Cloud Build with cache for faster build

tiotobing - DevOps

Pada saat melakukan proses build Docker di sebuah machine untuk pertama kali tentu membutuhkan waktu yang lebih lama. Salah satu keuntungan dari menggunakan Docker adalah kita dapat menggunakan cache untuk mempercepat waktu build, karena layer-layer yang sudah tersedia dan yang dianggap tidak mendapat perubahan akan secara default tetap menggunakan cache layer yang sebelumnya.

Prerequisite:
- Memiliki akun GCP (billing enabled)
- Google Container Registry , Google Cloud Build

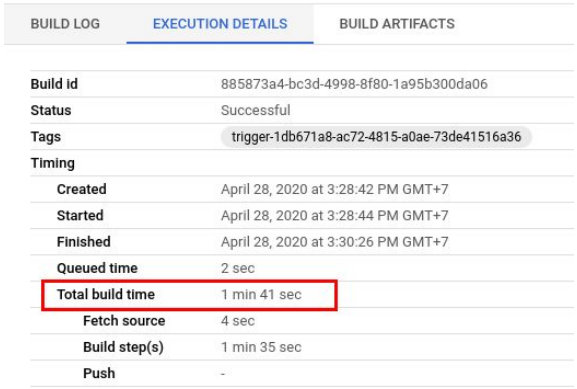
```
Build Docker image with cache": Already have image (with digest): gcr.io/cloud-builders/docker
Build Docker image with cache": Sending build context to Docker daemon  3.984MB
Build Docker image with cache": Step 1/7 : FROM keymetrics/pm2:12-alpine
Build Docker image with cache": 12-alpine: Pulling from keymetrics/pm2
Build Docker image with cache": cbdbe7a5bc2a: Already exists
Build Docker image with cache": f98f8aade4b1: Already exists
Build Docker image with cache": 6c9c37ffd044: Already exists
Build Docker image with cache": 003e8eba0035: Already exists
Build Docker image with cache": 26dad6269170: Already exists
Build Docker image with cache": Digest: sha256:c0a3f3017cff09e1c8570216a716f41969f96ddc47a828653835df69095637f6
Build Docker image with cache": Status: Downloaded newer image for keymetrics/pm2:12-alpine
Build Docker image with cache":  ---> 397617d761c1
Build Docker image with cache": Step 2/7 : WORKDIR /usr/app
Build Docker image with cache":  ---> Using cache
Build Docker image with cache":  ---> 263ea59452ef
Build Docker image with cache": Step 3/7 : COPY ./package*.json ./
Build Docker image with cache":  ---> Using cache
Build Docker image with cache":  ---> 9ba718f4d500
Build Docker image with cache": Step 4/7 : RUN npm install
Build Docker image with cache":  ---> Using cache
Build Docker image with cache":  ---> 00002bd25532
Build Docker image with cache": Step 5/7 : COPY ./ ./
Build Docker image with cache":  ---> 599c7809d95e
Build Docker image with cache": Step 6/7 : EXPOSE 3333
Build Docker image with cache":  ---> Running in 4f438c0393a4
Build Docker image with cache": Removing intermediate container 4f438c0393a4
Build Docker image with cache":  ---> f65d182a44f6
Build Docker image with cache": Step 7/7 : CMD ["pm2-runtime", "start", "ecosystem.config.js"]
Build Docker image with cache":  ---> Running in cd0cd5def3cd
Build Docker image with cache": Removing intermediate container cd0cd5def3cd
Build Docker image with cache":  ---> cc9db2d246bb
Build Docker image with cache": Successfully built cc9db2d246bb
Build Docker image with cache": Successfully tagged gcr.io/narasi-infrastructure/be_nodejs_apps:latest
```

Begitu juga di Google Cloud Build, untuk teknik cache sendiri ada dua cara (*sampai dokumentasi ini ditulis*) yang bisa diterapkan, yaitu

- Using Kaniko cache (https://cloud.google.com/cloud-build/docs/kaniko-cache)
- Using cache Docker Image

Untuk kali ini kita akan fokus membahas yang kedua, yaitu Using cache Docker image.

Sebagai acuan untuk aplikasi yang sama, sebelum menerapkan cache di cloud build, waktu yang didapatkan untuk membuild hingga rolling update adalah sebagai berikut



Pertama, perhatikan layer pada Dockerfile. Ini merupakan salah satu hal yang penting, tidak akan menjadi efisien penggunaan cache jika layer tidak tepat penempatannya.

