

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی کامپیوتر و فناوری اطلاعات

رایانش ابری

پروژه نهایی

نام دانشجویان:

دانیال کردمدانلو

علی فرجی

**گام اول**

در این گام یک اپلیکیشن به زبان java به کمک فریم ورک spring boot و دیتابیس mysql توسعه داده شد.

این اپلیکشن، تنظیمات را به صورت فایل کانفیگ می‌گیرد.

مثلاً دستور زیر:

java -jar target/finalProject-0.1-spring-boot.jar --spring.config.name=applicationConfig --spring.config.location=file:///home/ali/appConfigs/

فایل(های) کانفیگ شامل موارد زیر می‌باشد:

app.config.expire=5

server.port=8081

spring.datasource.url=jdbc:mysql://172.17.0.2:3306/mysql

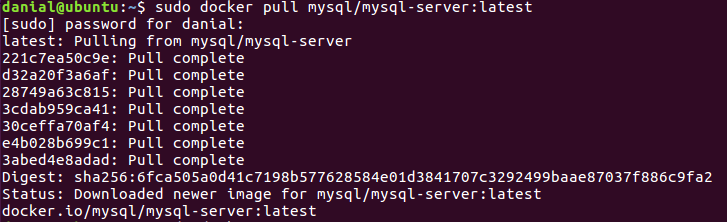
spring.datasource.username = root

spring.jpa.generate-ddl=true

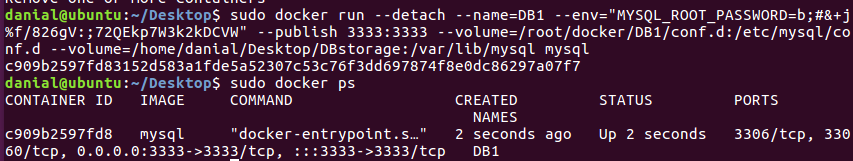
spring.datasource.password = 123456

# فاز دوم

در گام اول اقدام به ساخت container برای دیتابیس می کنیم تا در نهایت اپلیکیشن با آن ارتباط برقرار کند. به این منظور از image رسمی mysql استفاده می کنیم.



در نهایت از دستور زیر استفاده می کنیم بعد از این که image را pull کردیم.



در این جا با مشخص کردن مقادیر environment مقدار password را برای root دیتابیس مشخص می کنیم و همچنین port داخلی را به خارج برای ارتباط با دیتابیس مپ می کنیم و در نهایت هم برای persistant بودن مکان ذخیره شدن داده های دیتابیس؛ از دستور mount و حالت bind استفاده می کنیم در این حالت یک آدرس فیزیکی در داخل سیستم عامل به محیط مجازی داخلی دیتابیس مپ میشود و در نهایت داده های ذخیره شده در داخل container در محل مشخص شده ذخیره می شوند و همین امر باعص می شود در مراجعه بعدی داده ها نیز وجود داشته باشند چون در local باقی می مانند حتی اگر container را از حالت اجرا خارج کنیم و دوباره آن را run کنیم.

