دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

مبانی رایانش ابری

تمرین دوم (فاز ۰ و ۱)

داکر

طراحان تمرین:

حسین فخارزاده، حمید رضائی

استاد درس:

دکتر جوادی

مهلت نهایی ارسال پاسخ:

۱۰ آبان ۱۴۰۳

مقدمه

هدف از این تمرین، کار با داکر[1] است. بنابراین در این تمرین یک پروژه بسیار ساده را با استفاده از داکر، کانتینرایز[2] می‌کنید. برای پیش نیاز، لازم است داکر را بر روی سیستم خود نصب کرده باشید. پیشنهاد ما این است که برای سهولت و عدم برخورد به مشکلات، از یک توزیع لینوکس برای این کار استفاده کنید. برای راهنما می توانید از لینک‌های زیر استفاده کنید. دقت داشته باشید که برای اتصال به dockerhub، نیاز به تنظیم شکن یا استفاده از فیلترشکن دارید.

Get Docker | Docker Docs

[1] Docker
[2] Containerize

فاز ۰

بخش ۱)

در این بخش از چند دستور ابتدایی داکر استفاده میکنیم. برای این کار ابتدا ایمیج بازی sonic را با دستور زیر pull کنید:

```
docker pull ghcr.io/aut-cloud-computing-fall-2024/sonic:latest
```

سپس فعالیت های زیر را به ترتیب انجام دهید:

1. مشخص کنید این ایمیج از چند لایه ساخته شده است و روی چه پورتی اجرا میشود؟
2. ایمیج پول شده را اجرا کنید.
3. لاگ های ایجاد شده توسط کانتینر ایمیج را نشان دهید.
4. ایمیج را روی یک پورت دلخواه دیگر اجرا کنید.
5. در نهایت کانتینر را متوقف کرده و آن را حذف کنید.

موارد زیر را در فایل گزارش نمایش دهید:

- پول کردن ایمیج
- نمایش لیست ایمیج های موجود روی سیستم خود
- نمایش لایه های ایمیج
- نمایش پورت اجرایی کانتینر
- نمایش بازی در مرورگر
- لاگ های ایجاد شده توسط کانتینر
- تغییر پورت کانتینر و اجرای آن روی پورت جدید
- توقف و حذف کانتینر

بخش ۲)

در زیر یک Dockerfile ساده آورده شده است. در گزارش خود مشخص کنید دستورات هر خط چه کاری انجام میدهند؟

```
FROM python:3.9
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

در داکرهاب یك اکانت بسازید و سپس فعالیت‌های زیر را به ترتیب انجام دهید:

1. یک برنامه ساده بنویسید که نام، نام خانوادگی، شماره دانشجویی شما و عبارت زیر را به ترتیب پرینت کند
Welcome To Cloud Computing Course - Fall 2024
همچنین لازم است که Dockerfile خود را به گونه‌ای بنویسید که پس از ایجاد کانتینر[3] از ایمیجی که ساخته‌اید، کد داده شده بلافاصله پس از بالا آمدن کانتینر، در آن اجرا شود و نتایج آن نمایش داده شود.
2. با استفاده از Dockerfileی که نوشته‌اید، ایمیج خود را build نمایید.
اکثر اوقات استفاده از ایمیج های عادی در Dockerfile به عنوان Base، باعث زیاد شدن حجم ایمیج build شده میشود. تحقیق کنید که چگونه میتوان ایمیجی با حجم کمتر ساخت و سپس Dockerfile خود را تغییر دهید و ایمیج جدید را با یک tag جداگانه مشخص کنید.
3. ایمیج جدید را بر روی داکرهاب push نمایید.
4. برای تست کردن ایمیج جدید، ایمیج خود را از داکرهاب pull کنید و یك کانتینر از آن بالا بیاورید.

موارد زیر را در فایل گزارش نمایش دهید:

- ساخت ایمیج از روی Dockerfile
- ایجاد کانتینر از ایمیج ساخته شده و نمایش نتیجه آن
- توضیح چگونگی کم کردن حجم ایمیج و ساخت ایمیج کم حجم
- ارسال ایمیج کم حجم بر روی داکرهاب و نتیجه آن
- دریافت ایمیج کم حجم از داکرهاب
- نمایش لیست ایمیج های موجود بر روی سیستم خود
- ساخت کانتینر از ایمیج کم حجم دریافت شده از داکرهاب
- نمایش نتیجه ساخت کانتینر

[3] Container

بخش ۳)

فرض کنید یک Dockerfile با محتویات زیر داریم:

```
FROM python:3.9
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

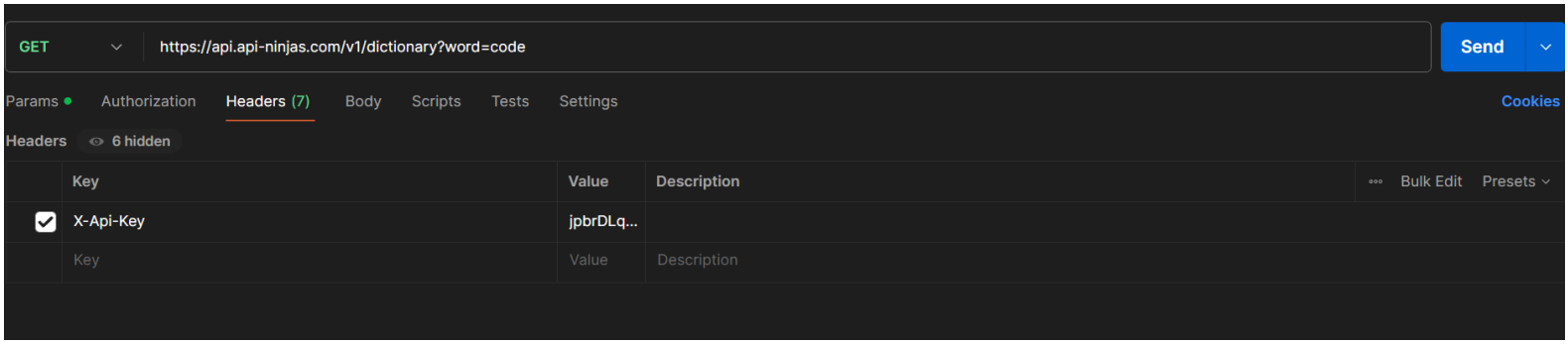
به سوالات زیر پاسخ دهید:

1. چرا هر بار که دستور docker build را روی این داکرفایل اجرا میکنیم بیلد کردن ایمیج طول میکشد (حتی اگر تغییرات خیلی جزئی باشند)؟
2. توضیح دهید چگونه میتوان این فایل را ادیت کرد تا هنگامی که صرفا کدبیس را تغییر میدهیم ولی dependency ها ثابت هستند، ایمیج سریعتر build شود؟
3. توضیح دهید فرآیند caching در داکر چگونه است و تغییر انجام شده چرا build را سریعتر میکند؟

فاز ۱

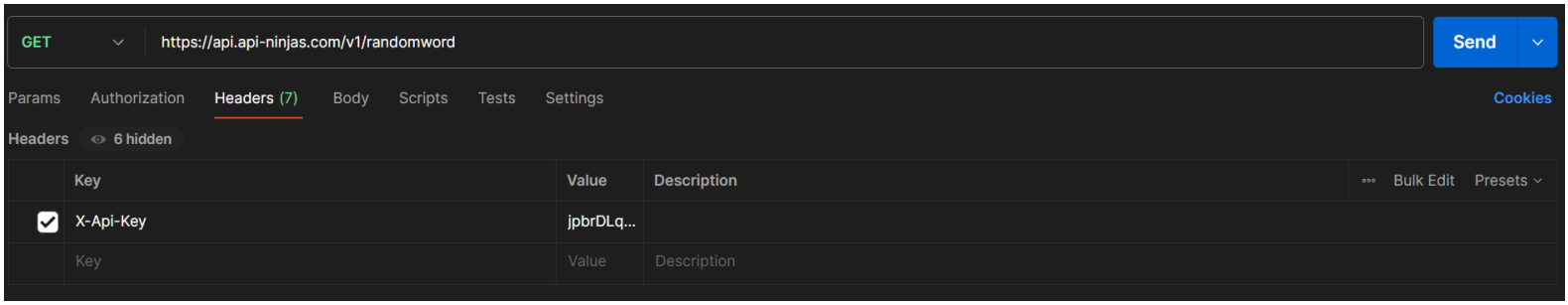
فرض کنید که قصد دارید برای اپلای زبان مطالعه کنید. به همین دلیل تصمیم می‌گیرید که یک سرور را برای یادگیری زبان انگلیسی توسعه دهید. برای این کار از دو API که در ادامه آمده اند استفاده خواهید کرد:

1. اولین API مربوط به گرفتن Definition یک کلمه ورودی است.
2. دومین API به طور رندوم یک کلمه انگلیسی را بازمیگرداند.




```
{
    "definition": "1. A body of law, sanctioned by legislation, in which the rules of law to be specifically applied by the courts are set forth in systematic form; a compilation of laws by public authority; a digest. Note: The collection of laws made by the order of Justinian is sometimes called, by way of eminence. \"The Code\" Wharton. 2. Any system of rules or regulations relating to one subject; as, the medical code, a system of rules for the regulation of the professional conduct of physicians; the naval code, a system of rules for making communications at sea means of signals. Code civil or Code Napoleon, a code enacted in France in 1803 and 1804, embodying the law of rights of persons and of property generally. Abbot.",
    "word": "code",
    "valid": true
}
```


یک نمونه درخواست و پاسخ از API شماره یک




```
{
    "word": [
        "little"
    ]
}
```


یک نمونه درخواست و پاسخ از API شماره دو

سرور شما باید دو API به کاربر ارائه دهد. API اول یک کلمه انگلیسی را به عنوان ورودی دریافت میکند و معنی آن را به عنوان خروجی بازمیگرداند. API دوم به طور رندوم یک کلمه انگلیسی و **معنی آن** را بازمیگرداند. دقت کنید که حتما از ابتدا از گیت و گیتهاب به عنوان version control استفاده کنید چون **در ادامه به آن نیاز پیدا خواهید کرد**

توجه: برای استفاده از APIها شما باید در سایت داده شده ثبت نام کنید و از API KEYای که در اختیار شما قرار داده میشود در Header درخواست های خود به سایت استفاده کنید.

بعد از اینکه نتیجه را به کاربر گزارش کردید، باید با استفاده از redis پاسخ آن را کش کنید (برای 5 دقیقه) و در صورتی که کاربر دوباره معنی آن کلمه را درخواست کرد، جواب او را از redis بدهید. (در پاسخ برگردانده شده توسط سرور باید مشخص باشد که جواب از redis گرفته شده است یا از api-ninjas)
برنامه شما باید قابلیت کانفیگ پذیری داشته باشد و بتوان **مدت زمان ذخیره پاسخ در ردیس**، **پورت اجرایی سرور** و **همچنین API KEY** را از طریق کانفیگ تغییر داد.

یک داکرفایل برای برنامه خود بنویسید و آن را build کنید و ایمیج آن را روی داکرهاب push کنید. سپس ایمیج push شده را pull کنید و آن را اجرا کنید.

توجه: دقت کنید که باید یک کانتینر جداگانه برای redis اجرا کنید و کانتینر سروری که نوشته اید را به آن وصل کنید.
می‌توانید ایمیج redis را با دستور زیر pull کنید:

```
docker pull ghcr.io/aut-cloud-computing-fall-2024/redis:latest
```

توجه: برای اینکه دو کانتینری که اجرا کردید بتوانند به یکدیگر وصل شوند باید یک network تعریف کنید و دو کانتینر را به آن وصل کنید.