(contoh Dockerfile)

```
Dockerfile > ...
1   #maintainer tiotobing18
2   FROM keymetrics/pm2:12-alpine
3
4   # Set working directory
5   WORKDIR /usr/app
6
7   # Copy "package.json" and "package-lock.json" before other files
8   COPY ./package*.json ./
9
10  # Install dependencies
11  RUN npm install
12
13  # Copy all files
14  COPY ./ ./
15
16  # Add Test [Jest] - To Do
17  #RUN npm test
18
19  # Expose the listening port
20  EXPOSE 3333
21
22  # Launch app with PM2
23  CMD ["pm2-runtime", "start", "ecosystem.config.js"]
24
```

Disini saya memisahkan layer install *dependencies* dengan file yang lainnya, karena pada dasarnya yang paling sering terjadi perubahan bukan di file package.json melainkan file-file lainnya. Sehingga goalnya adalah step install *dependencies* yang konon lama akan lebih cepat karena menggunakan cache.

Kedua, Cloudbuild.yaml

```
#maintainer tiotobing

steps:

- id: Download cache artifact
  name: 'gcr.io/cloud-builders/docker'
  entrypoint: 'bash'
  args:
  - '-c'
  - |
          docker pull gcr.io/$PROJECT_ID/be_nodejs_apps:latest || exit 0
- id: Build Docker image with cache
  name: 'gcr.io/cloud-builders/docker'
  args: [
           'build',
           '-t', 'gcr.io/$PROJECT_ID/be_nodejs_apps:latest',
           '--cache-from', 'gcr.io/$PROJECT_ID/be_nodejs_apps:latest',
           '.'
       ]

- id: Push Backend Apps image to GCR
  name: 'gcr.io/cloud-builders/docker'
  args: ['push', 'gcr.io/$PROJECT_ID/be_nodejs_apps']


# Step to checking run container before rolling update
- id: Rolling Update Container
  name: 'gcr.io/cloud-builders/gcloud'
  entrypoint: 'bash'
  args:
  - '-c'
  - |
          check_vm_template=$(gcloud compute instance-groups managed describe
${_VM_GRP_NAME} --zone ${_ZONE} | grep -A 3 ${_TEMPLATE_UPDATE} | grep
calculated | awk '{print $2}');
               if (($check_vm_template == 0));
                       then
                               gcloud compute instance-groups managed
```

```
rolling-action start-update ${_VM_GRP_NAME} --version template=${_TEMPLATE}
--canary-version template=${_TEMPLATE_UPDATE},target-size=3 --max-unavailable
30% --max-surge=4 --zone ${_ZONE}
                      else
                              gcloud compute instance-groups managed
rolling-action start-update ${_VM_GRP_NAME} --version
template=${_TEMPLATE_UPDATE} --canary-version
template=${_TEMPLATE},target-size=3 --max-unavailable 30% --max-surge=4 --zone
${_ZONE}
               fi

substitutions:
   _VM_GRP_NAME: 'api-staging'
   _TEMPLATE: 'staging-api-template'
   _TEMPLATE_UPDATE: 'template-backend'
   _ZONE: 'asia-southeast1-a'
```

Yang menjadi step penggunaan cache adalah

```
'--cache-from','gcr.io/$PROJECT_ID/[Nama-image]:latest',
```

Sebelum melakukan step build, cloud build akan terlebih dahulu mendownload artifact image yang sama sebelumnya dari google registry. Jadi pastikan untuk melakukan hal ini, image harus tersedia di google registry terlebih dahulu. Atau jika memang benar-benar pertama sekali melakukan build, maka waktu akan lebih lama karena belum menerapkan caching.

Hasil

BUILD LOG | EXECUTION DETAILS | BUILD ARTIFACTS

Build id	6f947d8e-47c9-468d-9cab-2975e0ec0aba
Status	Successful
Timing	
Created	May 5, 2020 at 11:30:11 AM GMT+7
Started	May 5, 2020 at 11:30:13 AM GMT+7
Finished	May 5, 2020 at 11:31:01 AM GMT+7
Queued time	1 sec
Total build time	48 sec
Fetch source	4 sec
Build step(s)	42 sec
Push	-