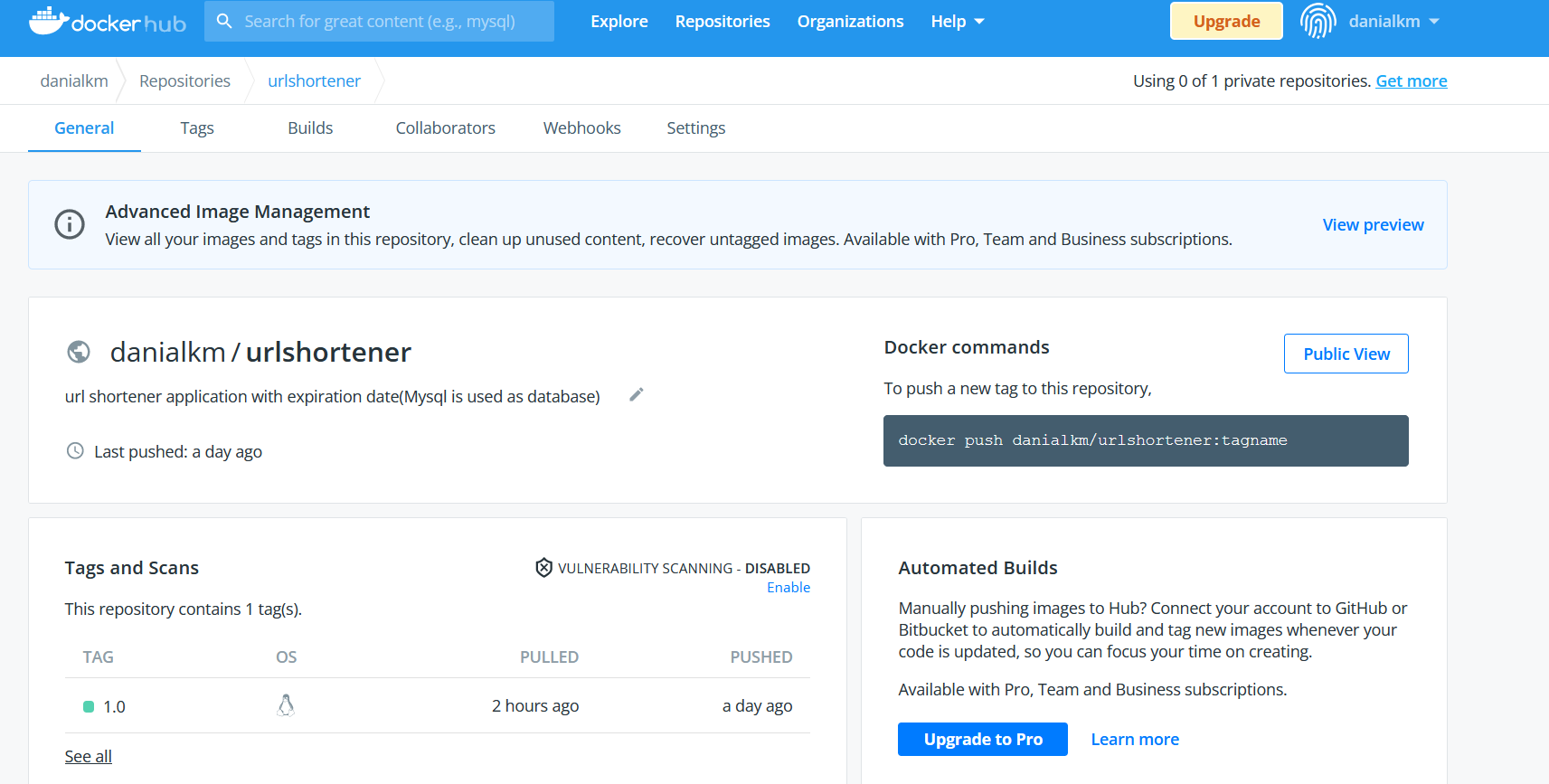
برای ساخت image خود اپلیکیشن نیز از multistage build استفاده شده است؛ به این شکل مکه در ابتدای کار یک image دارای maven و jdk را دریافت می کنیم و بعد از کپی کردن فایل های سورس به داخل کانتینر آن را بیلد می کنیم در نهایت خروجی بیلد را به کانتینر دوم که تنها jre را برای اجرای فایل دارد و سبک تر می باشد منتقل می کنیم.(داکر فایل در پیوست آمده است.)







