AWS Docker Elastic Container Registery (ECR) Update and Push

Fan Wang

2020-09-13

Contents

1 ECR Setup

Go to the **RMD**, **PDF**, or **HTML** version of this file. Go back to Python Code Examples Repository (bookdown site) or the pyfan Package (API).

1.1 Pull from Elastic Container Registry (ECR)

Given docker files already on ECR, in EC2, first, get password, then pull.

```
# log in
# copy the output from the line below and paste
aws ecr get-login --no-include-email
# this is copied from output of the command above
docker login -u AWS -p PASSWORDPASSWORDPASSWORDPASSWORD https://XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws
# pull from docker
docker pull XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fancondaXX
```

1.2 Update Elastic Container Registry (ECR)

There is a local conda file, perhaps some project repo have been updated, need to update docker file on ECR (or create new ones). Controlling EC2 can be done manually, or via SSM.

1. Start a EC2 Instance
2. Create a docker folder on EC2 instance (on remote)
3. scp update dockerfile in EC2 docker folder (local to remote)
4. build on remote server docker container (on remote)
5. push from EC2 updated docker to ECR (on remote)

First, after creating/starting a EC2 instance, create a docker file and scp update:

```
# Then a sequences of SSM calls:
# on local machine:
ssh -i "C:/Users/fan/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-pair-us_east_nv.pem" ec2-user@54.161.29

# if new instance, create a docker folder under main
# on remote machine
mkdir /home/ec2-user/docker

# ssm call to remove current dockerfile
# on remote machine
rm /home/ec2-user/docker/Dockerfile

# run local scp command to copy lateste Dockerfile to EC2, local scp generated by ec2managee
# on local machine
scp -o StrictHostKeyChecking=accept-new -i C:/Users/fan/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-pair
```

Second, start container service remotely, and build new container:

```
# start docker service on ec2
# on remote machine
sudo service docker start

# On remote machine
cd /home/ec2-user/docker
docker build -t fanconda6 --build-arg CACHE_DATE=2020-09-21-22-43-52 .
```

Third, push new container to ECR (tag, get token, login, push):

```
# Start Container Service
sudo service docker start

# CD into folder on remote
cd /home/ec2-user/docker

# tag docker
docker tag fancondaxxx XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fancondaxxx

# ECR Docker Log in
# ssm.get_authorization_token(registryIds=[boto3aws.aws_keys()['main_aws_id']])
# Decode authorization token
docker login -u AWS -p TOKENX6XXXxXXg1XXg3OXO= https://XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com

# ECR Docker Push to ECR
docker push XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fancondaxxx
```

1.3 PyFan Procedures

1. Start EC2 instance
2. Push to ECR
3. Get SSH link to EC2, SSH into EC2
4. *sudo service docker start* and *docker images* (or see pull earlier)
5. start docker image and enter to access via command line: *docker run -t -i fanconda /bin/bash*

1.4 More Example ECR Code and Outputs

1.4.1 Example Docker File for AWS

Note that that private git repos are pulled in. Note also that AWS keys are set up to allow for various access to AWS services.

```
FROM continuumio/anaconda3

VOLUME /data

# Conda update
RUN conda update conda

# https://github.com/ContinuumIO/docker-images/issues/49#issuecomment-311556456
RUN apt-get update && \
    apt-get install libgl1-mesa-glx -y

# Install Conda additional packages that i use
RUN conda install -c conda-forge interpolation
RUN conda install -c conda-forge boto3

# see https://github.com/moby/moby/issues/22832, this allows for code below to run without --no-cache
ARG CACHE_DATE=2000-01-01

# Clone our private GitHub Repository: PyFan
RUN git clone https://b123451234dfc025a836927PRIVATETOKEND1239@github.com/FanWangEcon/pyfan.git /pyfan/

# Make port 80 available to the world outside this container
EXPOSE 80

# Install software
ENV PYTHONPATH /pyfan/

ENV AWS_BUCKET_NAME=BucketName
ENV AWS_ACCESS_KEY_ID=XKIXXXGSXXXBZXX43XXX
ENV AWS_SECRET_ACCESS_KEY=xxTgp9r0f4XXXXXXX1XXlG1vTy07wydxXXXXXX11

# Run
CMD ["python", "/pyfan/pyfan/graph/exa/scatterline3.py"]
```

1.4.2 Example SSM communication

```
# aws_keys stores keys
aws_keys_dict = aws_keys()
ssm = boto3.client('ssm',
                   aws_access_key_id=aws_keys_dict['aws_access_key_id'],
                   aws_secret_access_key=aws_keys_dict['aws_secret_access_key'],
                   region_name=aws_keys_dict['region'])
commands = 'rm /home/ec2-user/docker/Dockerfile'
resp = client.send_command(
        DocumentName="AWS-RunShellScript", # One of AWS' preconfigured documents
        Parameters={'commands': commands},
        InstanceIds=[instance_id])
```

