TIC

TALENTO TECH

Misión 1

INTELIGENCIA ARTIFICIAL

Innovador

Tema 2: Docker para entornos de desarrollo de IA

UTraining

UNIVERSIDAD DE ANTIOQUIA

Universidad de Caldas

ubicua Technology

Campista, llegó el momento de retar tus conocimientos y que los pongas a prueba a través de los diferentes recursos que encontrarás en este espacio como son: conceptos, ejemplos, herramientas, actividades prácticas y retos, los cuales te ayudarán a alcanzar los objetivos trazados en el Nivel Innovador.

En el mundo del desarrollo de IA, la reproducibilidad y la gestión de dependencias son cruciales. Es aquí donde Docker emerge como una herramienta invaluable. Docker permite encapsular aplicaciones y sus dependencias en contenedores, unidades de software ligeras y portables que se ejecutan de manera aislada.

En primer lugar, es importante comprender qué son los contenedores y cómo funcionan. Un contenedor Docker es esencialmente un paquete de software que contiene todo lo necesario para ejecutar una aplicación: código, bibliotecas, herramientas del sistema y configuraciones. Al estar aislados del sistema anfitrión, los contenedores garantizan que la aplicación se ejecute de la misma manera en cualquier entorno, eliminando el temido "funciona en mi máquina".

Para construir estos contenedores, se utiliza un archivo llamado Dockerfile. Este archivo contiene instrucciones paso a paso para crear la imagen del contenedor, incluyendo la instalación de dependencias, la copia del código fuente y la configuración del entorno de ejecución. En el contexto de la IA, un Dockerfile podría incluir la instalación de frameworks como TensorFlow o PyTorch, junto con las bibliotecas específicas del proyecto.

De esta manera, Docker facilita la gestión de dependencias y la creación de entornos reproducibles. Al definir las dependencias en el Dockerfile, se asegura que todos los miembros del equipo trabajen con las mismas versiones de las bibliotecas, evitando conflictos y errores. Además, la portabilidad de los contenedores permite compartir y desplegar proyectos de IA de forma rápida y sencilla en diferentes plataformas, ya sea en la nube, en un servidor local o en una máquina virtual. En definitiva, Docker se posiciona como una herramienta fundamental para

optimizar el desarrollo de proyectos de IA, garantizando la consistencia, la portabilidad y la eficiencia en cada etapa del proceso.

En el contexto actual, donde los proyectos de inteligencia artificial (IA) están en constante evolución, es crucial contar con herramientas que faciliten la gestión y el despliegue de aplicaciones de manera eficiente. Docker se ha posicionado como una solución poderosa para crear entornos consistentes y reproducibles, especialmente en el desarrollo de IA. Este curso está diseñado para proporcionar una comprensión clara de Docker y su aplicación en entornos de desarrollo de IA, permitiendo a los campistas crear y gestionar contenedores de manera efectiva. El objetivo principal es que, al finalizar, los campistas sean capaces de configurar entornos de desarrollo de IA utilizando Docker, garantizando la reproducibilidad y la gestión adecuada de dependencias.

1. Introducción a Docker y contenedores:

El primer paso hacia la utilización de Docker en el desarrollo de IA es comprender qué es Docker y cómo los contenedores pueden transformar la forma en que se despliegan y gestionan las aplicaciones. Docker es una plataforma que automatiza la creación, despliegue y ejecución de aplicaciones en contenedores. Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias, permitiendo que la aplicación se ejecute de manera rápida y confiable en cualquier entorno. A diferencia de las máquinas virtuales, los contenedores son ligeros y comparten el kernel del sistema operativo, lo que los hace más eficientes en términos de recursos.

Durante esta sección del curso, los campistas dedicarán dos horas a explorar los conceptos básicos de Docker y los contenedores. Se presentarán ejemplos prácticos que demuestren cómo los contenedores encapsulan todas las dependencias necesarias para ejecutar una aplicación, eliminando los problemas de "funciona en mi máquina". Se realizarán actividades de bootcamp

que permitirán a los campistas crear su primer contenedor, configurar un entorno básico y entender cómo Docker facilita la portabilidad de las aplicaciones. El objetivo específico es que los campistas comprendan los conceptos fundamentales de Docker y sean capaces de ejecutar aplicaciones simples en contenedores.

