronaSDK, consulte la AWS La documentación sobre las neuronas. SDK

Contenido

Requisitos previos

Antes de utilizar este tutorial, debería haber completado los pasos de configuración de Lanzar una instancia con DLAMI AWS Neuron. También debe estar familiarizado con el aprendizaje profundo y el uso del. DLAMI

Activación del entorno Conda

Active el entorno conda TensorFlow -Neuron mediante el siguiente comando:

```
source activate aws_neuron_tensorflow_p36
```

Para salir del entorno Conda actual, ejecute el siguiente comando:

```
source deactivate
```

Compilación de Resnet50

Cree un script de Python llamada denominado **`tensorflow_compile_resnet50.py`** que tenga el siguiente contenido. Este script de Python compila el modelo Keras ResNet 50 y lo exporta como un modelo guardado.

```
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')
```

Compile el modelo con el siguiente comando:

```
python tensorflow_compile_resnet50.py
```

El proceso de compilación tardará unos minutos. Cuando concluya, la salida debe tener el siguiente aspecto:

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

Después de la compilación, el modelo guardado se comprime en **`ws_resnet50/resnet50_neuron.zip`**. Descomprima el modelo y descargue la imagen de muestra para la inferencia mediante los siguientes comandos:

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

ResNet50 Inferencia

Cree un script de Python llamada denominado **`tensorflow_infer_resnet50.py`** que tenga el siguiente contenido. Este script ejecuta la inferencia en el modelo descargado utilizando un modelo de inferencia compilado previamente.

```
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])
```

Ejecute la inferencia en el modelo mediante el siguiente comando:

```
python tensorflow_infer_resnet50.py
```

El resultado debería tener el siguiente aspecto:

```
...
```

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

Paso siguiente

Utilización AWS Servicio de neuronas TensorFlow

Utilización AWS Servicio de neuronas TensorFlow

Este tutorial muestra cómo construir un gráfico y añadir un AWS Paso de compilación de neuronas antes de exportar el modelo guardado para usarlo con TensorFlow Serving. TensorFlow Serving es un sistema de servidor que permite ampliar las inferencias en una red. Neuron TensorFlow Serving utiliza lo mismo que el Serving normalAPI. TensorFlow La única diferencia es que un modelo guardado debe compilarse para AWS La inferencia y el punto de entrada tienen un nombre binario diferente. `tensorflow_model_server_neuron` El binario se encuentra en `/usr/local/bin/tensorflow_model_server_neuron` y está preinstalado en. DLAMI

Para obtener más información sobre la neuronaSDK, consulte la AWS La documentación sobre las neuronas. SDK

Contenido

Requisitos previos

Antes de utilizar este tutorial, debería haber completado los pasos de configuración de Lanzar una instancia con DLAMI AWS Neuron. También debe estar familiarizado con el aprendizaje profundo y el uso del. DLAMI

Activación del entorno Conda

Active el entorno conda TensorFlow -Neuron mediante el siguiente comando:

```
source activate aws_neuron_tensorflow_p36
```

Si necesita salir del entorno Conda actual, ejecute:

```
source deactivate
```

Compilación y exportación del modelo guardado

Cree un script de Python denominado `tensorflow-model-server-compile.py` con el siguiente contenido. Este script construye un gráfico y lo compila con Neuron. A continuación, exporta el gráfico compilado como un modelo guardado.

```
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)
```

Compile el modelo con el siguiente comando:

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

El resultado debería tener el siguiente aspecto:

```
...
```

```
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
```

Distribución del modelo guardado

Una vez compilado el modelo, puede usar el siguiente comando para distribuir el modelo guardado con el binario tensorflow_model_server_neuron:

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

El resultado debería tener el siguiente aspecto. El servidor organiza el modelo compilado en el dispositivo Inferentia para prepararlo para DRAM la inferencia.

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

Generación de solicitudes de inferencia al servidor de modelos

Cree un script de Python denominado `tensorflow-model-server-infer.py` con el siguiente contenido. Este script ejecuta la inferencia a través de gRPC, que es el marco de servicio.

```
import numpy as np
import grpc
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
```

```
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc
from tensorflow.keras.applications.resnet50 import decode_predictions

if __name__ == '__main__':
    channel = grpc.insecure_channel('localhost:8500')
    stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
    img_file = tf.keras.utils.get_file(
        "./kitten_small.jpg",
        "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
    img = image.load_img(img_file, target_size=(224, 224))
    img_array = preprocess_input(image.img_to_array(img)[None, ...])
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'resnet50_inf1'
    request.inputs['input'].CopyFrom(
        tf.contrib.util.make_tensor_proto(img_array, shape=img_array.shape))
    result = stub.Predict(request)
    prediction = tf.make_ndarray(result.outputs['output'])
    print(decode_predictions(prediction))
```

Ejecute la inferencia en el modelo mediante g RPC con el siguiente comando:

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

El resultado debería tener el siguiente aspecto:

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

Usando MXNet -Neuron y el AWS Compilador de neuronas

La compilación MXNet -Neuron API proporciona un método para compilar un gráfico modelo que se puede ejecutar en un AWS Dispositivo de inferencia.

En este ejemplo, se utiliza API para compilar un modelo ResNet -50 y para ejecutar la inferencia.

Para obtener más información sobre la neuronaSDK, consulte la AWS La documentación sobre las neuronas. SDK

Contenido

Requisitos previos

Antes de utilizar este tutorial, debería haber completado los pasos de configuración de Lanzar una instancia con DLAMI AWS Neuron. También debe estar familiarizado con el aprendizaje profundo y el uso del. DLAMI

Activación del entorno Conda

Active el entorno conda MXNet -Neuron mediante el siguiente comando:

```
source activate aws_neuron_mxnet_p36
```

Para salir del entorno Conda actual, ejecute:

```
source deactivate
```

Compilación de Resnet50

Cree un script de Python denominado **`mxnet_compile_resnet50.py`** con el siguiente contenido. Este script usa la compilación MXNet -Neuron de Python API para compilar un modelo ResNet -50.

```
import mxnet as mx
import numpy as np

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

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

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

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

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

Compile el modelo con el siguiente comando:

```
python mxnet_compile_resnet50.py
```

La compilación tardará unos minutos. Cuando haya finalizado, los siguientes archivos estarán en su directorio actual:

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

ResNetInferencia 5.0

Cree un script de Python denominado **`mxnet_infer_resnet50.py`** con el siguiente contenido. Este script descarga una imagen de muestra y la utiliza para ejecutar la inferencia con el modelo compilado.

```
import mxnet as mx
import numpy as np

path='http://data.mxnet.io/models/imagenet/'
mx.test_utils.download(path+'synset.txt')

fname = mx.test_utils.download('https://raw.githubusercontent.com/awslabs/mxnet-model-
server/master/docs/images/kitten_small.jpg')
img = mx.image.imread(fname)

# convert into format (batch, RGB, width, height)
img = mx.image.imresize(img, 224, 224)
```

```
# resize
img = img.transpose((2, 0, 1))
# Channel first
img = img.expand_dims(axis=0)
# batchify
img = img.astype(dtype='float32')

sym, args, aux = mx.model.load_checkpoint('compiled_resnet50', 0)
softmax = mx.nd.random_normal(shape=(1,))
args['softmax_label'] = softmax
args['data'] = img
# Inferentia context
ctx = mx.neuron()

exe = sym.bind(ctx=ctx, args=args, aux_states=aux, grad_req='null')
with open('synset.txt', 'r') as f:
    labels = [l.rstrip() for l in f]

exe.forward(data=img)
prob = exe.outputs[0].asnumpy()
# print the top-5
prob = np.squeeze(prob)
a = np.argsort(prob)[::-1]
for i in a[0:5]:
    print('probability=%f, class=%s' %(prob[i], labels[i]))
```

Ejecute la inferencia con el modelo compilado mediante el siguiente comando:

```
python mxnet_infer_resnet50.py
```

El resultado debería tener el siguiente aspecto:

```
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
```

Paso siguiente

Uso del servicio MXNet de modelos -Neuron

Uso del servicio MXNet de modelos -Neuron

En este tutorial, aprenderá a utilizar un MXNet modelo previamente entrenado para realizar la clasificación de imágenes en tiempo real con Multi Model Server ()MMS. MMSes una easy-to-use herramienta flexible para utilizar modelos de aprendizaje profundo que se entrenan con cualquier marco de aprendizaje automático o aprendizaje profundo. Este tutorial incluye un paso de compilación que utiliza AWS Neuron y una implementación del MMS usoMXNet.

Para obtener más información sobre la neuronaSDK, consulte la AWS La documentación sobre las neuronas. SDK

Contenido

Requisitos previos

Antes de utilizar este tutorial, debería haber completado los pasos de configuración de Lanzar una instancia con DLAMI AWS Neuron. También debe estar familiarizado con el aprendizaje profundo y el uso del. DLAMI

Activación del entorno Conda

Active el entorno MXNet conda -Neuron mediante el siguiente comando:

```
source activate aws_neuron_mxnet_p36
```

Para salir del entorno Conda actual, ejecute:

```
source deactivate
```

Descarga del código de ejemplo

Para ejecutar este ejemplo, descargue el código de ejemplo mediante los siguientes comandos:

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

```
cd multi-model-server/examples/mxnet_vision
```

Compile el modelo.

Cree un script de Python denominado `multi-model-server-compile.py` con el siguiente contenido. Este script compila el modelo ResNet 50 con el objetivo del dispositivo 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)
```

Para compilar el modelo, utilice el siguiente comando:

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

El resultado debería tener el siguiente aspecto:

```
...
[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!
```

Cree un archivo `signature.json` con el siguiente contenido para configurar el nombre y la forma de entrada:

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

Descargue el archivo `synset.txt` con el siguiente comando. Este archivo es una lista de nombres para ImageNet las clases de predicción.

```
curl -O https://s3.amazonaws.com/model-server/model_archive_1.0/examples/
squeezenet_v1.1/synset.txt
```

Cree una clase de servicio personalizada siguiendo la plantilla de la carpeta `model_server_template`. Copie la plantilla en su directorio de trabajo actual mediante el siguiente comando:

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

Edite el módulo `mxnet_model_service.py` para reemplazar el contexto `mx.cpu()` por el contexto `mx.neuron()` de la siguiente manera. También debe comentar la copia de datos innecesaria, `model_input` ya que MXNet -Neuron no es compatible con NDArray y Gluon. APIs

```
...
self.mxnet_ctx = mx.neuron() if gpu_id is None else mx.gpu(gpu_id)
...
```

```
#model_input = [item.as_in_context(self.mxnet_ctx) for item in model_input]
```

Empaquete el modelo con model-archiver utilizando los siguientes comandos:

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

Ejecutar inferencia

Inicie el servidor multimodelo y cargue el modelo que lo utiliza RESTful API mediante los siguientes comandos. Asegúrese de que neuron-rtd se está ejecutando con la configuración predeterminada.

```
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
```

Ejecute la inferencia utilizando una imagen de ejemplo con los siguientes comandos:

```
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
```

El resultado debería tener el siguiente aspecto:

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

Para limpiar después de la prueba, ejecute un comando de eliminación mediante el servidor de modelos RESTful API y detenga el servidor de modelos mediante los siguientes comandos:

```
curl -X DELETE http://127.0.0.1:8081/models/resnet-50_compiled

multi-model-server --stop
```

Debería ver los siguientes datos de salida:

```
{
  "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
```

Usando PyTorch -Neuron y el AWS Compilador de neuronas

La compilación PyTorch -Neuron API proporciona un método para compilar un gráfico modelo que se puede ejecutar en un AWS Dispositivo de inferencia.

Un modelo entrenado debe compilarse en un destino de Inferentia antes de poder implementarlo en instancias Inf1. El siguiente tutorial compila el modelo torchvision ResNet 50 y lo exporta como un módulo guardado. TorchScript A continuación, el modelo se utiliza para ejecutar la inferencia.

Para mayor comodidad, el tutorial utiliza una instancia Inf1 tanto para la compilación como para la inferencia. En la práctica, puede compilar el modelo con otro tipo de instancia, como la familia de instancias c5. A continuación, debe implementar el modelo compilado en el servidor de inferencia Inf1. Para obtener más información, consulte la .AWS Documentación sobre neuronas. PyTorch SDK

Contenido

Requisitos previos

Antes de utilizar este tutorial, debería haber completado los pasos de configuración de Lanzar una instancia con DLAMI AWS Neuron. También debe estar familiarizado con el aprendizaje profundo y el uso del. DLAMI

Activación del entorno Conda

Active el entorno conda PyTorch -Neuron mediante el siguiente comando:

```
source activate aws_neuron_pytorch_p36
```

Para salir del entorno Conda actual, ejecute:

```
source deactivate
```

Compilación de Resnet50

Cree un script de Python denominado **`pytorch_trace_resnet50.py`** con el siguiente contenido. Este script usa la compilación PyTorch -Neuron de Python API para compilar un modelo ResNet -50.

> **Note**
>
> Hay una dependencia entre las versiones de torchvision y el paquete de torch que debe tener en cuenta al compilar los modelos de torchvision. Estas reglas de dependencia se pueden gestionar a través de pip. Torchvision==0.6.1 coincide con la versión torch==1.5.1, mientras que torchvision==0.8.2 coincide con la versión 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")
```

Ejecute el script de compilación.

```
python pytorch_trace_resnet50.py
```

La compilación tardará unos minutos. Cuando haya finalizado, el modelo compilado se guardará como `resnet50_neuron.pt` en el directorio local.

ResNetInferencia 5.0

Cree un script de Python denominado **`pytorch_infer_resnet50.py`** con el siguiente contenido. Este script descarga una imagen de muestra y la utiliza para ejecutar la inferencia con el modelo compilado.

```
import os
import time
import torch
import torch_neuron
import json
import numpy as np

from urllib import request

from torchvision import models, transforms, datasets

## Create an image directory containing a small kitten
os.makedirs("./torch_neuron_test/images", exist_ok=True)
request.urlretrieve("https://raw.githubusercontent.com/awslabs/mxnet-model-server/
master/docs/images/kitten_small.jpg",
```

```
                    "./torch_neuron_test/images/kitten_small.jpg")


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

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

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

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

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

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

## Predict
results = model_neuron( image )

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

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

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

Ejecute la inferencia con el modelo compilado mediante el siguiente comando:

```
python pytorch_infer_resnet50.py
```

El resultado debería tener el siguiente aspecto:

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

La ARM64 DLAMI

AWS ARM64GPUDLAMIsestán diseñados para proporcionar un alto rendimiento y rentabilidad para las cargas de trabajo de aprendizaje profundo. En concreto, el tipo de instancia G5g incluye una instancia basada en ARM64 AWS El procesador Graviton2, que fue creado desde cero por AWS y optimizado para la forma en que los clientes ejecutan sus cargas de trabajo en la nube. AWS ARM64GPUDLAMIsestán preconfigurados con Docker, NVIDIA Docker, NVIDIA Driver, Cu CUDADNN,, así como con marcos de aprendizaje automático popularesNCCL, como y. TensorFlow PyTorch

Con el tipo de instancia G5G, puede aprovechar las ventajas de precio y rendimiento de Graviton2 para implementar modelos de aprendizaje profundo GPU acelerados a un costo significativamente menor en comparación con las instancias basadas en x86 con aceleración. GPU

Seleccione un ARM64 DLAMI

Lanza una instancia G5G con ARM64 DLAMI la que prefieras.

Para step-by-step obtener instrucciones sobre cómo lanzar unDLAMI, consulte Lanzamiento y configuración de un DLAMI.

Para obtener una lista de las más recientes ARM64DLAMIs, consulte las notas de la versión de DLAMI.

Introducción

Los siguientes temas le muestran cómo empezar a utilizar el ARM64DLAMI.

Contenido

Uso del ARM64 GPU PyTorch DLAMI

La AWS Deep Learning AMIs está listo para usarse con el procesador Arm64 y viene GPUs optimizado para. PyTorch ARM64GPU PyTorch DLAMIIncluye un entorno Python preconfigurado con y TorchServepara PyTorchcasos TorchVisionde uso de inferencia y entrenamiento de aprendizaje profundo.

Contenido

Verificar el entorno de PyTorch Python

Conéctese a su instancia de G5g y active el entorno base de Conda con el siguiente comando:

```
source activate base
```

La línea de comandos debe indicar que está trabajando en el entorno base de Conda, que contiene PyTorch TorchVision, y otras bibliotecas.

```
(base) $
```

Compruebe las rutas de herramientas predeterminadas del PyTorch entorno:

```
(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)
```

Ejecute un ejemplo de entrenamiento con PyTorch

Ejecute un ejemplo de trabajo de MNIST formación:

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

El resultado debería tener un aspecto similar al siguiente:

```
...
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%)
```

Ejecute un ejemplo de inferencia con PyTorch

Utilice los siguientes comandos para descargar un modelo densenet161 previamente entrenado y ejecutar la inferencia mediante: 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
```

El resultado debería tener un aspecto similar al siguiente:

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

Utilice los siguientes comandos para anular el registro del modelo densenet161 y detener el servidor:

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

El resultado debería tener un aspecto similar al siguiente:

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

Inferencia

En esta sección se proporcionan tutoriales sobre cómo ejecutar inferencias utilizando los marcos y las herramientas DLAMI de la aplicación.

Herramientas de inferencia

- TensorFlow Sirviendo

Distribución de modelos

Las siguientes son las opciones de servicio de modelos instaladas en el Deep Learning AMI con Conda. Haga clic en una de las opciones para obtener información acerca de cómo utilizarla.

Temas

- TensorFlow Sirviendo
- TorchServe

TensorFlow Sirviendo

TensorFlow Serving es un sistema de servicio flexible y de alto rendimiento para modelos de aprendizaje automático.

¡`tensorflow-serving-api`Viene preinstalado con Deep Learning AMI con Conda! Encontrará un ejemplo de scripts para entrenar, exportar y servir un MNIST modelo. `~/examples/tensorflow-serving/`

Para ejecutar cualquiera de estos ejemplos, primero conéctese a su Deep Learning AMI con Conda y active el TensorFlow entorno.

