

به نام خدا



رایانش ابری

تمرین دوم

داکر و مقدمات کوبرنتیز

طراحی تمرین:

آقایان بوکانی و توگلی

استاد درس:

آقای دکتر جوادی

مهلت نهایی ارسال پاسخ:

چهارشنبه ۳۰ فروردین ۱۴۰۲

مقدمه

هدف از این تمرین، کار با داکر¹ و کوبرنتیز² است. بنابراین در این تمرین یک پروژه بسیار ساده را با استفاده از داکر، کانتینرایز³ می‌کنیم و سپس بر روی سرور کوبرنتیز دیپلوی می‌کنیم. برای پیش‌نیاز لازم است داکر و مینی‌کیوب⁴ را بر روی سیستم خود نصب کرده باشید. برای راهنمایی می‌توانید از لینک‌های زیر استفاده کنید. دقت داشته باشید که برای اتصال به dockerhub و کار با Minikube نیاز به تنظیم **شکن** و یا استفاده از فیلترشکن دارید.

<https://docs.docker.com/get-docker/>

<https://minikube.sigs.k8s.io/docs/start/>

گام اول

در ابتدا باید در **داکرهاب** یک حساب کاربری بسازید.

cURL یک ابزار بسیار قدرتمند برای ارسال هر نوع درخواستی در بستر اینترنت توسط هر پروتکلی می‌باشد. در این گام شما باید یک ایمج داکر⁵ بر پایه لینوکس (ترجیحاً alpine ولی برای راحتی کار ubuntu هم مشکلی ندارد) بسازید که در آن امکان استفاده از دستور cURL وجود داشته باشد. سپس ایمج ساخته شده را بر روی داکرهاب آپلود نمایید. سپس برای تست کردن ایمج ساخته شده، ایمج خود را از داکرهاب دانلود کنید و یک کانتینر از آن بالا بیاورید. سپس یک درخواست cURL به wttr.in/moon ارسال کنید.

موارد زیر را در فایل گزارش نمایش دهید:

- ارسال ایمج ساخته شده بر روی داکرهاب و نتیجه آن
- نمایش لیست ایمج‌های موجود بر روی سیستم خود
- دریافت ایمج ساخته شده از داکرهاب
- ساختن کانتینر از ایمج دریافت شده از داکرهاب
- ارسال درخواست به endpoint اشاره شده با کمک دستور cURL و نتیجه آن

¹ Docker

² Kubernetes

³ Containerize

⁴ Minikube

⁵ Docker Image

گام دوم

کوتاه‌کننده⁶ URL یکی از ابزارهایی که است که امروزه به صورت گسترده مورد استفاده قرار می‌گیرد. از مزایای استفاده از آن می‌توان به خواناتر کردن URL، کمتر کردن تعداد کاراکترهای URL و همچنین اضافه کردن قابلیت ردیابی لینک‌ها و توانایی انجام آنالیز برای توسعه‌ی کسب‌وکار خود اشاره کرد. در این گام قرار است که سروری را برای کوتاه‌کردن URL توسعه دهیم. محدودیتی در API استفاده شده وجود ندارد و بسته به سلیقه خود می‌توانید از هر API موجود برای دریافت اطلاعات استفاده کنید (پیشنهاد می‌کنیم از API‌های دارای کلید⁷ به دلیل دسترس‌پذیری بالاتر استفاده کنید). در قسمت زیر لیست API‌های پیشنهادی آورده شده‌اند:

- [APILayer](#) (نیاز به کلید)
- [Rebrandly](#) (نیاز به کلید)
- [Bitly](#) (نیاز به Access Token)
- [GoTiny](#) (بدون نیاز به احراز هویت)
- [Short-Link-API](#) (بدون نیاز به احراز هویت)

توجه ۱: طبیعتاً برای استفاده از API‌هایی که به کلید نیاز دارند، باید حساب کاربری بسازید (هنگام فعال کردن حساب کاربری حتماً قسمت spam ایمیل خود را نیز چک کنید).

توجه ۲: می‌توانید از هر API دیگری غیر از موارد بالا استفاده کنید.

توجه ۳: با هر زبان برنامه‌نویسی دلخواه می‌توان سرور را توسعه داد.

