1 - Development environment

The development's docker-compose file is structured as follows:

```
version: "3.9"
services:
  app_dev:
    container_name: app_dev
    restart: always
    build:
      context: ./container_app/
      dockerfile: Dockerfile.dev
    image: app_dev:1.0
  nginx_dev:
    environment:
     - NGINX_SERVER_ADDRESS=$NGINX_SERVER_ADDRESS
     - NGINX_LOCATION=$NGINX_LOCATION
     - NGINX_LOCATION_PORT=$NGINX_LOCATION_PORT
    container_name: nginx_dev
    restart: always
    build:
      context: ./container_app/nginx/
      dockerfile: Dockerfile
    image: nginx_dev:1.0
    ports:
      - "5000:80"
    depends_on:
      - app_dev
```

Image building is integrated into "build" section in order to reduce "docker build" commands.
Environment's variables are stored into .env.[environment_chain]:

```
[sga@sga-work Test_Project]$ ls -la
total 36
drwxrwxr-x.  4 sga sga 4096 Dec 12 22:30 .
drwx------. 44 sga sga 4096 Dec 12 03:02 ..
drwxrwxr-x.  4 sga sga 4096 Dec 11 18:23 container_app
-rw-rw-r--.  1 sga sga  565 Dec 12 01:25 docker-compose-dev.yml
-rw-rw-r--.  1 sga sga  573 Dec 11 17:23 docker-compose-prod.yml
-rw-rw-r--.  1 sga sga   84 Dec 10 16:05 .env.dev
-rw-rw-r--.  1 sga sga   92 Dec 11 17:19 .env.prod
-rw-rw-r--.  1 sga sga 2356 Dec 12 03:01 .gitlab-ci.yml
drwxrwxr-x.  3 sga sga 4096 Dec 12 02:55 terraform
```

.env.dev contains these values:

```
NGINX_SERVER_ADDRESS=docker_app_dev
NGINX_LOCATION=app_dev
NGINX_LOCATION_PORT=5000
```

Inside container_app/nginx/conf/, there's a nginx default.conf template where variables will be substituted with containter_app/nginx/nginx-variables_setup.sh script.
The mentioned script is the Dockerfile's entrypoint.

```
#!/usr/bin/env sh
set -eu

envsubst '${NGINX_SERVER_ADDRESS} ${NGINX_LOCATION} ${NGINX_LOCATION_PORT}' < /etc/nginx/conf.d/default.conf.template > /etc/nginx/conf.d/default.conf

exec "$@"
```

container_app/nginx/Dockerfile:

```
FROM nginx:alpine

RUN rm /etc/nginx/nginx.conf
COPY conf/nginx.conf /etc/nginx/
RUN rm /etc/nginx/conf.d/default.conf
COPY conf/default.conf.template /etc/nginx/conf.d/
COPY nginx-variables_setup.sh /

ENTRYPOINT ["/nginx-variables_setup.sh"]
CMD ["nginx", "-g", "daemon off;"]
```

Let's bring the docker-compose up with "docker-compose -f docker-compose-dev.yml --env-file .env.dev up --build" command:

```
Successfully built 3a1f1500dccd
Successfully tagged nginx_dev:1.0
Creating app_dev ... done
Creating nginx_dev ... done
Attaching to app_dev, nginx_dev
app_dev       | Adding tag: testing
app_dev       | Adding tag: calculation
app_dev       | Loading swagger docs for function: blueprint_x.test
app_dev       | Loading swagger docs for function: blueprint_x.plus_x
app_dev       | Loading swagger docs for function: blueprint_y.test
app_dev       | Loading swagger docs for function: blueprint_y.minus_y
app_dev       |  * Serving Flask app 'src.app' (lazy loading)
app_dev       |  * Environment: production
app_dev       |    WARNING: This is a development server. Do not use it in a production deployment.
app_dev       |    Use a production WSGI server instead.
app_dev       |  * Debug mode: on
app_dev       |  * Running on all addresses.
app_dev       |    WARNING: This is a development server. Do not use it in a production deployment.
app_dev       |  * Running on http://192.168.32.2:5000/ (Press CTRL+C to quit)
app_dev       |  * Restarting with stat
app_dev       |  * Debugger is active!
app_dev       |  * Debugger PIN: 109-090-055
```

The default.conf is correctly filled:

```
[sga@sga-work Test_Project]$ docker exec $(docker ps | grep nginx_dev | awk '{print $1}') cat /etc/nginx/conf.d/default.conf
server {

    listen 80;
    server_name docker_app_dev;

    location / {
        proxy_pass http://app_dev:5000;

        # Do not change this
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /static {
        rewrite ^/static(.*) /$1 break;
        root /static;
    }
}
```