```
$ source activate tensorflow2_p310
```

Ahora, desplácese a los directorios que contienen la carpeta de scripts de ejemplo.

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

Cómo servir un modelo de incepción

A continuación, se muestra un ejemplo que puede probar para servir distintos modelos como Inception. Como regla general, necesitará un modelo que funcione y que los scripts de cliente estén ya descargados en su servidor. DLAMI

Cómo servir y probar la inferencia con un modelo de incepción

1. Descargue el modelo.

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

2. Descomprima el modelo.

```
$ unzip INCEPTION.zip
```

3. Descargue una imagen de un perro esquimal.

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

4. Lance el servidor. Tenga en cuenta que, para Amazon Linux, debe cambiar el directorio que se utiliza para `model_base_path` de `/home/ubuntu` a `/home/ec2-user`.

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

5. Con el servidor ejecutándose en primer plano, tendrá que lanzar otra sesión de terminal para continuar. Abre una nueva terminal y actívala TensorFlow con`source activate tensorflow2_p310`. A continuación, utilice su editor de texto preferido para crear un script que tenga el siguiente contenido. Denomínelo `inception_client.py`. Este script tomará un nombre de archivo de imagen como parámetro y obtendrá un resultado de predicción a partir del modelo entrenado previamente.

```
from __future__ import print_function

import grpc
import tensorflow as tf
import argparse

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

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


def main():
```

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

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

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

  print("Inception Client Passed")


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

6. Ahora ejecute el script pasando la ubicación y el puerto del servidor y el nombre de archivo de la foto del perro esquimal como parámetros.

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

Entrena y sirve a un MNIST modelo

En este tutorial vamos a exportar un modelo y después lo distribuiremos con la aplicación `tensorflow_model_server`. Por último, puede probar el servidor del modelo con un script del cliente de ejemplo.

Ejecute el script que entrenará y exportará un MNIST modelo. Como único argumento del script debe proporcionar una ubicación de carpeta para guardar el modelo. Por ahora, lo pondremos en `mnist_model`. El script creará la carpeta por usted.

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

Sea paciente, ya que el script puede tardar un rato en proporcionar resultados. Cuando se haya completado el entrenamiento y se haya exportado el modelo, debería ver lo siguiente:

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

El siguiente paso consiste en ejecutar `tensorflow_model_server` para distribuir el modelo exportado.

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

Se proporciona un script del cliente para probar el servidor.

Para probarlo, tendrá que abrir una nueva ventana de la terminal.

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

Más características y ejemplos

Si está interesado en obtener más información sobre TensorFlow Serving, visite el TensorFlow sitio web.

También puede usar TensorFlow Serving with Amazon Elastic Inference. Consulta la guía sobre cómo usar Elastic Inference with TensorFlow Serving para obtener más información.

TorchServe

TorchServe es una herramienta flexible para ofrecer modelos de aprendizaje profundo que se han exportado desde PyTorch. TorchServe viene preinstalado con el Deep Learning AMI con Conda.

Para obtener más información sobre el uso TorchServe, consulte Model Server para PyTorch ver la documentación.

Temas

Sirva un modelo de clasificación de imágenes en TorchServe

Este tutorial muestra cómo crear un modelo de clasificación de imágenes con TorchServe. Utiliza un modelo DenseNet -161 proporcionado por PyTorch. Una vez que el servidor está en funcionamiento,

escucha las solicitudes de predicción. Al cargar una imagen, en este caso una imagen de un gatito, el servidor devuelve una predicción de las 5 principales clases coincidentes de entre las clases con las que se haya entrenado el modelo.

A modo de ejemplo, un modelo de clasificación de imágenes en TorchServe

1. Conéctese a una instancia de Amazon Elastic Compute Cloud (AmazonEC2) con Deep Learning AMI con Conda v34 o posterior.
2. Active el entorno de `pytorch_p310`.

```
source activate pytorch_p310
```

3. Clone el TorchServe repositorio y, a continuación, cree un directorio para almacenar sus modelos.

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

4. Archive el modelo utilizando el archivador de modelos. El `extra-files` parámetro usa un archivo del `TorchServe` repositorio, así que actualiza la ruta si es necesario. Para obtener más información sobre el archivador de modelos, consulte el archivador de modelos Torch para. 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. Ejecute TorchServe para iniciar un punto final. Al agregar `> /dev/null` se silencia la salida del registro.

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

6. Descarga una imagen de un gatito y envíala al punto final de TorchServe predicción:

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

El punto final de predicción devuelve una predicción JSON similar a las cinco predicciones principales siguientes, en las que la imagen tiene un 47% de probabilidades de contener un gato egipcio, seguido de un 46% de probabilidades de que tenga un gato atigrado.

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

7. Cuando termine la prueba, detenga el servidor.