همچنین برای جلوگیری از ارسال درخواست‌های اضافه در برنامه خود باید از کش ردیس⁸ استفاده کنید. برای استفاده از ردیس، ایمیج آن را دانلود کنید و یک کانتینر از آن بالا بیاورید. در کلید URL طولانی و در مقدار URL کوتاه مربوط به آن را نگه دارید و مقادیر باید به صورت پیش‌فرض به مدت پنج دقیقه اعتبار داشته باشند.

⁶ Shortener

⁷ API Key

⁸ Redis

در شکل‌های زیر، نمونه‌هایی از این درخواست‌ها آورده شده‌اند.

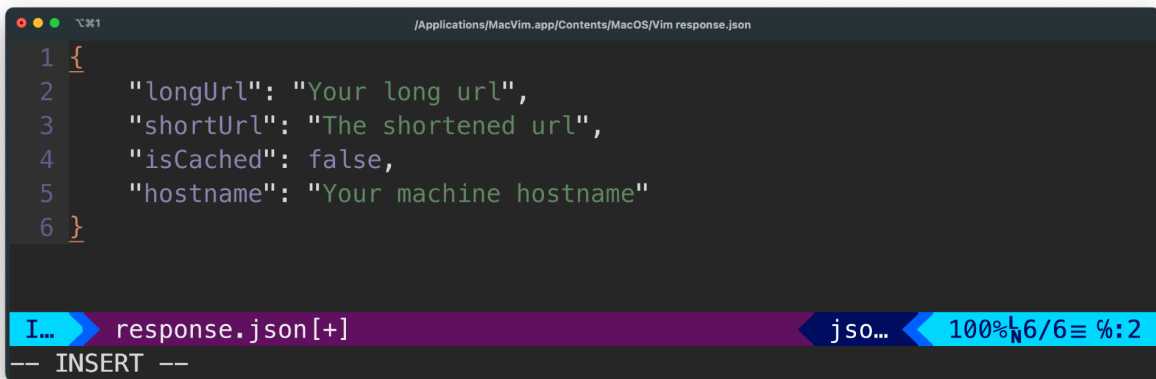
```
macbookpro@2arian3.local ~  
➔ curl -s https://api.rebrandly.com/v1/links \  
-H 'Content-Type: application/json' \  
-H 'apikey: ec9cad667f3c4405afce85d46df24ecc' \  
-d '{"destination": "https://www.tradingview.com/symbols/BTCUSD/", "domain": {"fullNa  
me": "rebrand.ly"}}' | jq | grep shortUrl  
"shortUrl": "rebrand.ly/m3g1n6v",  
macbookpro@2arian3.local ~  
➔ |
```

شکل ۱ نمونه‌ای از درخواست API مربوط به Rebrandly برای کوتاه کردن یک URL

```
macbookpro@2arian3.local ~  
➔ curl -s -X POST https://gotiny.cc/api \  
-H "Content-Type: application/json" \  
-d '{"input": "https://www.tradingview.com/symbols/BTCUSD/"}' | jq  
[  
  {  
    "long": "https://www.tradingview.com/symbols/BTCUSD/",  
    "code": "8p67df"  
  }  
]  
macbookpro@2arian3.local ~  
➔
```

شکل ۲ نمونه‌ای از درخواست API مربوط به GoTiny برای کوتاه کردن یک URL

در شکل زیر هم یک نمونه‌ی خروجی‌ای که سرور توسعه داده شده به ازای هر درخواست باید داشته باشد، آورده شده است.



```
1 {
2   "longUrl": "Your long url",
3   "shortUrl": "The shortened url",
4   "isCached": false,
5   "hostname": "Your machine hostname"
6 }
```

I... response.json[+] j so... 100% 6/6 %:2
-- INSERT --

شکل ۳ ساختار خروجی هر درخواست به سرور توسعه داده شده

نکات مربوط به خروجی:

- در صورتی که مقدار URL کوتاه شده از ردیس خوانده شده است، مقدار isCached برابر با true می باشد و در غیر این صورت false است.
- همانطور که از شکل معلوم است، نام میزبان^۹ را نیز در خروجی باید بیاورید.