1.4.3 Outputs from Docker Build

outputs from docker build

```
json.py - jdump - 47 -  2020-09-22 16:04:32,459 - INFO list_command_invocation-cur_output
:[
    "Sending build context to Docker daemon  3.072kB\r\r",
    "Step 1/16 : FROM continuumio/anaconda3",
    " ---> 472a925c4385",
    "Step 2/16 : VOLUME /data",
    " ---> Using cache",
    " ---> cf4e6a503f00",
    "Step 3/16 : RUN conda update conda",
    " ---> Using cache",
    " ---> 542901f01365",
    "Step 4/16 : RUN apt-get update && apt-get install libgl1-mesa-glx -y",
    " ---> Using cache",
    " ---> 6672960aa00c",
    "Step 5/16 : RUN conda install -c conda-forge interpolation",
    " ---> Using cache",
    " ---> efd86a4259a4",
    "Step 6/16 : RUN conda install -c conda-forge boto3",
    " ---> Using cache",
    " ---> bd0146dac9b3",
    "Step 7/16 : ARG CACHE_DATE=2000-01-01",
    " ---> Using cache",
    " ---> dc40688e3720",
    "Step 8/16 : RUN git clone https://XXXX@github.com/FanWangEcon/pyfan.git /pyfan/",
    " ---> Running in 9c0c2a444540",
    "\u001b[91mCloning into '/pyfan'...",
    "\u001b[0mRemoving intermediate container 9c0c2a444540",
    " ---> c80480cc51a1",
    "Step 9/16 : RUN git clone https://XXXX@github.com/FanWangEcon/CondaProg.git /CondaProg/",
    " ---> Running in 07d9f665b760",
    "\u001b[91mCloning into '/CondaProg'...",
    "\u001b[0mRemoving intermediate container 07d9f665b760",
    " ---> a5ac6c6e1458",
    "Step 10/16 : EXPOSE 80",
    " ---> Running in 1a8ef516e236",
    "Removing intermediate container 1a8ef516e236",
    " ---> 13ab2965e892",
    "Step 11/16 : ENV PYTHONPATH /pyfan/",
    " ---> Running in 2d9e4b68164b",
    "Removing intermediate container 2d9e4b68164b",
    " ---> 0a74e69ce1c8",
    "Step 12/16 : ENV PYTHONPATH $PYTHONPATH:/CondaProg/",
    " ---> Running in ba59f1273f51",
    "Removing intermediate container ba59f1273f51",
    " ---> 11fd9d732e2e",
    "Step 13/16 : ENV AWS_BUCKET_NAME=BucketName",
    " ---> Running in e7a052d3eacf",
    "Removing intermediate container e7a052d3eacf",
    " ---> 5e294f562838",
    "Step 14/16 : ENV AWS_ACCESS_KEY_ID=XXXXX5GSDZSXXXX43XXX",
    " ---> Running in 60d810a8514f",
```

```
    "Removing intermediate container 60d810a8514f",
    " ---> 2fa1ac4e7d3b",
    "Step 15/16 : ENV AWS_SECRET_ACCESS_KEY=XXXXXXXXXXXXX",
    " ---> Running in 8b34126cee5d",
    "Removing intermediate container 8b34126cee5d",
    " ---> 93bd8b521d61",
    "Step 16/16 : CMD [\"python\", \"/CondaPrj/invoke/invoke.py\"]",
    " ---> Running in dd3ed44dcca7",
    "Removing intermediate container dd3ed44dcca7",
    " ---> 506f92a794cd",
    "Successfully built 506f92a794cd",
    "Successfully tagged fanconda:latest",
    "Total reclaimed space: 0B",
    ""
]
```

1.4.4 Output from Docker Push

```
json.py - jdump - 47 -  2020-09-22 16:05:32,986 - INFO list_command_invocation-cur_output
:[
    "The push refers to repository [XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda]",
    "63cc929545c3: Preparing",
    "d849f5d67bbb: Preparing",
    "f9c77b2e4c5f: Preparing",
    "7ffd6385ae0e: Preparing",
    "2fc88e09d363: Preparing",
    "50e089036495: Preparing",
    "6637031dbcc2: Preparing",
    "68d0bdfd0715: Preparing",
    "d0f104dc0a1f: Preparing",
    "50e089036495: Waiting",
    "6637031dbcc2: Waiting",
    "68d0bdfd0715: Waiting",
    "d0f104dc0a1f: Waiting",
    "2fc88e09d363: Layer already exists",
    "7ffd6385ae0e: Layer already exists",
    "50e089036495: Layer already exists",
    "6637031dbcc2: Layer already exists",
    "68d0bdfd0715: Layer already exists",
    "d0f104dc0a1f: Layer already exists",
    "d849f5d67bbb: Pushed",
    "f9c77b2e4c5f: Pushed",
    "63cc929545c3: Pushed",
    "latest: digest: sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX size: 2226",
    ""
]
```

1.4.5 AWS ECR Instructions

Under repositories listed under ECR, click on *View push command*, which shows:

```
# Retrieve an authentication token and authenticate your Docker client to your registry.
# Use the AWS CLI:
```

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin XXXX7367XX

# Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AW
# Build your Docker image using the following command. For information on building a Docker file from s
docker build -t fanconda .

# After the build completes, tag your image so you can push the image to this repository:
docker tag fanconda:latest XXXX7367XXXX.dkr.ecr.us-east-1.amazonaws.com/fanconda:latest
```

1.5 Command Line Sequence Gathered

Gathered sequence of command line operations:

```
ssh -i "G:/repos/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-pair-us_east_nv.pem" ec2-user@34.229.39.138

docker run -t -i fanconda /bin/bash

scp -o StrictHostKeyChecking=accept-new -i G:/repos/CondaPrj/boto3aws/aws_ec2/pem/fan_wang-key-pair-us_

sudo service docker start

cd /home/ec2-user/docker
docker build -t fanconda --build-arg CACHE_DATE=2020-12-22-10-58-57 .
docker system prune --force

cd /home/ec2-user/docker
docker tag fanconda 710673677961.dkr.ecr.us-east-1.amazonaws.com/fanconda

aws ecr get-login --no-include-email
docker login -u AWS -p XXXXX= https://710673677961.dkr.ecr.us-east-1.amazonaws.com

docker push 710673677961.dkr.ecr.us-east-1.amazonaws.com/fanconda
```