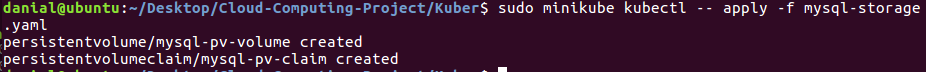




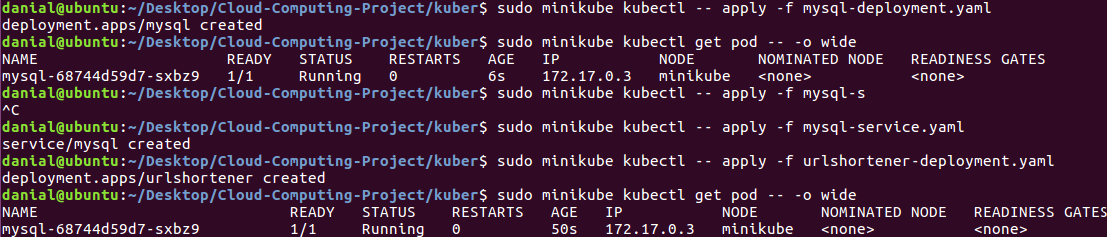
# فاز سوم

در این بخش با مشخص کردن فایل های secret.yaml و config-map.yaml به ترتیب داده های مهم مانند پسورد دیتابیس و در فایل بعدی نیز داده هایی که در اپلیکیشن استفاده می شود را مشخص می کنیم.(اپلیکیشن داده های خود را از طریق فایل های .properties می خواند و این فایل ها را باید در داخل container داشته باشد برای این منظور محتویات داخلی این فایل ها را در config-map مشخص می کنیم تا در نهایت به محیط داخلی mount شوند).

برای mysql نیاز داریم یک محیط persistant برای ذخیره سازی بسازیم که در mysql-storage.yaml این کار را انجام می دهیم و یک volume می سازیم و در نهایت در mysql-deployment تمامی موارد لازم را از جمله مقادیر environment که از فایل secret می خوانیم و همچنین فضای ساخته شده(volume) را نیز مشخص می کنیم تا داده ها را در آنجا ذخیره کند. درنهایت یک سرویس برای دسترسی به deployment ها می سازیم.



C:\Users\dania\OneDrive\Pictures\hw4\Screenshot 2022-01-24 113350.png

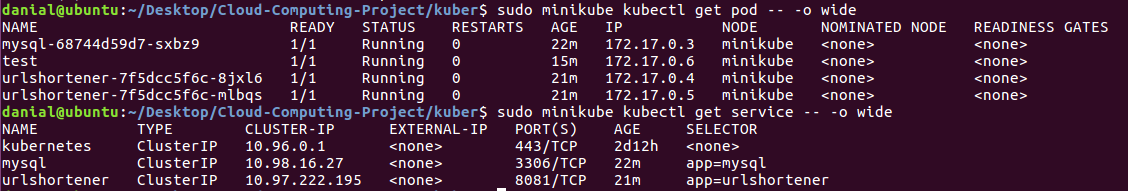


از replica به تعداد ۱ برای دیتابیس استفاده میکنیم زیرا با تنظیمات فعلی دیتابیس توانایی read و write دارد و با توجه به این سطح از دسترسی در صورت بیشتر از ۱ بودن تعداد replica در صورتی که تعداد کلاینت ها زیاد باشد و با هم دستور ادرس جدید را ارسال کنند هردو یک ایدی دریافت میکنند و در نهایت یکی از انها override می شود.

برای جلوگیری از این مشکل از تعداد رپلیکا یک استفاده می شود.

راهکار جایگزین استفاده از 2 مدل image با توانایی های متفاوت Read و write در دیتابیس می باشد. به این شکل که یک deployment با توانایی wrtite و یا Read و write داشته باشیم و در کنار آن چندین deployment با توانایی read داشته باشیم زیرا در صورت وجود چندین دستور همزمان Read مشکلی پیش نمی آید ولی برای Write این مشکل وجود دارد.

همین سیر را نیز برای پیاده سازی اپلیکیشن طی می کنیم و بعد از مشخص کردن config map مقادیر مورد نیاز را به Deployment پاس می دهیم و همچنین از image تولید شده در بخش داکر برای این قسمت استفاده می کنیم.

