Demonstrasyon: Standartlaştırılmış Proje Değerlendirme Akışı

Bu kılavuz, her öğrencinin projesinin standart bir formatta teslim edilmesini ve hocanın bu projeleri minimum eforla, çakışma yaşamadan değerlendirmesini sağlayan iş akışını tanımlar.

Bölüm 1: Hoca Tarafı - Hazırlık ve Talimatlar

Hocanın yapması gereken iki temel hazırlık vardır: Ortak temel imajı oluşturmak ve öğrencilere net bir talimat dosyası göndermek.

Hoca, daha önceki adımlarda detaylandırıldığı gibi, aşağıdaki base.Dockerfile'ı kullanarak python-java-env:1.0 adında ortak bir temel imaj oluşturur ve bunu kendi lokal Docker ortamında hazır tutar.

base.Dockerfile (Hocanın Bilgisayarında)

```
FROM python:3.11-alpine AS builder
RUN apk add --no-cache build-base linux-headers
WORKDIR /build-area
RUN pip wheel --no-cache-dir --wheel-dir=/wheels jupyter notebook findspark
psutil

FROM python:3.11-alpine
RUN apk add --no-cache openjdk17-jre bash
ENV JAVA_HOME=/usr/lib/jvm/java-17-openjdk
ENV PATH=$PATH:$JAVA_HOME/bin
WORKDIR /app
COPY --from=builder /wheels /wheels
RUN pip install --no-cache-dir /wheels/*
CMD ["bash"]
```

Oluşturma Komutu (Tek Seferlik)

```
docker build -t python-java-env:1.0 -f base.Dockerfile .
```

Hoca, öğrencilere aşağıdaki talimatları ve bir şablon docker-compose.yml dosyasını içeren bir duyuru yayınlar.

DUYURU: Proje Teslimat Kuralları

Sevgili öğrenciler,

Projenizi teslim ederken aşağıdaki kurallara uymanız, değerlendirme sürecinin hızlı ve adil bir şekilde yapılması için kritik öneme sahiptir. Lütfen tüm adımları dikkatle takip ediniz.

Teslimat Formatı:
Projenizi, aşağıdaki 3 dosyayı içeren bir .zip arşivi olarak teslim ediniz:

1. **Notebook Dosyanız (<OgrenciNo>.ipynb):**
    - Ana Python kodunuzu içeren Jupyter Notebook dosyanızın adını **kendi öğrenci numaranız** olarak değiştirin. (Örnek: 201805071.ipynb)

- **Veri Yolu (ÇOK ÖNEMLİ):** Kodunuzun içinde .csv dosyalarını okurken, dosya yolunu ./data/ ön ekiyle belirtmelisiniz. Bu, verilerin ortak bir klasörden okunmasını sağlayacaktır.

```
      # Örnek Doğru Kullanım
books = spark.read.csv('./data/Books.csv', inferSchema=True,
header=True)
ratings = spark.read.csv('./data/Ratings.csv', inferSchema=True,
header=True)
```

2. **Bağımlılık Dosyanız (requirements.txt):**
    - Projenizin ihtiyaç duyduğu **tüm harici Python kütüphanelerini** bu dosyaya, her satıra bir tane gelecek şekilde yazın.
    - Kullandığınız **PySpark sürümünü** mutlaka belirtin. (Örnek: pyspark==3.4.0)
    - Eğer projeniz temel kütüphaneler dışında bir şeye ihtiyaç duymuyorsa, bu dosyayı **boş** olarak teslim edebilirsiniz.
3. **Docker Compose Dosyanız (docker-compose.yml):**
    - Aşağıda verilen şablonu kullanın ve sadece **size özel olarak belirtilen yerleri** düzenleyin.
    - **Port Ayarları:** ports bölümünde, Jupyter portunu 80<NoSonİki> ve Spark UI portunu 40<NoSonİki> olarak değiştirin. Örneğin, numaranız 201805071 ise, son iki rakam 71'dir. Port ayarlarınız şöyle olmalıdır:

```
    ports:
  - "8071:8888"  # Jupyter Notebook
  - "4071:4040"  # Spark UI
```

