MioBook

مقدمه

هدف از این پروژه، آشنایی با ابزار داکر (Docker) است. شما در ابتدا باید بخش بک‌اند و فرانت‌اند پروژه خود را Dockerize کرده و پس از پوش کردن آن به Docker Hub، از ابزار Docker Compose استفاده کنید تا به راحتی بک‌اند و فرانت‌اند خود را به همراه پایگاه‌داده آن بالا بیاورید.

ابزار داکر

ابزار داکر به ما این امکان را می‌دهد که برنامه‌ها را به صورت مجزا و در محیطی کاملا ایزوله راه‌اندازی کنیم. به این محیط ایزوله، کانتینر (Container) می‌گویند. کانتینرها این امکان را برای توسعه‌دهندگان فراهم می‌کنند که یک برنامه را به همراه تمامی ماژول‌ها و وابستگی‌های آن (مانند کتابخانه‌ها) در کنار هم قرار داده و در قالب یک پکیج آماده اجرا، ارائه کنند. این پکیج می‌تواند در سیستم‌های مختلف بدون نیاز به نصب مجدد نیازمندی‌ها و وابستگی‌ها اجرا شود. بنابراین با وجود کانتینر یک برنامه، دیگر نگران تنظیمات و نیازمندی‌های آن نیستیم و می‌توانیم آن را به راحتی اجرا کنیم.
در واقع، کانتینر یک نمونه اجرا شده از یک Image است. هر Image یک پکیج غیر قابل تغییر از یک برنامه است که اطلاعات و دستورات مورد نیاز برای اجرای یک کانتینر از آن را نگهداری می‌کند.
جهت ساخت یک Image برای برنامه خود، از Dockerfile استفاده می‌کنیم. Dockerfile یک اسکریپت است که دستورات آن نحوه ساخت Image ما را مشخص می‌کنند. ما در یک Dockerfile، معمولا از یک Image پایه شروع می‌کنیم و کارهای مورد نیاز جهت راه‌اندازی برنامه خود در آن را به ترتیب انجام می‌دهیم.
جهت آشنایی با داکر این لینک را مطالعه کنید. داکر را روی سیستم خود نصب کنید و با مفاهیم Image و Container آشنا شوید. همچنین مزیت‌های داکر نسبت به یک VM را بررسی کنید.

داکرایز کردن فرانت‌اند

ما تا کنون پروژه فرانت‌اند مان که یک پروژه Single Page Application با ری‌اکت بود را توسط Vite در محیط دیباگ/dev اجرا می‌کردیم. این محیط اجازه استفاده از قابلیت‌هایی مانند Hot Reload و دیباگ راحت‌تر را می‌دهد. برای اجرای پروژه در محیط production، باید در ابتدا پروژه خود را build کنیم. این کار توسط دستوری از Vite انجام می‌گیرد که در نتیجه آن، چند فایل static خواهیم داشت که تا جای ممکن بهینه شده‌اند و حجم آنها هم توسط minify/uglify شدن کاهش یافته است.

در واقع، ما به یک ارائه‌کننده فایل‌های استتیک (static file server) نیاز داریم تا در پاسخ درخواست کاربر، فایل HTML اصلی پروژه را به همراه JS و CSS-های آن به او ارسال کند. پس از اینکه کاربر در مرورگر خود فایل‌ها را دریافت کرده است، دیگر کاری با سرور استتیک فرانت‌اند ما ندارد و از آنجا که همه محتوای سایت در یک HTML جمع‌آوری شده است، صرفا در صورت نیاز به بک‌اند ریکوئست می‌زند.
بنابراین، در محیط production برنامه‌های REST، یک سرور وظیفه پاسخ‌گویی به درخواست‌ها را دارد (بک‌اند) و یک سرور فایل‌های استتیک را ارائه می‌کند (که پروژه ری‌اکت نیز جزو این دسته قرار می‌گیرد).

شما باید یک Dockerfile بنویسید که پس از کپی کردن فایل‌های پروژه، آن را build کرده و در یک سرور فایل‌ها را ارائه کند. از Nginx به عنوان سرور خود استفاده کنید.
این کار مسئله‌ای مرسوم است و می‌توانید در اینترنت به جست‌وجوی راه آن بپردازید. به طور مثال، می‌توانید از این لینک و یا این لینک استفاده کنید.
همچنین برای آشنایی با دستورات Dockerfile، این لینک را بررسی کنید. حتما یک ایده از همه دستورات اصلی آن داشته باشید. به طور مثال، تفاوت RUN، CMD و ENTRYPOINT را مطالعه کنید.
پس از نوشتن Dockerfile، شما می‌توانید با استفاده از دستور:

```
docker build [-f Dockerfile] [-t <image-name>:<tag>] .
```

از روی Dockerfile خود Image را تولید کنید و سپس با استفاده از دستور:

```
docker run --name test -d -p 80:3000 <image-name>
```

یک کانتینر از روی Image خود بالا بیاورید که پورت 3000 داخل آن به 80 در سیستم شما مپ شده است.
درباره دستورات کار با کانتینرها (از جمله لیست کردن، توقف و حذف کردن آنها) مطالعه کنید.