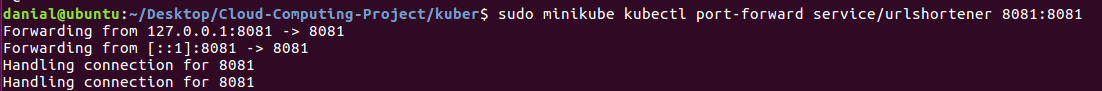


در نهایت از یک سرویس سوم دارای دستور curl استفاده می کنیم تا مجموعه را تست کنیم:



همچنین می توان از port forward استفاده کرد که در نهایت خروجی به شکل زیر خواهد بود:





در حالت porforward برای بررسی کامل نیز می توان از مرورگر برای تست اپ استفاده کرد.

# امتیازی یک) ساختن HPA برای auto scaling

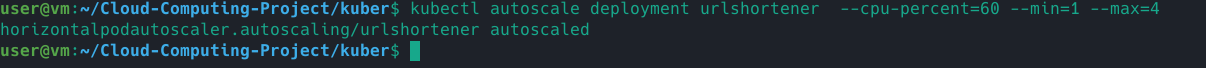
پارامترهایی که می‌توان از آن‌ها برای auto scaling استفاده کرد بسیار زیاد است.

انواع متریک‌ها شامل resource metrics، pod metrics، object metrics می‌باشند. همچنین تقریباً از هر اندازه‌گیری می‌توان برای auto scaling استفاده کرد مثلاً برای میانگین مصرف cpu می‌توان از متریک‌های خودش استفاده کرد و برای هر چیز دیگر می‌توان یک متریک External تعریف کرد و بر اساس آن auto scaling را انجام داد.

همچنین می‌توان این پارامترها را برای انجام عمل auto scaling ترکیب کرد.

ما از نوع resource metrics و مقدار میانگین مصرف CPU استفاده کرده‌ایم. زیرا میزان شلوغی سرور را نشان می‌دهد. می‌توانستیم بر روی میزان درخواست‌ها قرار دهیم ولی پیدا کردن مقدار مناسب برای میزان درخواست‌ها که به صورت آماری درست باشد، (تخمین صحیح در یک محیط واقعی برای عمل‌های کوتاه‌کردن لینک یا مشاهده لینک و تنظیم میزان درخواست) سخت است.

به کمک دستور زیر می‌توان تنظیمات auto scaling را بر اساس درصد مصرف CPU انجام داد.



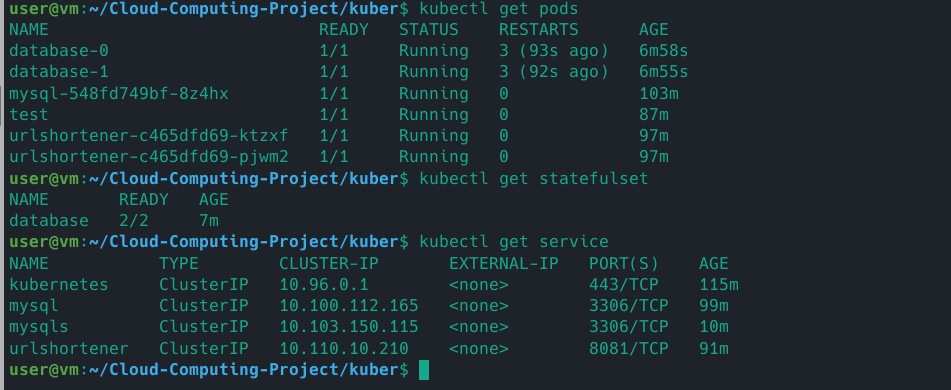
بعد از این دستور auto scaling تا نهایت ۴ پاد صورت می‌گیرد تا میانگین مصرف CPU در حد ۶۰ درصد در هر کدام نگه‌داشته شود.