```
torchserve --stop
```

Otros ejemplos

TorchServe tiene varios ejemplos que puede ejecutar en su DLAMI instancia. Puedes verlos en la página de ejemplos TorchServe del repositorio del proyecto.

Más información

Para obtener más TorchServe documentación, incluida la forma de configurar TorchServe con Docker y las TorchServe funciones más recientes, consulta la página del TorchServe proyecto en GitHub.

Actualización de la DLAMI

Aquí encontrará información sobre la actualización de la DLAMI y consejos sobre la actualización de software en ella.

Temas

Actualización a una nueva versión de la DLAMI

Las imágenes del sistema de la DLAMI se actualizan de forma periódica para aprovechar las nuevas versiones del marco de aprendizaje profundo, CUDA y otras actualizaciones de software, así como para el ajuste del desempeño. Si lleva un tiempo utilizando una DLAMI y desea aprovechar una actualización, tendrá que volver a lanzar una nueva instancia. Además, tendría que transferir manualmente cualquier conjunto de datos, puntos de comprobación u otros datos valiosos. En lugar de ello, puede utilizar Amazon EBS para retener los datos y asociarlos a una nueva DLAMI. De esta forma, puede actualizar a menudo, además de reducir el tiempo que se tarda en realizar la transición de los datos.

> **Note**
>
> Al adjuntar y desplazar volúmenes de Amazon EBS entre las DLAMI, debe tener tanto las DLAMI como el nuevo volumen en la misma zona de disponibilidad.

1. Utilice la consola de Amazon EC2 para crear un nuevo volumen de Amazon EBS. Para obtener instrucciones detalladas, consulte Creación de un volumen de Amazon EBS.
2. Adjunte el volumen de Amazon EB; recién creado a la DLAMI existente. Para obtener instrucciones detalladas, consulte Attaching an Amazon EBS Volume.
3. Transfiera sus datos, por ejemplo, conjuntos de datos, puntos de comprobación y archivos de configuración.
4. Lanzamiento de una DLAMI. Para obtener indicaciones detalladas, consulte Lanzamiento y configuración de una DLAMI.

5. Separe el volumen de Amazon EBS de la DLAMI antigua. Para obtener instrucciones detalladas, consulte Detaching an Amazon EBS Volume.
6. Adjunte el volumen de Amazon EBS a la nueva DLAMI. Siga las instrucciones desde el Paso 2 para adjuntar el volumen.
7. Después de verificar que los datos están disponibles en su nueva DLAMI, detenga y termine la DLAMI antigua. Para obtener instrucciones de limpieza detalladas, consulte Eliminación.

Sugerencias para actualizaciones de software

De vez en cuando, es posible que desee actualizar manualmente el software en la DLAMI. En general, se recomienda que utilice `pip` para actualizar paquetes de Python. También debería utilizar `pip` para actualizar paquetes dentro de un entorno de Conda en la AMI de aprendizaje profundo con Conda. Consulte el sitio web del marco de trabajo o del software para obtener las instrucciones de actualización y de instalación.

> **Note**
>
> No podemos garantizar que la actualización de un paquete se realice correctamente. Si se intenta actualizar un paquete en un entorno con dependencias incompatibles, se puede producir un error. En tal caso, debe ponerse en contacto con el responsable de la biblioteca para ver si es posible actualizar las dependencias del paquete. Como alternativa, puede intentar modificar el entorno de tal manera que permita la actualización. Sin embargo, es probable que esta modificación implique eliminar o actualizar los paquetes existentes, lo que significa que ya no podemos garantizar la estabilidad de este entorno.

AWS Deep Learning AMIs Viene con muchos entornos Conda y muchos paquetes preinstalados. Debido a la cantidad de paquetes preinstalados, es difícil encontrar un conjunto de paquetes que garanticen su compatibilidad. Es posible que aparezca una advertencia que diga: “The environment is inconsistent, please check the package plan carefully”. La DLAMI garantiza que todos los entornos proporcionados por ella sean correctos, pero no puede garantizar que los paquetes instalados por el usuario funcionen correctamente.

Reciba notificaciones sobre nuevas actualizaciones

> **Note**
>
> AWS Las AMI de aprendizaje profundo publican parches de seguridad cada semana. Se enviarán notificaciones de publicación de estos parches de seguridad incrementales, aunque es posible que no se incluyan en las notas oficiales de la versión.

Puede recibir notificaciones cada vez que se publique una nueva DLAMI. Las notificaciones se publican con Amazon SNS mediante el tema siguiente.

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

Los mensajes se publican aquí cuando se publica un nuevo DLAMI. La versión, los metadatos y los ID de AMI regionales de la AMI se incluirán en el mensaje.

Estos mensajes se pueden recibir mediante varios métodos diferentes. Le recomendamos que utilice el siguiente método.

1. Abra la consola de Amazon SNS.
2. En la barra de navegación, cambie la AWS región a EE.UU. Oeste (Oregón), si es necesario. Debes seleccionar la región en la que se creó la notificación de SNS a la que te estás suscribiendo.
3. En el panel de navegación, selecciona Suscripciones y Crear suscripción.
4. En el cuadro de diálogo Crear suscripción, haga lo siguiente:
    a. Para el ARN del tema, copie y pegue el siguiente nombre de recurso de Amazon (ARN): **`arn:aws:sns:us-west-2:767397762724:dlami-updates`**
    b. Para Protocol, elija uno de [Amazon SQS, AWS Lamda, Email, Email-JSON]
    c. Para Endpoint, introduzca la dirección de correo electrónico o el nombre de recurso de Amazon (ARN) del recurso que utilizará para recibir las notificaciones.
    d. Elija Crear una suscripción.
5. Recibirá un correo electrónico de confirmación con el asunto AWS Notificación: confirmación de suscripción. Abra el correo electrónico y elija Confirmar suscripción para completar la suscripción.

Seguridad en AWS Deep Learning AMIs

Seguridad en la nube en AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS y tú. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que se ejecuta AWS servicios en el AWS Nube. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte del AWS Programas de cumplimiento . Para obtener más información sobre los programas de cumplimiento que se aplican aDLAMI, consulte AWS Servicios incluidos en el ámbito de aplicación del programa de cumplimiento .
- Seguridad en la nube: su responsabilidad viene determinada por la AWS servicio que utiliza. Usted también es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utilizaDLAMI. Los siguientes temas muestran cómo configurarlo DLAMI para cumplir sus objetivos de seguridad y conformidad. También aprenderá a usar otros AWS servicios que le ayudan a supervisar y proteger sus DLAMI recursos.

Para obtener más información, consulta Seguridad en Amazon EC2.

Temas

Protección de datos en AWS Deep Learning AMIs

La AWS modelo de responsabilidad compartida modelo de se aplica a la protección de datos en AWS Aprendizaje profundoAMI. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los Nube de AWS. Usted es responsable de mantener el control sobre el contenido que está alojado en esta infraestructura. También es responsable de las tareas de configuración y administración de la seguridad del Servicios de AWS que utilices. Para obtener más información sobre la privacidad de los datos, consulte la sección Privacidad de datos FAQ. Para obtener información sobre la protección de datos en Europa, consulte la AWS Modelo de responsabilidad compartida y entrada de GDPR blog sobre AWS Blog de seguridad.

Para fines de protección de datos, le recomendamos que proteja Cuenta de AWS credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice la autenticación multifactorial (MFA) con cada cuenta.
- UtiliceSSL/TLSpara comunicarse con AWS recursos. Necesitamos TLS 1.2 y recomendamos TLS 1.3.
- Configure API y registre la actividad del usuario con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Trabajar con CloudTrail senderos en la AWS CloudTrail Guía del usuario.
- Uso AWS soluciones de cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita entre FIPS 140 y 3 módulos criptográficos validados para acceder AWS a través de una interfaz de línea de comandos oAPI, utilice un FIPS punto final. Para obtener más información sobre los FIPS puntos finales disponibles, consulte la Norma Federal de Procesamiento de Información (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con u otros DLAMI Servicios de AWS

utilizando la consolaAPI, AWS CLI, o AWS SDKs. Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, le recomendamos encarecidamente que no incluya información sobre las credenciales URL para validar su solicitud a ese servidor.

Identity and Access Management en AWS Deep Learning AMIs

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS recursos. IAMlos administradores controlan quién puede autenticarse (iniciar sesión) y quién está autorizado (tiene permisos) para usar DLAMI los recursos. IAMes un Servicio de AWS que puede utilizar sin coste adicional.

Para obtener más información sobre Identity and Access Management, consulte Identity and Access Management for Amazon EC2.

Temas

- Autenticación con identidades
- Administración de acceso mediante políticas
- IAMcon Amazon EMR

Autenticación con identidades

La autenticación es la forma de iniciar sesión en AWS utilizando tus credenciales de identidad. Debe estar autenticado (iniciar sesión en AWS) como Usuario raíz de la cuenta de AWS, como IAM usuario o asumiendo un IAM rol.