پس از تست سرور، یک بار کانتینر ردیس را حذف کنید و دوباره آن را ایجاد کنید و معنی یک کلمه تکراری را از سرور درخواست کنید. مشاهده می‌شود که معنی کلمه به صورت کش دیگر در redis **موجود نیست**. برای رفع این مشکل و persist کردن دیتا در صورت پایین آمدن کانتینر، باید از قابلیت Volume استفاده کرد. برای این کار یک volume ایجاد کنید و کانتینر redis را دوباره ایجاد کنید و آن را به volume ساخته شده وصل کنید تا دیتا persist شود.
موارد زیر را در فایل گزارش نمایش دهید:

- دریافت ایمیج redis و ساخت کانتینر آن
- داکرفایل نوشته شده برای سرور
- ساخت ایمیج از روی داکرفایل، push و pull کردن آن و اجرای آن
- ساخت شبکه برای برقراری ارتباط بین دو کانتینر

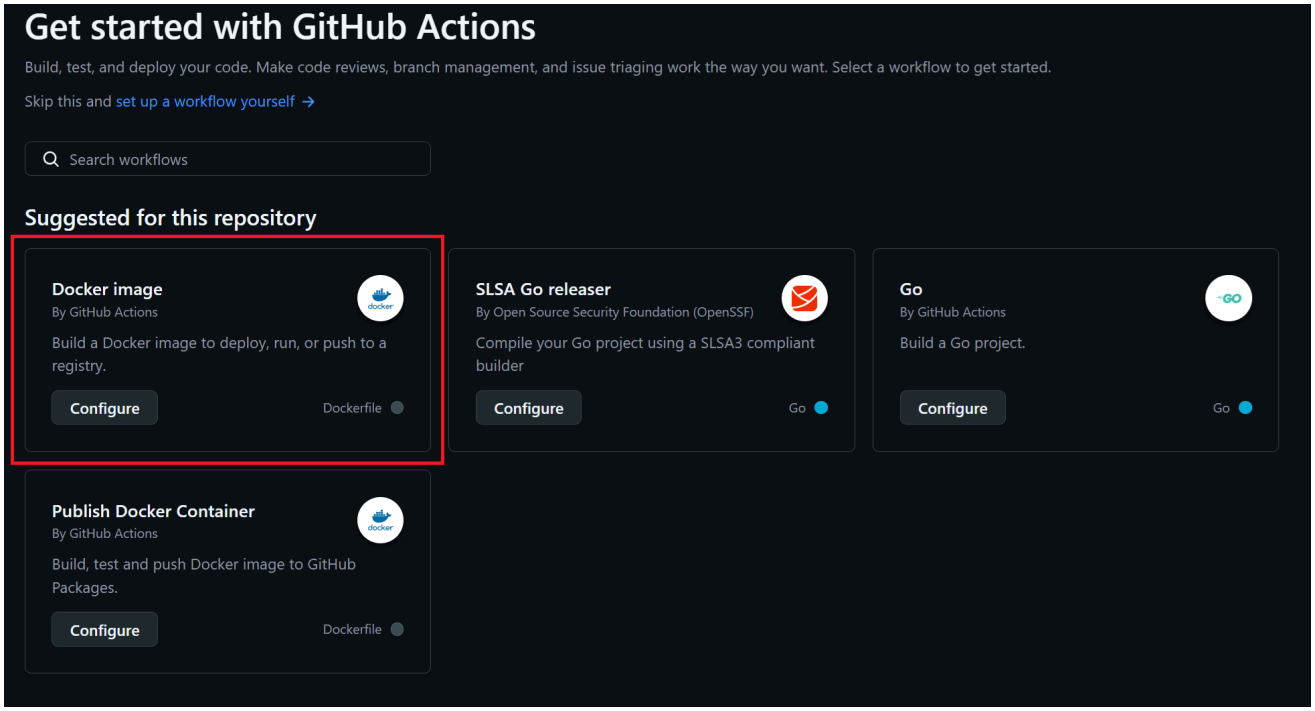
- ساخت Volume برای persist کردن دیتای redis
- نمایش کانتینر های ایجاد شده

در ادامه یک فایل docker-compose تعریف کنید و ایمیج های استفاده شده در گام قبل را در آن استفاده کنید و این بار سیستم را با داکر کامپوز اجرا کنید. دقت کنید که باید Dockerfileای که نوشته اید دوباره در این فایل build شود و همچنین network و volume را نیز باید در این فایل تعریف کنید.