Executing a curl to test it:

```
app_dev       |    WARNING: This is a development server. Do not use it in a production deployment.
app_dev       |  * Running on http://192.168.80.2:5000/ (Press CTRL+C to quit)
app_dev       |  * Restarting with stat
app_dev       |  * Debugger is active!
app_dev       |  * Debugger PIN: 141-213-424
app_dev       | 192.168.80.3 - - [12/Dec/2021 21:56:40] "GET /api/v1/path_for_blueprint_x/test HTTP/1.0" 200 -
nginx_dev     | 192.168.80.1 - - [12/Dec/2021:21:56:40 +0000] "GET /api/v1/path_for_blueprint_x/test HTTP/1.1" 200 55 "-" "curl/7.76.1" "-"
```

sga@sga-work:~ 238x28

```
[sga@sga-work ~]$ date; curl localhost:5000/api/v1/path_for_blueprint_x/test
Sun 12 Dec 22:56:40 CET 2021
{
  "msg": "I'm the test endpoint from blueprint_x."
}
[sga@sga-work ~]$
```

Executing “pytest” command:

```
app_dev       |  * Debugger PIN: 141-213-424
app_dev       | 192.168.80.3 - - [12/Dec/2021 21:56:40] "GET /api/v1/path_for_blueprint_x/test HTTP/1.0" 200 -
nginx_dev     | 192.168.80.1 - - [12/Dec/2021:21:56:40 +0000] "GET /api/v1/path_for_blueprint_x/test HTTP/1.1" 200 55 "-" "curl/7.76.1" "-"
nginx_dev     | 192.168.80.1 - - [12/Dec/2021:21:57:53 +0000] "GET /api/v1/path_for_blueprint_x/test HTTP/1.1" 200 55 "-" "python-requests/2.25.1" "-"
app_dev       | 192.168.80.3 - - [12/Dec/2021 21:57:53] "GET /api/v1/path_for_blueprint_x/test HTTP/1.0" 200 -
app_dev       | 192.168.80.3 - - [12/Dec/2021 21:57:53] "GET /api/v1/path_for_blueprint_y/test HTTP/1.0" 200 -
nginx_dev     | 192.168.80.1 - - [12/Dec/2021:21:57:53 +0000] "GET /api/v1/path_for_blueprint_y/test HTTP/1.1" 200 55 "-" "python-requests/2.25.1" "-"
app_dev       | 192.168.80.3 - - [12/Dec/2021 21:57:53] "POST /api/v1/path_for_blueprint_x/plus HTTP/1.0" 200 -
nginx_dev     | 192.168.80.1 - - [12/Dec/2021:21:57:53 +0000] "POST /api/v1/path_for_blueprint_x/plus HTTP/1.1" 200 36 "-" "python-requests/2.25.1" "-"
app_dev       | 192.168.80.3 - - [12/Dec/2021 21:57:53] "POST /api/v1/path_for_blueprint_y/minus HTTP/1.0" 200 -
nginx_dev     | 192.168.80.1 - - [12/Dec/2021:21:57:53 +0000] "POST /api/v1/path_for_blueprint_y/minus HTTP/1.1" 200 35 "-" "python-requests/2.25.1" "-"
app_dev       | 192.168.80.3 - - [12/Dec/2021 21:57:53] "GET /api/swagger.json HTTP/1.0" 200 -
nginx_dev     | 192.168.80.1 - - [12/Dec/2021:21:57:53 +0000] "GET /api/swagger.json HTTP/1.1" 200 3234 "-" "python-requests/2.25.1" "-"
```

sga@sga-work:~/Test_Project 238x28

```
[sga@sga-work Test_Project]$ date; pytest
Sun 12 Dec 22:57:52 CET 2021
============================= test session starts ==============================
platform linux -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: /home/sga/Test_Project
plugins: testinfra-2.1.0
collected 5 items

container_app/minimal-flask-example/test/test_endpoints.py .....

============================== 5 passed in 0.58s ===============================
[sga@sga-work Test_Project]$
```