2. Creación de Dockerfiles para proyectos de IA

- Una vez que los campistas tengan una comprensión sólida de Docker, el siguiente paso es aprender a crear Dockerfiles, los cuales son instrucciones que Docker sigue para construir una imagen. Una imagen es una plantilla que contiene todo lo necesario para ejecutar una aplicación, como el código fuente, bibliotecas, dependencias y configuraciones. Los Dockerfiles permiten definir de manera declarativa cómo se construirá la imagen, especificando las etapas desde la instalación de dependencias hasta la configuración del entorno de ejecución.
- En esta sección, se dedicarán tres horas a la creación de Dockerfiles específicos para proyectos de IA. Se explorará cómo escribir Dockerfiles eficientes, optimizando el tamaño de las imágenes y reduciendo los tiempos de construcción. Los campistas aprenderán a incluir bibliotecas y frameworks de IA, como TensorFlow o PyTorch, en sus Dockerfiles, asegurando que el entorno de desarrollo esté configurado correctamente para ejecutar modelos de IA. Las actividades de bootcamp estarán orientadas a la práctica, donde los campistas crearán Dockerfiles para diferentes tipos de proyectos de IA, desde simples modelos hasta aplicaciones más complejas que involucren múltiples servicios. El objetivo es que los campistas puedan crear Dockerfiles que no solo construyan imágenes correctamente, sino que también optimicen el rendimiento y la reproducibilidad del entorno de desarrollo.

3. Gestión de dependencias y entornos reproducibles

- El último tema se enfoca en uno de los principales desafíos en el desarrollo de IA: la gestión de dependencias y la creación de entornos reproducibles. En proyectos de IA, es común tener una gran cantidad de dependencias, desde librerías de procesamiento de datos hasta frameworks específicos de aprendizaje automático. La falta de control sobre estas dependencias puede llevar a problemas de

incompatibilidad y dificultades para replicar resultados en diferentes entornos. Docker ofrece una solución a este problema al encapsular todas las dependencias dentro de contenedores, garantizando que el entorno de desarrollo sea idéntico en cualquier máquina donde se ejecute.

- En esta sección de tres horas, los campistas aprenderán a gestionar dependencias de manera efectiva utilizando Docker. Se cubrirán temas como la creación de imágenes base con todas las dependencias necesarias, la integración de herramientas de gestión de paquetes, y la implementación de buenas prácticas para garantizar la reproducibilidad. Además, se abordará cómo utilizar Docker Compose, una herramienta que permite definir y gestionar aplicaciones multicontenedor, para crear entornos más complejos.
- Las actividades de bootcamp incluirán la creación de un entorno completo para un proyecto de IA, desde la definición de las dependencias hasta la implementación de un flujo de trabajo reproducible. El objetivo es que los campistas logren crear entornos de desarrollo que puedan ser compartidos y ejecutados en cualquier máquina, sin preocuparse por problemas de compatibilidad o versiones de dependencias.
- A lo largo del curso, los campistas tendrán la oportunidad de aplicar de manera práctica los conceptos aprendidos, asegurando que al finalizar estén capacitados para implementar Docker en sus proyectos de IA. Con un enfoque en la práctica a través de actividades de bootcamp, el curso permitirá a los campistas no solo entender la teoría detrás de Docker, sino también aplicarla en escenarios reales de desarrollo de IA.
- Este curso está diseñado para que los campistas avancen paso a paso, desde la comprensión básica de Docker hasta la creación de entornos reproducibles, con objetivos claros y medibles que les permitan evaluar su progreso y adquirir habilidades prácticas para mejorar su flujo de trabajo en el desarrollo de IA.

Aprendizajes prácticos

Ejercicios prácticos: Docker para entornos de desarrollo de IA (parte 1)

Ejemplo 1: Crear una imagen Docker para un entorno de Python con TensorFlow (60 minutos).

Objetivo: Demostrar cómo crear una imagen Docker que contenga Python y TensorFlow para proyectos de IA.

Pasos:

1. **Crear un Dockerfile**
2. **Construir la imagen**
3. **Ejecutar un contenedor basado en la imagen**
4. **Verificar la instalación de TensorFlow**

Archivo Docker:

```
# Usa una imagen base oficial de Python
 DESDE python:3.8-slim-buster

 # Establece el directorio de trabajo en el contenedor
 directorio_trabajo /aplicación

 # Copia los archivos de requisitos al directorio de trabajo
 COPIA requisitos.txt .

 #Instala las dependencias
 EJECUTAR pip install --no-cache-dir -r requirements.txt

 # Copia el resto de la aplicación al directorio de trabajo
 COPIAR . .

 # Comando para ejecutar la aplicación
 CMD ["python", "aplicación.py"]
```