توجه داشته باشید که در پروژه فرانت‌اند خود، شما باید در همه fetch-ها (ارسال ریکوئست به بک‌اند) صرفا از مسیر ریکوئست (به طور مثال، /api/books) استفاده کرده باشید و آدرس بک‌اند (مثلا localhost:8000) را در آن نگذاشته باشید. ما تا کنون جهت ارسال ریکوئست به بک‌اند، از قابلیت proxy در Vite استفاده می‌کردیم. این قابلیت به این صورت کار می‌کرد که مثلا همه مسیرهایی که با /api شروع می‌شوند را به یک محل دیگر ارسال می‌کرد. این یعنی ریکوئست به سرور فرانت‌اند می‌آید و ما باید آن را به بک‌اند forward کنیم.
برای انجام این کار، از قابلیت reverse proxy بودن Nginx استفاده می‌کنیم. در کانفیگ Nginx، همه مسیرهایی که به /api می‌آیند را به آدرس بک‌اند فروارد کنید. حتما از environment variable-ها استفاده کنید و آدرس بک‌اند را مستقیم در کانفیگ قرار ندهید. با این کار می‌توانیم هنگام ساخت کانتینر از روی Image فرانت‌اند، به راحتی آدرس صحیح بک‌اند را تنظیم کنیم (docker run -e VAR=test).

داکرایز کردن بک‌اند

در این قسمت، شما برای پروژه بک‌اند خود Dockerfile می‌نویسید. خوب است که در اینجا نیز همانند فرانت‌اند، در یک مرحله ابتدا در یک base image که JDK دارد پروژه خود را build کنید، و سپس در یک base image که صرفا JRE دارد، jar ساخته شده از پروژه خود را اجرا کنید.
از آنجا که بیلد کردن پروژه ممکن است به دلیل دانلود کردن نیازمندهای آن (چه در فرانت‌اند با npm و چه در بک‌اند با maven) طول بکشد، می‌توانید پروژه خود را در سیستم خود بیلد کنید و صرفا برای تست کردن داکر، صاف فایل نهایی را به Image کپی کنید (ولی در نهایت، بیلد کردن پروژه باید به عهده Dockerfile شما باشد).
می‌توانید به عنوان یک مثال از داکرایز کردن پروژه جاوا، این لینک را مطالعه کنید.
شما احتمالا در application.properties آدرس سرور پایگاه‌داده خود را در spring.datasource.url قرار داده‌اید. از آنجا که Spring مقادیر تنظیم شده در environment variables را به مقادیر داخل فایل پراپرتیز ترجیح می‌دهد، ما می‌توانیم در هنگام ساخت کانتینر از روی Image بک‌اند، آدرس آن را به درستی تنظیم کنیم.

پوش کردن به رجیستری

در این قسمت شما باید دو Image ساخته شده برای فرانت‌اند و بک‌اند را به یک Container Image Registry عمومی ارسال کنید. Docker Hub از مشهورترین رجیستری‌ها برای آپلود و کار با Image-ها است.
برای این کار می‌توانید این لینک را مطالعه کنید.
پس از پوش شدن Image شما، همه می‌توانند آن را با استفاده از docker pull به راحتی دانلود کرده و از روی آن کانتینرهای خود را بالا بیاورند.

ابزار Compose

ابزار داکر کامپوز (Docker Compose) یک ابزار Container Orchestration است. این ابزار به ما این امکان را می‌دهد که چندین کانتینر مرتبط را به صورت هم‌زمان و هماهنگ اجرا کنیم. به طور مثال، به جای اینکه با یک دستور کانتینر بک‌اند و با یک دستور دیگر کانتینر فرانت‌اند را بسازیم، می‌توانیم با نوشتن یک فایل compose.yaml هماهنگی بین همه کانتینرهای خود را تعریف کنیم و سپس با docker compose up آنها را بالا بیاوریم.
داخل فایل متنی compose.yaml (که در نسخه‌های قدیمی‌تر docker-compose.yaml بود و با استفاده از برنامه docker-compose کار می‌کرد)، نحوه راه‌اندازی و پیکربندی هر یک از کانتینرها تعریف می‌شود. برای مثال، می‌توان مشخص کرد که هر سرویس از چه Image-ای استفاده می‌کند، چه port-هایی دارد، مقدار تنظیم شده برای environment variable-های آن چیست، ترتیب اجرای سرویس‌ها چگونه باشد و هر کدام چه network-ها و volume-هایی دارند.
برای آشنایی بیشتر با Compose، این لینک را مطالعه کنید.