# امتیازی دو) اجرای دیتابیس به کمک Stateful Set

برخی از اپلیکیشن‌ها (مثلاً دیتابیس) خودشان دارای state هستند و اگر به صورت deplyment باشند، پادها به علت به اشتراک گذاشتن یک فضا نمی‌توانند در سطح پاد دارای state باشند. برای همین از stateful set استفاده می‌کنیم تا برنامه‌ها در سطح پادها هم مثلاً دارای persistent volume جدا باشند و بتوانند برای خودشان state داشته باشند. همچنین ترتیب‌اجرا و خاتمه نیز در stateful set تضمین شده است و هر پاد هویت کاملاً مشخص دارد.

ابتدا یک فایل yaml به مانند فایل mysql-statefulset.yaml می‌نویسم که سرویس mysqls و stateful set به‌نام database را تعریف می‌کند.

بعد از apply فایل mysql-statefulset.yaml وضعیت کلاستر به صورت زیر در می آید.



# امتیازی سه) پیاده‌سازی helm chart

این ابزار از قالب پکیجینگ به نام چارت استفاده می‌کند. یک چارت در اصل مجموعه‌ای از فایل‌هایی توصیف منابع کوبرنتیز هستند.

هر chart را می‌توان گفت که یک دایرکتوری است که ساختار ویژه ای دارد.

فایل Chart.yaml اطلاعات چارت ما را داراست.

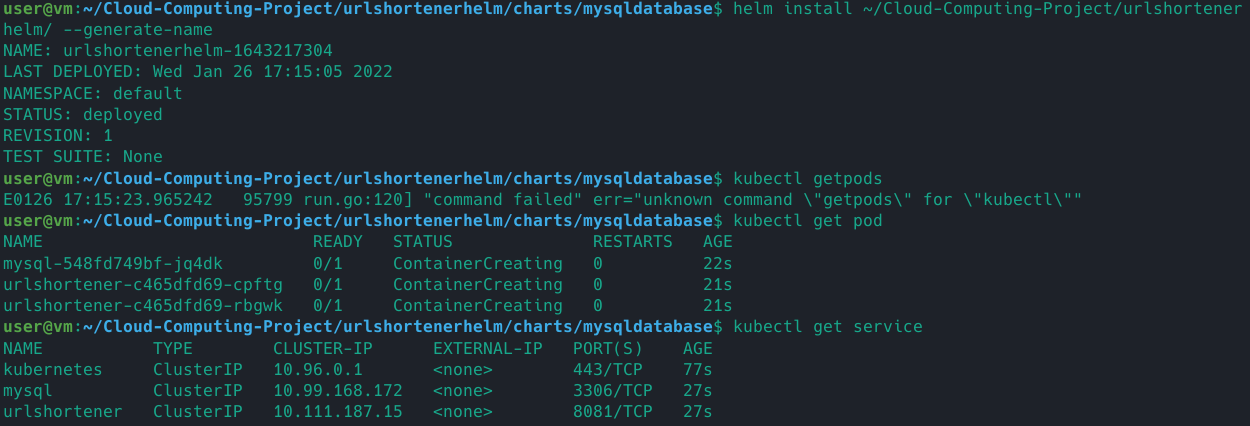
فایل values.yaml در اصل کانفیگ و تنظیمات چارت ما است که متغییرهای template نیز در اینجا تعریف می‌شوند.

پوشه chart مربوط به چارت‌هایی است که اپلیکیشن ما به آن‌ها وابسته است. مانند my sql که ما در اینجا در نظر گرفتیم و به آن subchart می‌گوییم.

پوشه templates تمامی فایل‌های yaml را نگه‌داری می‌کند و به اجازه می‌دهد که در فایل‌های yaml از متغییر ها به مانند template استفاده کنیم که این متغییرها در values.yaml

توجه کنیم که وقتی محتوای پوشه templates با مقادیر و کانفیگ‌های پوشه values.yaml ترکیب می‌شوند در اصل توصیف‌های منابع کوبرنتیز را ایجاد می‌کنند.