نهایتاً در شکل زیر یک نمونه درخواست به سرور توسعه داده شده و خروجی آن را می بینید.



```
macbookpro@2arian3.local ~
> curl -s localhost:8000 \
-d '{"url": "https://github.com/torvalds/linux"}' | jq
{
  "longUrl": "https://github.com/torvalds/linux",
  "shortUrl": "rebrand.ly/o8dcpql",
  "isCached": false,
  "hostname": "2arian3.local"
}
macbookpro@2arian3.local ~
> |
```

شکل ۴ نمایش یک درخواست به سرور توسعه داده شده و خروجی آن

^۹ Hostname

حال باید برای برنامه خود یک Dockerfile بنویسید و آن را build کنید و ایمج خود را بسازید و بر روی داکرهاب قرار دهید.

توجه: در این مرحله شما باید دو کانتینر داشته باشید. یکی مربوط به کش ردیس و دیگری مربوط به سرور خود.

راهنمایی: برای برقراری ارتباط با کش ردیس از داخل برنامه خود، باید یک network بین این دو کانتینر تعریف کنید تا کانتینرها بتوانند همدیگر را ببینند و با هم در ارتباط باشند.

می‌دانیم با پایین آمدن کانتینر ردیس، اطلاعات از دست خواهند رفت. برای جلوگیری از این اتفاق، باید اطلاعات کش ردیس را persist کنیم تا با پایین آمدن، اطلاعات از بین نروند و پس از بالا آمدن مجدد اطلاعات قبلی بازیابی شوند. برای این منظور باید از Volume استفاده کنید.

نکته مهم ۱: برای انجام این گام استفاده از docker compose مجاز نمی‌باشد.

نکته مهم ۲: پروژه باید برای فیلد های زیر کانفیگ پذیر باشد (نباید در کد به صورت hard code تعریف شوند)

- شماره پورتی که بر روی آن سرور خود را بالا می‌آورید
 - مدت زمان اعتبار لینک‌ها در کش ردیس (به دقیقه)
 - نقطه‌ی انتهایی¹⁰ مربوط به API
 - API Key (در صورتی که API نیاز به احراز هویت داشته باشد)
- متغیرهای بالا را یا به صورت فایل جداگانه کانفیگ و یا به صورت متغیرهای محیطی¹¹ تعریف کرده و بخوانید.

موارد زیر را در فایل گزارش نمایش دهید:

- دریافت ایمج ردیس و ساختن کانتینر از آن
- ساختن نتورک برای برقراری ارتباط بین دو کانتینر
- ساختن Volume جهت persist کردن اطلاعات کش
- بیلد کردن ایمج سرور نوشته شده با استفاده از داکرفایل
- ارسال ایمج ساخته شده بر روی داکرهاب و نمایش نتیجه آن
- نمایش اطلاعات ایمج سرور خود با استفاده از دستور docker inspect
- نمایش کانتینرهای موجود در سیستم خود با استفاده از دستور docker ps

¹⁰ Endpoint

¹¹ Environment Variables

- نمایش میزان منابع استفاده شده توسط کانتینر های موجود با استفاده از دستور docker stats

گام سوم

در این بخش هدف کار با یک ابزاری است که با آن می‌توانید یک کلاستر کوبرنتیز را به صورت ساده و سریع بر روی سیستم خود بالا بیاورید. این ابزار Minikube نام دارد. روش‌های مختلفی برای اجرای این ابزار پس از نصب وجود دارد. [اینجا](#) می‌توانید لیست کامل درایورهای Minikube و روش بالا آوردن آن را برای هر سیستم عامل مشاهده کنید. استفاده از هر یک از این درایورها برای بالا آوردن Minikube مجاز می‌باشد. تعدادی از درایورهای پیشنهادی عبارت‌اند از:

- VirtualBox: درایور پیش فرض Minikube می‌باشد. برای استفاده از این درایور باید VirtualBox را روی ماشین خود داشته باشید.
 - Docker: ساده‌ترین درایور برای استفاده این گزینه می‌باشد که در آن Minikube به صورت یک کانتینر داکر بالا می‌آید.
 - QEMU: با کمک این شبیه‌ساز¹²، Minikube به صورت یک ماشین مجازی بالا می‌آید. پس از راه‌اندازی Minikube شما باید برای سرور خود یک سری فایل‌های مورد نیاز جهت دیپلوی کردن آن بر روی کلاستر کوبرنتیز بنویسید.
 - فایل ConfigMap: در این فایل یک سری تنظیمات برنامه مانند آدرسی که برنامه باید به آن درخواست ارسال کند، پورت سرور، مدت زمان نگه‌داری کش و API key مشخص می‌شود (برای خوانایی بهتر این فایل، می‌توانید یک فایل کانفیگ جداگانه بنویسید و آدرس آن را به فایل ConfigMap بدهید).
 - فایل Deployment: جهت مدیریت کردن پادها می‌باشد. در این فایل تعداد replica های برنامه را برابر با ۲ قرار دهید.
 - فایل Service: این فایل جهت ایجاد دسترسی به سرور می‌باشد.
- حال که تمام فایل‌ها را ساختید آن‌ها را به همین ترتیب ساختشان، با استفاده از دستور kubectl apply بر روی Minikube اعمال کنید.