Puede iniciar sesión en AWS como identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios de (IAMIdentity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, el administrador configuró previamente la federación de identidades mediante roles. IAM Cuando accedes AWS al usar la federación, está asumiendo indirectamente un rol.

Según el tipo de usuario que sea, puede iniciar sesión en el AWS Management Console o el AWS portal de acceso. Para obtener más información sobre cómo iniciar sesión en AWS, consulta Cómo iniciar sesión en tu Cuenta de AWS en la AWS Sign-In Guía del usuario.

Si accedes AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no usa AWS herramientas, debe firmar las solicitudes usted mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar AWS APIsolicitudes en la Guía IAM del usuario.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo: AWS recomienda que utilice la autenticación multifactorial (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactorial en la AWS IAM Identity Center Guía del usuario y Uso de la autenticación multifactorial () MFA en AWS en la Guía del usuario de IAM.

Cuenta de AWS usuario raíz

Al crear un Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todas Servicios de AWS y los recursos de la cuenta. Esta identidad se denomina Cuenta de AWS usuario root y se accede a él iniciando sesión con la dirección de correo electrónico y la contraseña que utilizó para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de tareas que requieren que inicie sesión como usuario root, consulte Tareas que requieren credenciales de usuario root en la Guía del IAM usuario.

Usuarios y grupos de IAM

Un IAMusuario es una identidad dentro de su Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos utilizar credenciales temporales en lugar de crear IAM usuarios con credenciales de larga duración, como contraseñas y claves de acceso. Sin embargo, si tiene casos de uso específicos que requieren credenciales a largo plazo con IAM los usuarios, le recomendamos que rote las claves de acceso. Para obtener más información, consulte Rotar las claves de acceso con regularidad para los casos de uso que requieran credenciales de larga duración en la Guía del IAM usuario.

Un IAMgrupo es una identidad que especifica un conjunto de IAM usuarios. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar IAM los recursos.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte Cuándo crear un IAM usuario (en lugar de un rol) en la Guía del IAM usuario.

IAMFunciones

Un IAMrol es una identidad dentro de tu Cuenta de AWS que tiene permisos específicos. Es similar a un IAM usuario, pero no está asociado a una persona específica. Puede asumir temporalmente un IAM rol en el AWS Management Console cambiando de rol. Puede asumir un rol llamando a un AWS CLI o AWS APIoperación o mediante una operación personalizadaURL. Para obtener más información sobre los métodos de uso de roles, consulte Uso de IAM roles en la Guía del IAM usuario.

IAMlos roles con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información sobre los roles para la federación, consulte Creación de un rol para un proveedor de identidad externo en la Guía del IAM usuario. Si usa IAM Identity Center, configura un conjunto de permisos. Para controlar a qué pueden acceder sus identidades después de autenticarse, IAM Identity Center correlaciona el conjunto de permisos con un rol en. IAM Para obtener información sobre los conjuntos de permisos, consulte los conjuntos de permisos en la AWS IAM Identity Center Guía del usuario.
- Permisos de IAM usuario temporales: un IAM usuario o rol puede asumir un IAM rol para asumir temporalmente diferentes permisos para una tarea específica.
- Acceso multicuenta: puedes usar un IAM rol para permitir que alguien (un responsable de confianza) de una cuenta diferente acceda a los recursos de tu cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunos Servicios de AWS, puede adjuntar una política directamente a un recurso (en lugar de utilizar un rol como proxy). Para saber la diferencia entre las funciones y las políticas basadas en recursos para el acceso multicuenta, consulta el tema sobre el acceso a los recursos entre cuentas IAM en la Guía del IAM usuario.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio

haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.

- Sesiones de acceso directo (FAS): cuando utilizas un IAM usuario o un rol para realizar acciones en AWS, se le considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FASutiliza los permisos del director que llama a un Servicio de AWS, combinado con la solicitud Servicio de AWS para realizar solicitudes a los servicios intermedios. FASlas solicitudes solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS o recursos para completar. En este caso, debe tener permisos para realizar ambas acciones. Para obtener detalles sobre la política a la hora de realizar FAS solicitudes, consulte Reenviar las sesiones de acceso.
- Función de servicio: una función de servicio es una IAMfunción que un servicio asume para realizar acciones en su nombre. Un IAM administrador puede crear, modificar y eliminar un rol de servicio desde dentroIAM. Para obtener más información, consulte Crear un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
- Función vinculada a un servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un Servicio de AWS. El servicio puede asumir la función de realizar una acción en su nombre. Los roles vinculados al servicio aparecen en su Cuenta de AWS y son propiedad del servicio. Un IAM administrador puede ver, pero no editar, los permisos de las funciones vinculadas al servicio.

- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un IAM rol para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y se están creando AWS CLI o AWS APIsolicitudes. Esto es preferible a almacenar las claves de acceso dentro de la EC2 instancia. Para asignar un AWS Un rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia que se adjunte a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte Uso de un IAM rol para conceder permisos a aplicaciones que se ejecutan en EC2 instancias de Amazon en la Guía del IAM usuario.

Para saber si se deben usar IAM roles o IAM usuarios, consulte Cuándo crear un IAM rol (en lugar de un usuario) en la Guía del IAM usuario.

Administración de acceso mediante políticas

Usted controla el acceso en AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto en AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan en AWS como JSON documentos. Para obtener más información sobre la estructura y el contenido de los documentos de JSON políticas, consulte Descripción general de JSON las políticas en la Guía del IAM usuario.

Los administradores pueden utilizar AWS JSONpolíticas para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Para conceder a los usuarios permiso para realizar acciones en los recursos que necesitan, un IAM administrador puede crear IAM políticas. A continuación, el administrador puede añadir las IAM políticas a las funciones y los usuarios pueden asumir las funciones.

IAMlas políticas definen los permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre su función en AWS Management Console, el AWS CLI, o el AWS API.

Políticas basadas en identidad

Las políticas basadas en la identidad son documentos de política de JSON permisos que se pueden adjuntar a una identidad, como un IAM usuario, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener información sobre cómo crear una política basada en la identidad, consulte Creación de IAM políticas en la Guía del usuario. IAM

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles en su Cuenta de AWS. Las políticas gestionadas incluyen AWS las políticas gestionadas y las políticas gestionadas por el cliente. Para saber cómo elegir entre una política gestionada o una política en línea, consulte Elegir entre políticas gestionadas y políticas integradas en la Guía del IAMusuario.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de JSON política que se adjuntan a un recurso. Algunos ejemplos de políticas basadas en recursos son las políticas de confianza de IAM roles y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o Servicios de AWS.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar AWS políticas gestionadas desde una política basada IAM en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué directores (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLsson similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de JSON políticas.

Amazon S3, AWS WAF, y Amazon VPC son ejemplos de servicios que admitenACLs. Para obtener más informaciónACLs, consulte la descripción general de la lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una función avanzada en la que se establecen los permisos máximos que una política basada en la identidad puede conceder a una IAM entidad (IAMusuario o rol). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte los límites de los permisos para IAM las entidades en la Guía del IAMusuario.