موارد زیر را در فایل گزارش نمایش دهید:

- محتوای فایل docker-composeای که نوشته اید
- نمایش اجرای سرور با داکر کامپوز
- نمایش کانتینر های ایجاد شده با داکر کامپوز
- نمایش کل کانتینر های ایجاد شده با دستور docker ps -a
- نمایش میزان مصرف منابع استفاده شده توسط کانتینر های موجود با دستور docker stats

یکی از کاربرد های سیستم های ابری استفاده از آنها برای راحت‌تر کردن فرایند CI/CD[4] است. یکی از فواید CI خودکارسازی build و push کردن ایمیج بعد از انجام تغییرات در کدبیس می‌باشد. در بخش نهایی فاز یک قصد داریم از Github Actions برای continuous integration استفاده کنیم. برای این کار وارد ریپازیتوری ای که در ابتدای این فاز برای توسعه سرور ساخته اید شوید و وارد بخش Actions شوید و Docker Image را انتخاب کنید.



[4] continuous integration and continuous delivery/deployment

سپس فایل مورد نظر را ادیت کنید تا مطابق پروژه شما باشد (نام آن و تگ آن را ادیت کنید. همچنین دقت کنید بخش های لازم برای push کردن ایمیج مورد نظر به داکرهاب را نیز باید در این فایل اضافه کنید).

Edit Preview Spaces 2 No wrap

```yaml
name: Docker Image CI

on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

jobs:

  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: Build the Docker image
      run: docker build . --file Dockerfile --tag my-image-name:$(date +%s)

```

سپس یک تغییر در کدبیس خود اجرا کنید (برای مثال route تعریف شده برای یکی از endpoint های API را عوض کنید) و تغییر را push کنید. بررسی کنید که آیا ایمیج جدید به درستی ساخته می‌شود و به داکرهاب پوش میشود؟

موارد زیر را در فایل گزارش نمایش دهید:

- محتوای فایل workflowای که نوشتید
- اسکرین شات از انجام job های Github Actions
- انجام تغییرات در کدبیس و پوش کردن و نمایش job جدید در Github Actions
- نشان دادن push شدن ایمیج ساخته شده توسط Actions به داکرهاب

بخش امتیازی)

از prometheus برای ذخیره کردن metric هایی شامل: تعداد درخواست ارسال شده به هر API، تعداد درخواست هایی که از redis خوانده شده، تعداد درخواست های موفق و ناموفق و latency هریک از API های سرور استفاده کنید. دقت کنید که باید prometheus را در فایل داکر کامپوز خود نیز اضافه کنید.

نکات مربوط به تحویل تمرین

- تمرین دارای تحویل آنلاین می‌باشد. از استفاده از کدهایی که توانایی توضیح آنها را ندارید بپرهیزید!
- سوالات خود را می‌توانید از تدریسیاران مرتبط از طریق گروه متصل به کانال تدریسیاری مطرح کنید.
- هرگونه تقلب باعث صفر شدن طرفین می‌شود.

مواردی که باید ارسال شوند:

یک فایل زیپ با نام **StudentID_HW2.zip** که شامل موارد زیر می‌باشد (هر کدام از موارد را در پوشه‌های جداگانه قرار دهید)

- برای فاز ۰ موارد زیر آپلود شوند
  - Dockerfile نوشته شده برای فاز ۰ بخش 2
  - در صورتی که برای رفع مشکل موجود در فاز ۰ بخش ۳، Dockerfile جدید نوشته‌اید، آن را نیز آپلود نمایید.
- برا فاز ۱ تمامی فایل‌های پروژه (شامل کدها و configها) به همراه Dockerfile و docker compose پروژه و فایل workflow نوشته شده برای Github Actions
- گزارشی که حداقل باید شامل موارد مطرح شده در توضیحات تمرین (به همراه اسکرین شات) باشد.

موفق باشید

تیم تدریسیاری درس مبانی رایانش ابری