¹² Emulator

حال این ۳ فایل را برای ردیس نیز به صورت جداگانه بنویسید با این تفاوت که تعداد replica را در فایل Deployment برابر با ۱ قرار دهید. برای کش ردیس علاوه بر فایل‌های بالا نیاز به دو فایل دیگر جهت نگهداری اطلاعات دارید.

- فایل Persistent Volume: با این فایل حافظه را در کلاستر مشخص می‌کنید.
 - فایل Persistent Volume Claim: با این فایل درخواستی برای اختصاص حافظه‌ای - که با فایل قبل در کلاستر ایجاد کردید- برای یک کانتینر ایجاد می‌کنید.
- دوباره همانند قبل به ترتیب فایل‌ها را با دستور `kubectl apply` بر روی کلاستر اعمال کنید.

موارد زیر را در فایل گزارش نمایش دهید:

- با استفاده از دستور `kubectl get` صحت ایجاد منابع (کانفیگ‌مپ، دیپلویمنت، سرویس و پادها، PV، PVC) را بر روی کلاستر Minikube نشان دهید.
- همچنین آدرس IP پادها را با استفاده از endpointها نشان دهید.

گام چهارم

در این بخش می‌خواهیم صحت سیستم را تست کنیم. ابتدا ایمیجی که در گام اول (که با آن قابلیت ارسال درخواست¹³ با کمک cURL داشتید) ساخته بودید را بر روی کلاستر Minikube با استفاده از دستور `kubectl run` اجرا کنید. سپس به سرویسی که برای سرور خود ساختید درخواست بزنید.

موارد زیر را در فایل گزارش نمایش دهید:

- با استفاده از دستور `kubectl get` صحت ایجاد منابع را بر روی کلاستر Minikube نشان دهید.
- از آنجایی که تعداد replica ها را در فایل Deployment سرور خود بیش از ۱ قرار داده‌اید، با ارسال درخواست‌های مکرر به سرویس سرورتان، باید توزیع بار میان پادها را مشاهده کنید. این توزیع بار را نشان دهید (که درخواست تنها به یک پاد ارسال نمی‌شود).

¹³ Request

نکات مربوط به تحویل تمرین

- تمرین شما تحویل اسکایپی خواهد داشت. بنابراین از استفاده از کدهای یکدیگر یا کدهای موجود در وب که قادر به توضیح داده عملکرد آنها نیستید، پرهیزید!
- در تحویل اسکایپی از شما خواسته می‌شود تا با استفاده از فایل های دیپلویمنت نوشته شده، پروژه خود را از صفر بر روی کلاستر کوبرنتیز ببرید و در تعداد پادهای آن تغییر ایجاد کنید. همچنین باید بلد باشید پس از تغییر فایل کانفیگ، آن را بر روی پادها اعمال کنید.
- ابهامات خود را در سایت درس یا در گفت‌وگوی خصوصی تلگرام با تدریس‌یاران مطرح کنید و ما در سریع‌ترین زمان ممکن به آنها پاسخ خواهیم داد.

آنچه که باید ارسال کنید

- یک فایل زیپ با نام sid_HW2.zip که شامل موارد زیر است (هر مورد را در فولدر جداگانه قرار دهید)
- برای گام اول لازم است یکی از موارد زیر آپلود شود
 1. Dockerfile نوشته شده برای ساخت ایمیج مورد نظر
 2. تمامی دستوراتی که برای ساخت ایمیج مورد نظر از طریق docker commit انجام داده‌اید.
 - برای گام دوم تمامی فایل های پروژه به همراه Dockerfile آن و فایل config آن
 - برای گام سوم فایل های دیپلویمنت
 - گزارش که حداقل باید شامل موارد مطرح شده در توضیحات تمرین (به همراه اسکرین‌شات) باشد

موفق باشید

تیم درس مبانی رایانش ابری