- Políticas de control de servicios (SCPs): SCPs son JSON políticas que especifican los permisos máximos para una organización o unidad organizativa (OU) en AWS Organizations. AWS Organizations es un servicio para agrupar y administrar de forma centralizada múltiples Cuentas de AWS que es propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar las políticas de control de servicios (SCPs) a cualquiera de tus cuentas o a todas ellas. SCPLimita los permisos de las entidades en las cuentas de los miembros, incluidas todas Usuario raíz de la cuenta de AWS. Para obtener más información acerca de OrganizationsSCPs, consulte Políticas de control de servicios en AWS Organizations Guía del usuario.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información, consulte las políticas de sesión en la Guía del IAM usuario.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determina si se permite una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del IAM usuario.

IAMcon Amazon EMR

Puede usar… AWS Identity and Access Management con Amazon EMR para definir a los usuarios, AWS recursos, grupos, funciones y políticas. También puede controlar cuáles AWS servicios a los que pueden acceder estos usuarios y roles.

Para obtener más información sobre el uso IAM con AmazonEMR, consulta AWS Identity and Access Management para Amazon EMR.

Inicio de sesión y supervisión AWS Deep Learning AMIs

Sus AWS Deep Learning AMIs instance viene con varias herramientas GPU de monitoreo, incluida una utilidad que informa las estadísticas de GPU uso a Amazon CloudWatch. Para obtener más información, consulte GPUMonitoring and Optimization y Monitoring Amazon EC2.

Seguimiento de uso

Los siguientes ejemplos de AWS Deep Learning AMIs Las distribuciones de los sistemas operativos incluyen un código que permite AWS para recopilar información sobre el tipo de instancia, el ID de la instancia, el DLAMI tipo y el sistema operativo. No se recopila ni conserva información sobre los comandos utilizados en elDLAMI. No se recopila ni conserva ninguna otra información sobre el. DLAMI

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

Para excluirte del seguimiento del usoDLAMI, añade una etiqueta a tu EC2 instancia de Amazon durante el lanzamiento. La etiqueta debe usar la clave `OPT_OUT_TRACKING` con el valor asociado definido como `true`. Para obtener más información, consulta Cómo etiquetar tus EC2 recursos de Amazon.

Validación de conformidad para AWS Deep Learning AMIs

Auditores externos evalúan la seguridad y el cumplimiento de AWS Deep Learning AMIs como parte de múltiples AWS programas de cumplimiento. Para obtener información sobre los programas de conformidad compatibles, consulta Validación de conformidad para Amazon EC2.

Para obtener una lista de AWS servicios incluidos en el ámbito de programas de cumplimiento específicos, consulte AWS Servicios incluidos en el ámbito de aplicación del programa de cumplimiento . Para obtener información general, consulte AWS Programas de cumplimiento .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Informes de .

Su responsabilidad de cumplimiento al DLAMI utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Guías de inicio rápido sobre sobre seguridad y cumplimiento: estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en la seguridad y el cumplimiento en AWS.

- AWS Recursos de conformidad : esta colección de libros de trabajo y guías puede aplicarse a su sector y ubicación.
- Evaluación de los recursos con las reglas del AWS Config Guía para desarrolladores: la AWS Config El servicio evalúa en qué medida las configuraciones de sus recursos cumplen con las prácticas internas, las pautas y las regulaciones del sector.
- AWS Security Hub— Esto AWS El servicio proporciona una visión completa del estado de su seguridad interior AWS que le ayuda a comprobar su conformidad con los estándares y las mejores prácticas del sector de la seguridad.

Resiliencia en AWS Deep Learning AMIs

La AWS la infraestructura global se basa en AWS Regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información acerca de AWS Regiones y zonas de disponibilidad, consulte AWS Infraestructura global.

Para obtener información sobre las funciones que ayudan a respaldar sus necesidades de respaldo y resiliencia de datos, consulte Resiliencia en Amazon EC2.

Seguridad de la infraestructura en AWS Deep Learning AMIs

La seguridad de la infraestructura de AWS Deep Learning AMIs cuenta con el respaldo de AmazonEC2. Para obtener más información, consulte Seguridad de infraestructuras en Amazon EC2.

DLAMIpolítica marco de apoyo

Aquí puede encontrar detalles de la política de soporte para AWS Deep Learning AMIs (DLAMI) marcos.

Para obtener una lista de los DLAMI marcos que AWS actualmente es compatible, consulte la página de políticas de soporte de DLAMI Framework. En las tablas de esa página, tenga en cuenta lo siguiente:

- La versión actual especifica la versión del marco en el formato x.y.z. En este formato, x se refiere a la versión principal, y se refiere a la versión secundaria y z se refiere a la versión del parche. Por ejemplo, en la versión TensorFlow 2.10.1, la versión principal es 2, la versión secundaria es 10 y la versión del parche es 1.
- El final del parche especifica cuánto dura AWS es compatible con esa versión del marco.

Para obtener información detallada sobre temas específicosDLAMIs, consulteNotas de la versión de DLAMIs.

DLAMIsoporte marco FAQs

- ¿Qué versiones de marcos incluyen parches de seguridad?
- ¿Qué hacen las imágenes AWS ¿publicar cuando se publiquen nuevas versiones del framework?
- Qué imágenes se vuelven nuevas/ SageMaker AWS características?
- ¿Cómo se define la versión actual en la tabla de marcos compatibles?
- ¿Qué sucede si estoy ejecutando una versión que no está en la tabla de marcos compatibles?
- ¿Son DLAMIs compatibles con las versiones anteriores de TensorFlow?
- ¿Cómo puedo encontrar la última imagen parcheada de una versión de marco compatible?
- ¿Con qué frecuencia se publican nuevas imágenes?
- ¿Se instalará el parcheo de mi instancia mientras se ejecute mi carga de trabajo?
- ¿Qué ocurre cuando hay disponible una nueva versión del marco parcheada o actualizada?
- ¿Se actualizan las dependencias sin cambiar la versión del marco?
- ¿Cuándo finaliza el soporte activo para mi versión de marco?

- ¿Se parchearán las imágenes con versiones de marco que ya no se mantienen activamente?
- ¿Cómo utilizo una versión anterior de marco?
- ¿Cómo puedo cumplir up-to-date con los cambios de soporte en los marcos y sus versiones?
- ¿Necesito una licencia comercial para usar el repositorio de Anaconda?

¿Qué versiones de marcos incluyen parches de seguridad?

Si la versión del marco está etiquetada como compatible en la AWS Deep Learning AMIs Tabla de políticas de soporte de Framework, recibe parches de seguridad.

¿Qué hacen las imágenes AWS ¿publicar cuando se publiquen nuevas versiones del framework?

Publicamos las nuevas versiones DLAMIs poco después de que se publiquen nuevas versiones TensorFlow y PyTorch se publiquen. Esto incluye las versiones principales, las versiones principales y secundarias y las major-minor-patch versiones de los marcos. También actualizamos las imágenes cuando hay nuevas versiones de controladores y bibliotecas disponibles. Para obtener más información sobre el mantenimiento de imágenes, consulte ¿Cuándo finaliza el soporte activo para mi versión de marco?.

Qué imágenes se vuelven nuevas/ SageMaker AWS características?

Las nuevas funciones suelen publicarse en la última versión de DLAMIs for PyTorch y TensorFlow. Consulte las notas de la versión para ver una imagen específica para obtener más información sobre las novedades SageMaker o AWS características. Para obtener una lista de las disponiblesDLAMIs, consulte las notas de la versión de DLAMI. Para obtener más información sobre el mantenimiento de imágenes, consulte ¿Cuándo finaliza el soporte activo para mi versión de marco?.

¿Cómo se define la versión actual en la tabla de marcos compatibles?

La versión actual de AWS Deep Learning AMIs La tabla Framework Support Policy se refiere a la versión más reciente del marco que AWS está disponible en GitHub. Cada versión más reciente incluye actualizaciones de los controladores, las bibliotecas y los paquetes relevantes deDLAMI. Para obtener más información sobre el mantenimiento de imágenes, consulte ¿Cuándo finaliza el soporte activo para mi versión de marco?.

¿Qué sucede si estoy ejecutando una versión que no está en la tabla de marcos compatibles?

Si está ejecutando una versión que no está en AWS Deep Learning AMIs En la tabla de políticas de soporte de Framework, es posible que no tenga los controladores, bibliotecas y paquetes relevantes más actualizados. Para obtener una up-to-date versión posterior, le recomendamos que actualice a uno de los marcos compatibles disponibles utilizando la última versión DLAMI de su elección. Para obtener una lista de los disponiblesDLAMIs, consulte las notas de la versión de DLAMI.

¿Son DLAMIs compatibles con las versiones anteriores de TensorFlow?

¿No?. Admitimos la última versión de parche de la última versión principal de cada framework publicada 365 días después de su GitHub lanzamiento inicial, tal y como se indica en el AWS Deep Learning AMIs Tabla de políticas de soporte de Framework. Para obtener más información, consulte ¿Qué sucede si estoy ejecutando una versión que no está en la tabla de marcos compatibles?

¿Cómo puedo encontrar la última imagen parcheada de una versión de marco compatible?