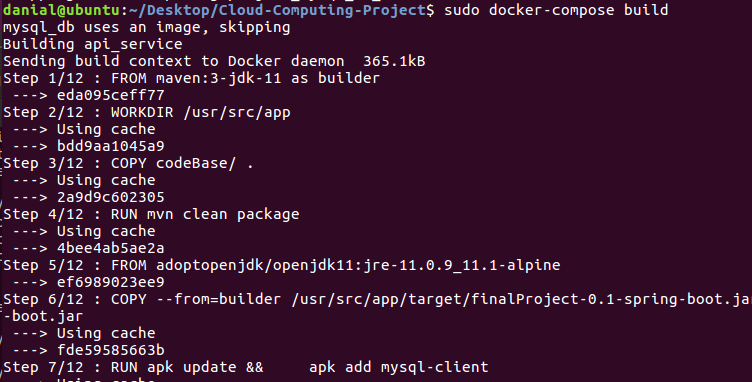
مانند زیر توسط یک دستور تمامی موراد مورد نیاز اپلیکیشن اجرا می شوند:



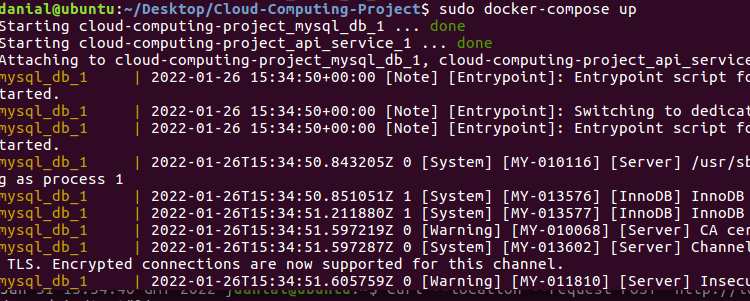
**امتیازی چهار) داکر کامپوز**

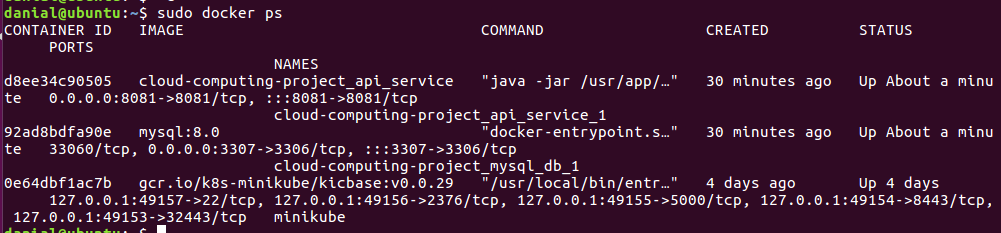
از داکر کامپوز برای اتوماتیک کردن کارها استفاده میکنیم به این شکل که با مشخص کردن داکر فایل مربوط به پروژه و همچنین مقادیر environment و mount کردن فایل ها وابستگی پروژه که دیتا بیس می باشد را نیز مشخص میکنیم و اطلاعات مربوط به دیتابیس مانند رمز و مقادیر مربوط به پورت را نیز مشخص میکنیم در نهایت داکر کامپوز اول دیتابیس را بالا می اورد تا وابستگی فراهم شود و در ادامه اپلیکیشن را بر روی کانتینر اجرا میکند. اطلاعات دیگر مانند volume و ریست اتوماتیک در صورت کرش کردن را نیز در داکر کامپوز مشخص کرده ایم.

روند اجرا و تست:



C:\Users\dania\OneDrive\Pictures\hw4\dsafas.png





برای بررسی دقیق تر کد های زده شده می توان به فایل readme در آدرس گیت زیر مراجعه کرد:

https://github.com/Danial-Kord/Cloud-Computing-Project.git