    - **Konteyner Adı:** container_name bölümüne, proje_<adiniz> şeklinde benzersiz bir isim verin. (Örnek: container_name: proje_yusuf)

Şablon docker-compose.yml Dosyası:

```
  services:
  project:
    image: python-java-env:1.0
    container_name: proje_adiniz # <-- BURAYI DEĞİŞTİRİN
    ports:
      - "80XX:8888"  # <-- XX YERİNE NUMARANIZIN SON İKİ HANESİ
      - "40XX:4040"  # <-- XX YERİNE NUMARANIZIN SON İKİ HANESİ
    volumes:
      - ../data:/app/data:ro
      - .:/app
    command: >
      sh -c "if [ -f requirements.txt ]; then pip install --no-cache-dir -r
requirements.txt; fi && python -m notebook --ip=0.0.0.0 --port=8888 --allow-root
--no-browser --NotebookApp.token="
```

Bölüm 2: Hocanın Değerlendirme İş Akışı (Demonstrasyon)

Hoca, öğrencilerden gelen teslimatları aşağıdaki gibi bir yapıya yerleştirir:

```
      /degerlendirmeler/
|
|-- /data/
|   |-- Books.csv, Ratings.csv, Users.csv
|
|-- /201805071_Ahmet/
|   |-- 201805071.ipynb
|   |-- requirements.txt  (içinde pyspark==3.3.0 yazıyor)
|   |-- docker-compose.yml  (portlar 8071 ve 4071 olarak ayarlı)
|
|-- /201906055_Ayse/
|   |-- 201906055.ipynb
|   |-- requirements.txt  (içinde pyspark==3.5.1 yazıyor)
|   |-- docker-compose.yml  (portlar 8055 ve 4055 olarak ayarlı)
```

Değerlendirme Adımları:

1. **Ahmet'in Projesini Başlatma:**

   ```
   cd /degerlendirmeler/201805071_Ahmet/
   docker compose up
   ```
   Hoca, tarayıcıda http://localhost:8071 adresine giderek Ahmet'in projesini inceler.

2. **Ayşe'nin Projesini Başlatma (Aynı Anda):**
   Hoca, **yeni bir terminal penceresi** açar.

   ```
   cd /degerlendirmeler/201906055_Ayse/
   docker compose up
   ```
   Hoca, aynı tarayıcıda yeni bir sekme açar ve http://localhost:8055 adresine giderek Ayşe'nin projesini inceler.

Sonuç:
Hoca, her öğrencinin kendi belirlediği port numarası sayesinde, iki (veya daha fazla) projeyi **aynı anda, birbirleriyle hiçbir çakışma yaşamadan** çalıştırabilir. Bir öğrencinin projesi Python 3.8 kullanırken, diğerininki Python 3.10 kullanabilir; bu durum hoca için hiçbir fark yaratmaz.

Değerlendirme bittiğinde, her proje klasöründe ayrı ayrı docker compose down komutunu çalıştırarak tüm ortamları temizler. Bu yapı, süreci maksimum düzeyde verimli, standart ve hatasız hale getirir.

Senaryo: Çok Eski Bir Spark Sürümü (Spark 2.4.8) Gereksinimi

Diyelim ki bir öğrenci, sadece Java 8 ile uyumlu olan çok eski bir Spark sürümü olan pyspark==2.4.8'i kullanmak zorunda kaldı.

Hocanın python-java-env:1.0 imajı Java 17 içerdiği için bu öğrencinin projesi çalışmayacaktır.

Adım 1: Yeni Bir Temel İmaj İçin Dockerfile Hazırlama

Hoca, bu özel durum için yeni bir "tarif" dosyası oluşturur. Bu dosyayı base-legacy.Dockerfile olarak adlandırabilir.