Para utilizar una DLAMI con la última versión del marco, recupere el DLAMIID y utilícelo para iniciarlo DLAMI mediante la EC2consola. Para ver un ejemplo AWS CLIcomandos para recuperar el AWS Deep Learning AMIs ID, consulte la página de notas de la DLAMI versión, las notas de la DLAMIversión de un solo marco. La versión del marco que elija debe estar etiquetada como compatible en el AWS Deep Learning AMIs Tabla de políticas de soporte de Framework.

¿Con qué frecuencia se publican nuevas imágenes?

Proporcionar versiones de parches actualizadas es nuestra máxima prioridad. Creamos imágenes parcheadas de forma rutinaria lo antes posible. Supervisamos las nuevas versiones del framework parcheadas (p. ej. TensorFlow 2.9 a TensorFlow 2.9.1) y nuevas versiones secundarias (p. ej. TensorFlow 2.9 a TensorFlow 2.10) y póngalos disponibles lo antes posible. Cuando TensorFlow se publica una versión existente de junto con una nueva versión deCUDA, publicamos una nueva versión DLAMI para esa versión TensorFlow con soporte para la nueva CUDA versión.

¿Se instalará el parcheo de mi instancia mientras se ejecute mi carga de trabajo?

No. Las actualizaciones de parches no DLAMI son actualizaciones «in situ».

Debe activar una EC2 instancia nueva, migrar las cargas de trabajo y los scripts y, a continuación, desactivar la instancia anterior.

¿Qué ocurre cuando hay disponible una nueva versión del marco parcheada o actualizada?

Consulte con regularidad la página de notas de la versión de su imagen. Le recomendamos que actualice a marcos nuevos, parcheados o actualizados cuando estén disponibles. Para ver una lista de las disponiblesDLAMIs, consulte las notas de la versión de DLAMI.

¿Se actualizan las dependencias sin cambiar la versión del marco?

Actualizamos las dependencias sin cambiar la versión del marco. Sin embargo, si una actualización de una dependencia provoca una incompatibilidad, creamos una imagen con una versión diferente. Asegúrese de consultar las notas de la versión DLAMI para obtener información actualizada sobre las dependencias.

¿Cuándo finaliza el soporte activo para mi versión de marco?

DLAMIlas imágenes son inmutables. Una vez creadas, no cambian. Hay cuatro razones principales por las que finaliza el soporte activo para una versión de marco:

- Actualizaciones (parche) de la versión del marco
- AWS parches de seguridad
- Fecha de finalización del parche (fecha de caducidad)
- Dependencia end-of-support

> **Note**
>
> Debido a la frecuencia con que se actualizan los parches de versiones y los parches de seguridad, te recomendamos que consultes la página de notas de la versión DLAMI con frecuencia y que los actualices cuando se produzcan cambios.

Actualizaciones (parche) de la versión del marco

Si tiene una DLAMI carga de trabajo basada en la versión TensorFlow 2.7.0 y TensorFlow publica la versión 2.7.1 a partir GitHub de entonces, AWS lanza una nueva versión DLAMI con TensorFlow

la versión 2.7.1. Las imágenes anteriores de la versión 2.7.0 ya no se mantienen de forma activa una vez que se publica la nueva imagen de la TensorFlow versión 2.7.1. La DLAMI TensorFlow versión 2.7.0 no recibe más parches. A continuación, se actualiza la página de notas de la DLAMI versión TensorFlow 2.7 con la información más reciente. No hay una página de notas de lanzamiento individual para cada parche menor.

Los artículos que se DLAMIs creen debido a una actualización de los parches se designan con un nuevo AMIidentificador.

AWS parches de seguridad

Si tiene una carga de trabajo basada en una imagen con la TensorFlow versión 2.7.0 y AWS crea un parche de seguridad y, a continuación, DLAMI se publica una nueva versión para la versión TensorFlow 2.7.0. La versión anterior de las imágenes, con la versión TensorFlow 2.7.0, ya no se mantiene activamente. Para obtener más información, consulte Si desea conocer los pasos ¿Se instalará el parcheo de mi instancia mientras se ejecute mi carga de trabajo? para encontrar la versión más recienteDLAMI, consulte ¿Cómo puedo encontrar la última imagen parcheada de una versión de marco compatible?

Las nuevas que se DLAMIs creen debido a una actualización de los parches se designan con un nuevo AMIidentificador.

Fecha de finalización del parche (fecha de caducidad)

DLAMIsllegan a la fecha de finalización del parche 365 días después de la fecha GitHub de lanzamiento.

En el caso de varios marcos DLAMIs, cuando se actualiza una de las versiones del marco, se requiere una nueva DLAMI con la versión actualizada. La DLAMI versión del marco anterior ya no se mantiene activamente.

> **Important**
>
> Hacemos una excepción cuando hay una actualización importante del marco. Por ejemplo, si la versión TensorFlow 1.15 se actualiza a la versión TensorFlow 2.0, seguiremos ofreciendo soporte para la versión más reciente de la TensorFlow 1.15 durante un período de dos años a partir de la fecha de GitHub lanzamiento o seis meses después de que el equipo de mantenimiento de Origin Framework deje de ofrecer soporte, la fecha que sea anterior.

Dependencia end-of-support

Si está ejecutando una carga de trabajo en una DLAMI imagen de la versión TensorFlow 2.7.0 con Python 3.6 y esa versión de Python está marcada para end-of-support, todas DLAMI las imágenes basadas en Python 3.6 dejarán de mantenerse activamente. Del mismo modo, si se marca una versión del sistema operativo como Ubuntu 16.04 end-of-support, todas las DLAMI imágenes que dependan de Ubuntu 16.04 dejarán de mantenerse de forma activa.

¿Se parchearán las imágenes con versiones de marco que ya no se mantienen activamente?

No. Las imágenes que ya no se mantengan activamente no tendrán nuevas versiones.

¿Cómo utilizo una versión anterior de marco?

Para usar una versión DLAMI de framework anterior, recupera el DLAMIID y úsalo para iniciarlo desde la DLAMI EC2 consola. En AWS CLIPara recuperar el AMI ID, consulte la página de notas de la versión en las notas de la DLAMIversión de un solo marco.

¿Cómo puedo cumplir up-to-date con los cambios de soporte en los marcos y sus versiones?

Siga up-to-date con DLAMI los marcos y las versiones mediante el AWS Deep Learning AMIs Tabla de políticas de soporte de Framework, notas DLAMI de la versión.

¿Necesito una licencia comercial para usar el repositorio de Anaconda?

Anaconda adoptó un modelo de licencia comercial para ciertos usuarios. Active DLAMIs Maintened se ha migrado a la versión de código abierto de Conda (conda-forge) disponible públicamente desde el canal Anaconda.

Cambios importantes en el NVIDIA controlador DLAMIs

El 15 de noviembre de 2023, AWS realizó cambios importantes en AWS Deep Learning AMIs (DLAMI) relacionados con el NIVIDA controlador que DLAMIs utilice. Para obtener información sobre los cambios y si afectan al uso de los mismosDLAMIs, consulteDLAMINVIDIAcambio de conductor FAQs.

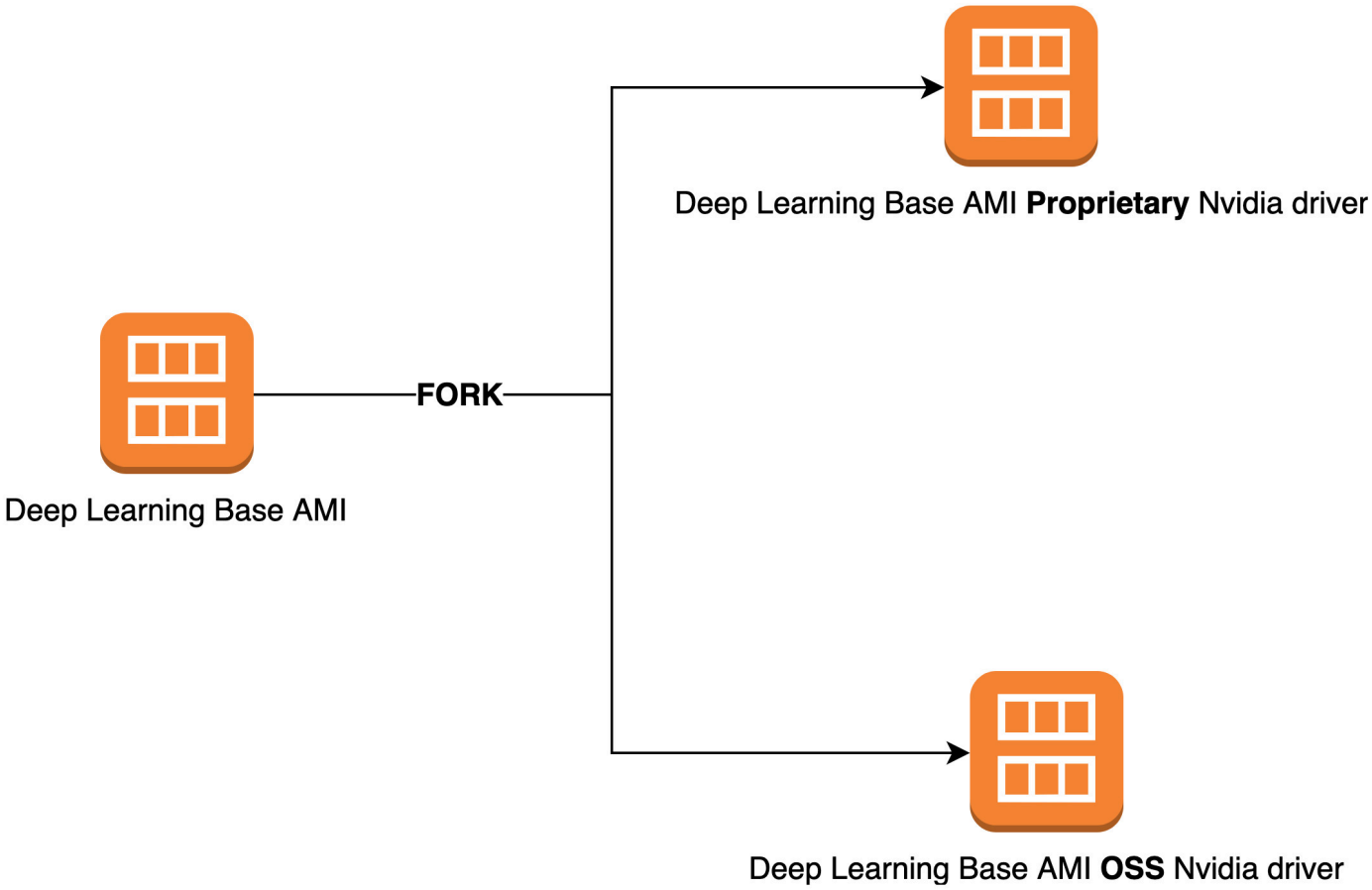
DLAMINVIDIAcambio de conductor FAQs

¿Qué ha cambiado?

Nos DLAMIs dividimos en dos grupos separados:

- DLAMIsque utilizan un controlador NVIDIA propietario (para admitir P3, P3dn, G3)
- DLAMIsque usan el NVIDIA OSS controlador (para admitir G4dn, G5, P4, P5)

Como resultado, creamos nuevos nombres DLAMIs para cada una de las dos categorías con nombres nuevos y nuevos. AMI IDs No DLAMIs son intercambiables. Es decir, los DLAMIs miembros de un grupo no apoyen las instancias que el otro grupo apoya. Por ejemplo, el DLAMI que admite P5 no es compatible con G3 y el DLAMI que admite G3 no admite P5.



¿Por qué era necesario este cambio?

Anteriormente, DLAMIs for NVIDIA GPUs incluía un controlador de núcleo propietario deNVIDIA. Sin embargo, la comunidad originaria del núcleo de Linux aceptó un cambio que impide que los controladores del núcleo propietarios, como el NVIDIA GPU controlador, se comuniquen con otros controladores del núcleo. Este cambio se desactiva GPUDirect RDMA en las instancias de las series P4 y P5, que es el mecanismo que permite utilizarlas de forma eficiente GPUs para la formación distribuida. EFA Como resultado, DLAMIs ahora utilice el controlador OpenRM (controlador de código NVIDIA abierto), vinculado a los EFA controladores de código abierto para admitir G4dn, G5, P4 y P5. Sin embargo, este controlador de OpenRM no es compatible con instancias más antiguas (como P3 y G3). Por lo tanto, para asegurarnos de seguir ofreciendo versiones actuales, eficaces y seguras DLAMIs que admitan ambos tipos de instancias, nos DLAMIs dividimos en dos grupos: uno con el controlador OpenRM (que admite G4dn, G5, P4 y P5) y otro con el controlador propietario anterior (que admite P3, P3dn y G3).

¿DLAMIsA qué afectó este cambio?

Este cambio afectó a todosDLAMIs.

¿Qué significa esto para ti?

Todas DLAMIs seguirán proporcionando funcionalidad, rendimiento y seguridad siempre que las ejecute en un tipo de instancia de Amazon Elastic Compute Cloud (AmazonEC2) compatible. Para determinar los tipos de instancias que DLAMI admite una EC2 instancia, consulta las notas de la versión yDLAMI, a continuación, busca EC2las instancias compatibles. Para obtener una lista de DLAMI las opciones compatibles actualmente y los enlaces a sus notas de versión, consulteNotas de la versión de DLAMIs.

Además, debe usar la correcta AWS Command Line Interface (AWS CLI) comandos para invocar la corrienteDLAMIs.

Para bases DLAMIs compatibles con P3, P3dn y G3, utilice este comando:

```
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
```

Para bases DLAMIs compatibles con G4dn, G5, P4 y P5, utilice este comando:

```
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
```

¿Hay alguna pérdida de funcionalidad con la versión más nueva? DLAMIs

No, no hay pérdida de funcionalidad. Las actuales DLAMIs proporcionan toda la funcionalidad, el rendimiento y la seguridad de las anterioresDLAMIs, siempre que las ejecutes en un tipo de EC2 instancia compatible.

¿Afectó este cambio a Deep Learning Containers?

No, este cambio no afectó AWS Deep Learning Containers, porque no incluyen el NVIDIA controlador. Sin embargo, asegúrese de ejecutar Deep Learning Containers AMIs que sean compatibles con las instancias subyacentes.

Información relacionada sobre DLAMI

Puede encontrar otros recursos con información relacionada DLAMI fuera del AWS Deep Learning AMIs Guía para desarrolladores. Activado AWS re:Post, consulte las preguntas DLAMI de otros clientes o haga las suyas propias. En el AWS Blog de Machine Learning y otros AWS blogs, lee publicaciones oficiales sobreDLAMI.

AWS re:Post

Etiqueta: AWS Deep Learning AMIs

AWS Blog

- AWS Blog de Machine Learning | Categoría: AWS Deep Learning AMIs
- AWS Blog de Machine Learning | Capacitación más rápida con la versión TensorFlow 1.6 optimizada en instancias EC2 C5 y P3 de Amazon
- AWS Blog de Machine Learning | Nuevo AWS Deep Learning AMIs para profesionales de Machine Learning
- AWS Partner Network (APN) Blog | Nuevos cursos de formación disponibles: Introducción a Machine Learning y Deep Learning en AWS
- AWS Blog de noticias | Adéntrate en el aprendizaje profundo con AWS

Notas de la versión de DLAMIs

Aquí encontrarás las notas de versión detalladas de todas las versiones compatibles actualmente AWS Deep Learning AMIs (DLAMI) opciones.

Para ver las notas de versión de DLAMI los marcos que ya no admitimos, consulte la sección Archivo de notas de versiones de marcos no compatibles de la página de políticas de soporte de DLAMI marcos.

> **Note**
>
> La AWS Deep Learning AMIs tienen un ritmo de publicación de parches de seguridad cada noche. No incluimos estos parches de seguridad incrementales en las notas de lanzamiento oficiales.

Base DLAMIs

GPU

- X86
  - AWS Base de aprendizaje profundo AMI (Amazon Linux 2)
  - AWS Base de aprendizaje profundo AMI (Ubuntu 22.04)
  - AWS Base de aprendizaje profundo AMI (Ubuntu 20.04)
- ARM64
  - AWS Base de aprendizaje profundo ARM64 AMI (Ubuntu 22.04)
  - AWS Base de aprendizaje profundo ARM64 AMI (Amazon Linux 2)

AWS Neurona

- X86
  - AWS Base de aprendizaje profundo AMI Neuron (Amazon Linux 2)
  - AWS Base de aprendizaje profundo AMI Neuron (Ubuntu 20.04)

Qualcomm

- X86
  - AWS Base de aprendizaje profundo Qualcomm AMI (Amazon Linux 2)

Marco único DLAMIs

PyTorch-específico AMIs

GPU

- X86
  - AWS Aprendizaje profundo AMI GPU PyTorch 2.3 (Ubuntu 20.04)
  - AWS Aprendizaje profundo AMI GPU PyTorch 2.3 (Amazon Linux 2)
  - AWS Aprendizaje profundo AMI GPU PyTorch 2.2 (Ubuntu 20.04)
  - AWS Aprendizaje profundo AMI GPU PyTorch 2.2 (Amazon Linux 2)
  - AWS Aprendizaje profundo AMI GPU PyTorch 1.13 (Amazon Linux 2)
  - AWS Aprendizaje profundo AMI GPU PyTorch 1.13 (Ubuntu 20.04)
- ARM64
  - AWS Aprendizaje profundo ARM64 AMI GPU PyTorch 2.3 (Ubuntu 22.04)
  - AWS Aprendizaje profundo ARM64 AMI GPU PyTorch 2.2 (Ubuntu 20.04)

AWS Neurona

- X86
  - AWS AMINeuron de aprendizaje profundo PyTorch 1.13 (Amazon Linux 2)
  - AWS AMINeuron de aprendizaje profundo PyTorch 1.13 (Ubuntu 20.04)

TensorFlow-específico AMIs

GPU

- X86
  - AWS Aprendizaje profundo AMI GPU TensorFlow 2.16 (Amazon Linux 2)
  - AWS Aprendizaje profundo AMI GPU TensorFlow 2.16 (Ubuntu 20.04)
  - AWS Aprendizaje profundo AMI GPU TensorFlow 2.15 (Amazon Linux 2)

- AWS Aprendizaje profundo AMI GPU TensorFlow 2.15 (Ubuntu 20.04)

AWS Neurona

- X86
  - AWS AMINeuron de aprendizaje profundo TensorFlow 2.10 (Amazon Linux 2)
  - AWS AMINeuron de aprendizaje profundo TensorFlow 2.10 (Ubuntu 20.04)

Marco múltiple DLAMIs

> **Tip**
>
> Si usa solo un marco de aprendizaje automático, le recomendamos un marco único DLAMI.

GPU

- X86
  - AWS Aprendizaje profundo AMI (Amazon Linux 2)

AWS Neurona

- X86
  - AWS AMINeurona de aprendizaje profundo (Ubuntu 22.04)

Características obsoletas de DLAMI

En la siguiente tabla se enumeran las funciones obsoletas de AWS Deep Learning AMIs (DLAMI), la fecha en que las pusimos en desuso y los detalles sobre el motivo por el que las pusimos en desuso.

Característica	Date	Detalles
Ubuntu 16.04	10/07/2021	Ubuntu Linux 16.04 LTS llegó al final de su período de cinco años LTS el 30 de abril de 2021 y su proveedor ya no lo admite. A partir de octubre de 2021, ya no habrá actualizaciones de Deep Learning Base AMI (Ubuntu 16.04) en las nuevas versiones. Las versiones anteriores seguirán estando disponibles.
Amazon Linux	7 de octubre de 2021	Amazon Linux es a end-of-lifepartir de diciembre de 2020. A partir de octubre de 2021, ya no hay actualizaciones de Deep Learning AMI (Amazon Linux) en las nuevas versiones. Las versiones anteriores de Deep Learning AMI (Amazon Linux) seguirán estando disponibles.
Chainer	01/07/2020	Chainer ha anunciado el final de los principales lanzamientos a partir de diciembre de 2019. En

Característica	Date	Detalles
		consecuencia, dejaremos de incluir los entornos de Chainer Conda a DLAMI partir de julio de 2020. Las versiones anteriores DLAMI que contienen estos entornos seguirán estando disponibles. Proporcio naremos actualizaciones a estos entornos solo si la comunidad de código abierto publica correcciones de seguridad para estos marcos.
Python 3.6	15/06/2020	Debido a las solicitudes de los clientes, vamos a pasar a Python 3.7 para nuevas versiones de TF/MX/PT.
Python 2	01/01/2020	La comunidad de código abierto de Python ha terminado oficialmente la compatibilidad con Python 2. Las MXNet comunidades TensorFlow PyTorch, y también han anunciado que las versiones TensorFlo w 1.15, TensorFlow 2.1, PyTorch 1.4 y MXNet 1.6.0 serán las últimas compatibl es con Python 2.

Historial de documentos de DLAMI

La siguiente tabla proporciona un historial de las DLAMI versiones recientes y los cambios relacionados con la AWS Deep Learning AMIs Guía para desarrolladores.

Cambios recientes

Cambio	Descripción	Fecha
ARM64 DLAMI	La AWS Deep Learning AMIs ahora admite imágenes basadas en el procesador GPUs Arm64.	29 de noviembre de 2021
TensorFlow 2.	El aprendizaje profundo AMI con Conda ahora viene con TensorFlow 2 con CUDA 10.	3 de diciembre de 2019
AWS Inferencia	El aprendizaje profundo ahora admite AMI AWS El hardware Inferentia y el AWS Neurona. SDK	3 de diciembre de 2019
Uso de TensorFlow Serving con un modelo Inception	Se agregó un ejemplo para usar la inferencia con un modelo Inception para TensorFlow Serving, tanto con Elastic Inference como sin él.	28 de noviembre de 2018
Elastic Inference	Se han añadido los requisito s previos e información relacionada con Elastic Inference a la guía de configuración.	28 de noviembre de 2018
Instalación PyTorch desde una compilación nocturna	Se agregó un tutorial que explica cómo desinstalar PyTorch e instalar una versión	25 de septiembre de 2018

	nocturna PyTorch en su Deep Learning AMI con Conda.	
Tutorial de Conda	El ejemplo MOTD se actualizó para reflejar una versión más reciente.	23 de julio de 2018

Cambios anteriores

La siguiente tabla proporciona un historial de las DLAMI versiones anteriores y los cambios relacionados anteriores a julio de 2018.

Cambio	Descripción	Fecha
TensorFlow con Horovod	Se ha añadido un tutorial para entrenar ImageNet con Horovod TensorFlow .	6 de junio de 2018
Actualización de guía	Se ha añadido la guía de actualización.	15 de mayo de 2018
Nueva regiones y nuevo tutorial de 10 minutos	Se han añadido más regiones: EE.UU. Oeste (Norte de California), América del Sur, Canadá (Central), UE (Londres) y UE (París). Además, la primera versión de un tutorial de 10 minutos titulado: «Cómo empezar con el aprendizaje AMI profundo».	26 de abril de 2018
Tutorial de Chainer	Se agregó un tutorial para usar Chainer en los CPU modos múltiple GPUGPU, individual y único. CUDAla integración se actualizó de	28 de febrero de 2018

Cambio	Descripción	Fecha
	CUDA 8 a CUDA 9 para varios marcos.	
Linux AMIs v3.0, además de la introducción de MXNet Model Server, TensorFlow Serving y TensorBoard	Se agregaron tutoriales para Conda AMIs con nuevas capacidades de servidor de modelos y visualizaciones utilizando MXNet Model Server v0.1.5, Serving v1.4.0 y TensorFlow v0.4.0. TensorBoard AMIy las capacidades del marco CUDA descritas en Conda y en las descripciones generales. CUDA Las notas de la versión más recientes se han movido a https://aws.amazon.com/releasenotes/	25 de enero de 2018
Linux v2.0 AMIs	Base, Source y Conda AMIs actualizados a la versión 2.1. NCCL Source y Conda AMIs se actualizaron con las versiones MXNet 1.0, PyTorch 0.3.0 y Keras 2.0.9.	11 de diciembre de 2017
AMISe agregaron dos opciones de Windows	AMIsLanzamiento de Windows 2012 R2 y 2016: se agregó a la guía de AMI selección y a las notas de la versión.	30 de noviembre de 2017
Publicación inicial de la documentación	Descripción detallada del cambio con un enlace al tema o la sección que se ha modificado.	15 de noviembre de 2017

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.