Requisitos.txt:

```
flujo tensorial==2.6.0
 número = 1,19,5
```

Aplicación.py:

```
importar tensorflow como tf
 importar numpy como np

 print("Versión de TensorFlow:", tf.__version__)
 print("Versión de NumPy:", np.__version__)

 # Crear un tensor simple
 tensor = tf.constante([[1, 2], [3, 4]])
 imprimir("Tensor:", tensor)

 # Realizar una operación simple
 resultado = tf.matmul(tensor, tf.transpose(tensor))
 print("Resultado de la multiplicación de matrices:", resultado)
```

Comandos para construir y ejecutar:

```
# Construir la imagen
 docker build -t tensorflow-python:latest .

 #Ejecutar el contenedor
 docker run --rm tensorflow-python:último
```

Ejemplo 2: Usar un contenedor Docker para entrenar un modelo de IA (60 minutos).

Objetivo: Mostrar cómo usar un contenedor Docker para entrenar un modelo de regresión lineal simple.

Pasos:

1. **Crear un script de Python para entrenar el modelo**
2. **Modificar el Dockerfile para incluir el nuevo script**
3. **Construir la imagen actualizada**
4. **Ejecutar el contenedor para entrenar el modelo**

Modelo_de_tren.py:

```
importar tensorflow como tf
 importar numpy como np
 importar matplotlib.pyplot como plt

 # Generar datos sintéticos
 np.random.seed(0)
 X = np.linspace(0, 10, 100).reshape(-1, 1)
 y = 2 * X + 1 + np.random.randn(100, 1)

 # Crear y compilar el modelo
 modelo = tf.keras.Sequential([
     tf.keras.layers.Dense(1, forma_de_entrada=(1,))
 ])
 modelo.compile(optimizador='sgd', pérdida='mse')

 # Entrenar al modelo
 historial = modelo.fit(X, y, épocas=100, verbose=0)

 # Visualizar los resultados
 plt.dispersión(X, y)
 plt.plot(X, modelo.predict(X), color='rojo')
 plt.title('Regresión Lineal')
 plt.xlabel('X')
 plt.ylabel('y')
 plt.savefig('/app/diagrama_de_regresión.png')
 print("Gráfico guardado como regression_plot.png")

 # Imprimir los pesos del modelo
 print("Pesos del modelo:", model.get_weights())
```

Modificar el Dockerfile:

```
... (mantener las líneas anteriores)

 # Instalar matplotlib
 EJECUTAR pip install matplotlib

 # ... (mantener las líneas restantes)

 # Cambiar el comando para ejecutar el nuevo script
 CMD ["python", "modelo_de_entrenamiento.py"]
```

Comandos para construir y ejecutar:

```
# Construir la imagen actualizada
 docker build -t tensorflow-train:latest .

 #Ejecutar el contenedor
 docker run --rm -v $(pwd):/app tensorflow-train:latest
```

Ejercicios prácticos: Docker para entornos de desarrollo de IA (Parte 2)

Ejemplo 1: Crear un entorno de desarrollo de IA con Docker Compose (60 minutos).

Objetivo: Demostrar cómo crear un entorno de desarrollo de IA con múltiples servicios usando Docker Compose.

Pasos:

1. **Crear un archivo docker-compose.yml con servicios para Jupyter Notebook y TensorFlow Serving.**

2. **Configurar una red Docker para la comunicación entre servicios.**
3. **Usar volúmenes para persistir notebooks y modelos.**
4. **Ejecutar el entorno y verificar su funcionamiento.**

Docker-compose.yml

```
versión: '3'
 servicios:
   Jupyter:
     imagen: jupyter/tensorflow-notebook
     puertos:
       - "8888:8888"
     volúmenes:
       - ./cuadernos:/casa/jovyan/trabajo
     redes:
       - red de inteligencia artificial

   servicio de tensorflow:
     imagen: tensorflow/serving
     puertos:
       - "8501:8501"
     volúmenes:
       - ./modelos:/modelos
     ambiente:
       - MODEL_NAME=mi_modelo
     redes:
       - red de inteligencia artificial

 redes:
   red ai:
     conductor: puente

 volúmenes:
   Cuadernos:
   modelos:
```

Comandos para ejecutar:

```
# Iniciar los servicios
 docker-compose arriba -d

 # Verificar los servicios en ejecución
 docker-compose ps

 # Obtener la URL de Jupyter Notebook
 registros de docker-compose jupyter
```

Ejemplo 2: Entrenar y servir un modelo de IA usando Docker Compose (60 minutos).

Objetivo: Mostrar cómo entrenar un modelo de IA y servirlo usando TensorFlow Serving en un entorno Docker Compose.