base-legacy.Dockerfile:

```
# Bu Dockerfile, Java 8 gerektiren eski projeler için bir temel oluşturur.

# --- AŞAMA 1: BUILDER ---
# Temel Python sürümünü de projenin ihtiyacına göre değiştirebiliriz.
# Örneğin, Python 3.7 eski projeler için daha uygun olabilir.
FROM python:3.7-alpine AS builder

RUN apk add --no-cache build-base linux-headers
WORKDIR /build-area

# Bu imajın temel kütüphanelerini tanımlıyoruz.
# PySpark'ı burada belirtmiyoruz, çünkü öğrenci kendi versiyonunu kuracak.
RUN pip wheel --no-cache-dir --wheel-dir=/wheels jupyter notebook findspark
psutil


# --- AŞAMA 2: FINAL ---
FROM python:3.7-alpine

# EN ÖNEMLİ DEĞİŞİKLİK: Java 17 yerine Java 8 kuruyoruz.
# Alpine'ın paket deposunda 'openjdk8-jre' olarak bulunur.
RUN apk add --no-cache openjdk8-jre bash

# Ortam değişkenlerini Java 8'e göre güncelliyoruz.
ENV JAVA_HOME=/usr/lib/jvm/java-1.8-openjdk
ENV PATH=$PATH:$JAVA_HOME/bin

WORKDIR /app
COPY --from=builder /wheels /wheels
RUN pip install --no-cache-dir /wheels/*

CMD ["bash"]
```

Adım 2: Yeni Temel İmajı Oluşturma (Build)

Hoca, bu yeni Dockerfile'ı kullanarak, farklı bir isim ve etiketle yeni bir temel imaj oluşturur. Bu, mevcut python-java-env:1.0 imajına **dokunmaz**.

Generated bash

```
docker build -t python-java-env:legacy-1.0 -f base-legacy.Dockerfile .
```

- **-t python-java-env:legacy-1.0**: Yeni imaja hem farklı bir isim/etiket (legacy) hem de bir versiyon numarası verdik. Bu sayede hangi imajın ne için olduğu net bir şekilde anlaşılır.

Bu komut bittiğinde, hocanın docker images listesinde artık iki temel imajı olacaktır:

```
      REPOSITORY           TAG          ...
python-java-env      1.0          ...  (Java 17 / Python 3.11'li)
python-java-env      legacy-1.0   ...  (Java 8 / Python 3.7'li)
```

Adım 3: Öğrencinin docker-compose.yml Dosyasını Düzenleme

Hoca, bu özel gereksinime sahip öğrencinin docker-compose.yml dosyasını açar ve sadece **tek bir satırı** değiştirir.

Öğrencinin docker-compose.yml Dosyasındaki Değişiklik:

```
      services:
  project:
    # 'image' satırını yeni oluşturulan legacy imajı kullanacak şekilde
günceller.
    image: python-java-env:legacy-1.0

    # Geri kalan her şey aynı kalır...
    ports:
      - "80XX:8888"
      - "40XX:4040"
    volumes:
      - ../data:/app/data:ro
      - .:/app
    command: >
      sh -c "pip install --no-cache-dir -r requirements.txt && ..."
```

Adım 4: Projeyi Çalıştırma

Artık hoca, bu öğrencinin klasörüne gidip docker compose up komutunu çalıştırdığında, Docker otomatik olarak doğru temel imajı (legacy-1.0) seçecek ve öğrencinin eski pyspark==2.4.8 sürümünü, uyumlu olduğu Java 8 ortamında sorunsuzca kurup çalıştıracaktır.

Sonuç ve Esneklik

Bu yaklaşım, hocaya inanılmaz bir esneklik sunar:

- **Çoklu Temel Ortamlar:** Hoca, farklı altyapı gereksinimleri için birden fazla "temel imaj" hazırlayıp bilgisayarında tutabilir.
- **Minimal Değişiklik:** Bir projenin tamamen farklı bir altyapıda çalıştırılması, öğrencinin docker-compose.yml dosyasındaki **sadece image: satırını** değiştirmek kadar basittir.
- **Geleceğe Uyumlu:** Gelecekte çıkacak ve belki de Java 21 gerektirecek bir Spark 4.0 sürümü için de hoca, sadece yeni bir base.Dockerfile hazırlayıp python-java-env:spark4-1.0 gibi yeni bir temel imaj oluşturabilir.

Bu, modüler ve "tak-çıkar" (plug-and-play) bir sistem kurduğumuzun ve bu sistemin öngörülemeyen gereksinimlere bile kolayca adapte olabileceğinin en güzel kanıtıdır.