با مفاهیم networking و volume-ها (دو نوع named volume و host/bind mount) در داکر آشنا شوید.
دو کانتینر که روی یک network قرار دارند، هم‌دیگر را می‌شناسند (به طور کامل از هم ایزوله نیستند) و اسم کانتینر آنها به IP آن کانتینر resolve می‌شود. بنابراین یک کانتینر می‌تواند در صورت قرار گرفتن در network یکسان، مثلا به آدرس http://some-container:8080 ریکوئست ارسال کند.
volume-ها در داکر می‌گذارند که حافظه‌ای ثابت برای یک کانتینر وجود داشته باشد. این یعنی در صورت ساخته شدن مجدد آن کانتینر، بخشی از حافظه آن حذف نشده و باقی بماند. volume-ها باید به یک فولدر در داخل کانتینر mount شوند. برای این کار یا صرفا یک named volume را mount می‌کنیم (که حافظه دائم پشت آن را خود داکر هندل می‌کند) و یا یک bind mount می‌سازیم که یک فولدر در سیستم خودمان را به یک فولدر در کانتینر مپ می‌کند.

اتصال پایگاه‌داده

در این قسمت شما باید یک compose.yaml بنویسید که کانتینر فرانت‌اند و بک‌اند شما را به همراه پایگاه‌داده بالا می‌آورد. برای پایگاه‌داده، یک کانتینر از روی base image پایگاه‌داده MySQL بالا بیاورید و با استفاده از environment variable-های Image آن، نام دیتابیس، یوزر و رمز آن را تعیین کنید. همچنین برای از بین نرفتن داده‌های پایگاه‌داده با هر بار ریستارت شدن آن، یک volume برای آن تعریف کنید.
حال شما می‌توانید از اسم کانتینر پایگاه‌داده برای مقداردهی متغیر datasource برای کانتینر بک‌اند، و از اسم کانتینر بک‌اند برای مقداردهی متغیر env خود برای فروارد کردن ریکوئست‌ها در Nginx استفاده کنید.
خوب است که با المان healthcheck در compose آشنا شوید و با تعریف آن، امکان استفاده از depends_on را داشته باشید تا کانتینرهای شما به ترتیب بالا بیایند.

ما تا اینجا از گذاشتن آدرس و رمز در Image-های خود پرهیز کردیم و آدرس‌ها را با استفاده از env به کانتینرها دادیم. از آنجا که compose.yaml فایل مهمی بوده و در ریپو پروژه production خود نیز پوش می‌شود، نباید داخل آن رمز پایگاه‌داده مستقیم آورده شود.
راه حل Compose برای مقادیر محرمانه، استفاده از secret است. یک secret محتوای یک فایل (که در ریپو ما پوش نمی‌شود و مثلا حاوی رمز پایگاه‌داده است) را به طور مستقیم به فولدر /run/secrets/ در داخل کانتینر کپی می‌کند. این کار برخلاف env-ها که با inspect کردن کانتینر مقدارشان مشخص است، فقط در صورت exec کردن به داخل کانتینر مقدارشان مشخص می‌شود. ما صرفا باید آدرس فایل را از طریق env به کانتینر و برنامه خود در داخل آن نشان دهیم. به طور مثال، کانتینر MySQL با متغیر MYSQL_PASSWORD_FILE می‌تواند رمز را از روی فایل بخواند.
از آنجا که application.properties امکان خواندن مقدار یک فیلد از روی فایل را به طور پیش‌فرض ندارد، شما باید با افزودن برخی کدها این کار را انجام دهید و رمز پایگاه‌داده را از روی secret به بک‌اند برسانید.

نکات پایانی

- این تمرین در گروه‌های حداکثر دو نفره انجام می‌شود. برای تحویل آن کافی است که یکی از اعضای گروه، لینک مخزن گیت‌هاب و Hash مربوط به آخرین کامیت پروژه را در سایت درس آپلود کند. پروژه شما بر روی این کامیت مورد ارزیابی قرار می‌گیرد.
- حتما کاربر IE-S04 را به پروژه خود اضافه کنید.
- ساختار مناسب و تمیزی کد برنامه، بخشی از نمره همه پروژه‌های شما خواهد بود. بنابراین در طراحی ساختار برنامه و همچنین خوانایی کد دقت زیادی به خرج دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده شباهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاستی که در کلاس گفته شده است کسر خواهد شد.
- سوالات خود را تا حد ممکن در گروه درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آنها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این تمرین ارتباط برقرار کنید.