Pasos:

1. **Crear un script de Python para entrenar un modelo simple.**
2. **Modificar el docker-compose.yml para incluir un servicio de entrenamiento.**
3. **Entrenar el modelo y guardarlo en un volumen compartido.**
4. **Servir el modelo entrenado usando TensorFlow Serving.**

Modelo_de_tren.py:

```
importar tensorflow como tf
 importar numpy como np
 importar sistema operativo

 # Generar datos sintéticos
 x_tren = np.random.rand(1000, 1)
 tren_y = 2 * tren_x + 1 + np.random.randn(1000, 1) * 0.1

 # Crear y compilar el modelo
 modelo = tf.keras.Sequential([
     tf.keras.layers.Dense(1, forma_de_entrada=(1,))
 ])
 modelo.compile(optimizador='adam', pérdida='mse')

 # Entrenar al modelo
 modelo.fit(x_train, y_train, épocas=100, verbose=0)

 # Guardar el modelo en formato SavedModel
 model_dir = '/modelos/mi_modelo/1'
 os.makedirs(directorio_modelo, exist_ok=Verdadero)
 modelo.save(directorio_del_modelo)
 print(f"Modelo guardado en {model_dir}")
```

Modificar docker-compose.yml:

```
versión: '3'
 servicios:
   Jupyter:
     imagen: jupyter/tensorflow-notebook
     puertos:
       - "8888:8888"
     volúmenes:
       - ./cuadernos:/casa/jovyan/trabajo
     redes:
       - red de inteligencia artificial

   tren:
     construir:
       contexto: .
       archivo docker: Dockerfile.train
     volúmenes:
       - ./modelos:/modelos
     redes:
       - red de inteligencia artificial

   servicio de tensorflow:
     imagen: tensorflow/serving
     puertos:
       - "8501:8501"
     volúmenes:
       - ./modelos:/modelos
     ambiente:
       - MODEL_NAME=mi_modelo
     redes:
       - red de inteligencia artificial

 redes:
   red ai:
     conductor: puente

 volúmenes:
   Cuadernos:
   modelos:
```

Archivo Dockerfile.train:

```
DESDE python:3.8-slim-buster

 EJECUTAR pip install tensorflow numpy

 directorio_trabajo /aplicación
 COPIA train_model.py .

 CMD ["python", "modelo_de_entrenamiento.py"]
```

Comandos para ejecutar:

```
# Construir y ejecutar los servicios
 docker-compose arriba --build

 # Verificar que el modelo se ha guardado
 docker-compose exec tensorflow-serving ls /modelos/mi_modelo/1

 # Hacer una predicción usando el modelo servido
 curl -X POST
"http://localhost:8501/v1/models/mi_modelo:predict" \
      -d '{"instancias": [[2.0], [3.0], [4.0]]}'
```

Material complementario

Campista, en este espacio encontrarás ayudas en diferentes formatos que pueden potenciar tu proceso de aprendizaje.

- **Procesamiento del lenguaje natural con python (NLTK).**
  https://jesuslc.com/2012/11/28/procesamiento-del-lenguaje-natural-con-python-nltk/

- **Guía de Docker para principiantes: cómo crear tu primera aplicación Docker.**
  https://www.freecodecamp.org/espanol/news/guia-de-docker-para-principiantes-como-crear-tu-primera-aplicacion-docker/

- **Servidores y configuración de Docker: crea el primer contenedor.**
  https://www.redeszone.net/tutoriales/servidores/instalacion-configuracion-docker-contenedor/

- **Guía completa para programadores en Python usando Docker.**
  https://elblogpython.com/tecnologia/guia-completa-para-programadores-en-python-usando-docker/

- **Cómo crear un contenedor Docker en 2024.**
  https://www.hostinger.es/tutoriales/como-crear-contenedor-docker

- **Docker: Getting started.**
  https://docs.docker.com/guides/getting-started/

- **Docker: Building best practices.**

https://docs.docker.com/build/building/best-practices/

Parte 2

- **Servicio de TensorFlow con Docker.**
  https://www.tensorflow.org/tfx/serving/docker?hl=es

- **Docker: Contenedores en redes independientes y como conectarlos.**
  https://blog.tiraquelibras.com/?p=887

- **¿Qué son las redes en docker?**
  https://keepcoding.io/blog/que-son-las-redes-en-docker/

- **Infraestructura LAMP con Docker Compose.**
  https://openwebinars.net/blog/infraestructura-lamp-con-docker-compose/

- **Getting Started with MongoDB.**
  https://www.mongodb.com/docs/manual/tutorial/getting-started/

- **Docker: Volumes.**
  https://docs.docker.com/storage/volumes/