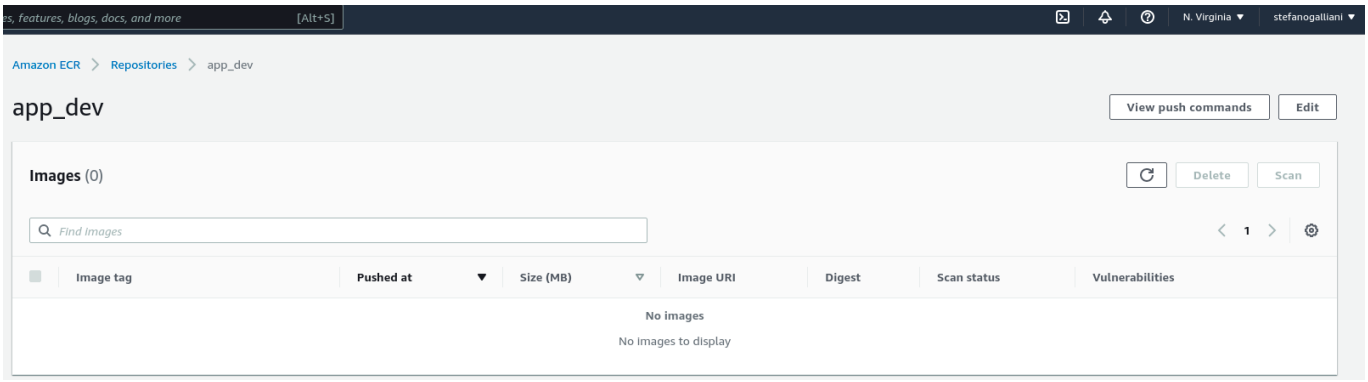
2 - Build the infrastructure

Inside ./terraform directory, there are credentials and ecr_aws.tf files.
aws_access_key_id and aws_secret_access_key are needed to perform ECR creation.
To test this part, I used my AWS account:

```
[sga@sga-work terraform]$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_ecr_repository.app_dev will be created
  + resource "aws_ecr_repository" "app_dev" {
      + arn                 = (known after apply)
      + id                  = (known after apply)
      + image_tag_mutability = "MUTABLE"
      + name                = "app_dev"
      + registry_id         = (known after apply)
      + repository_url      = (known after apply)
      + tags_all            = (known after apply)

      + image_scanning_configuration {
          + scan_on_push = true
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run
"terraform apply" now.
[sga@sga-work terraform]$
```

```
[sga@sga-work terraform]$ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_ecr_repository.app_dev will be created
  + resource "aws_ecr_repository" "app_dev" {
      + arn                 = (known after apply)
      + id                  = (known after apply)
      + image_tag_mutability = "MUTABLE"
      + name                = "app_dev"
      + registry_id         = (known after apply)
      + repository_url      = (known after apply)
      + tags_all            = (known after apply)

      + image_scanning_configuration {
          + scan_on_push = true
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_ecr_repository.app_dev: Creating...
aws_ecr_repository.app_dev: Creation complete after 2s [id=app_dev]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

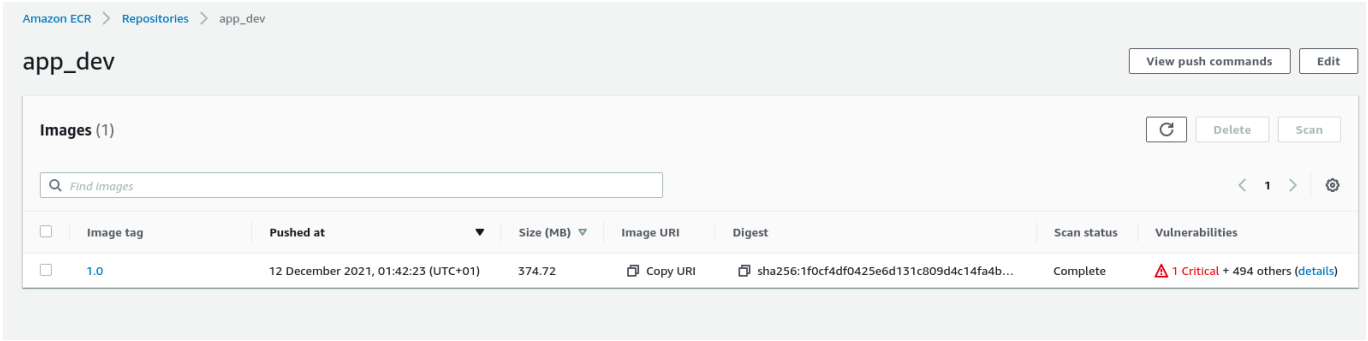
The ECR is now available:



After login via “docker login -u AWS -p $(aws ecr get-login-password --region ${REGION}) ${AWS_ACCESS_ID}.dkr.ecr.${REGION}.amazonaws.com”, it’s possible pushing images to the repo.
First of all, it’s necessary to tag the image with “docker tag app_dev:1.0 094944678802.dkr.ecr.us-east-1.amazonaws.com/app_dev:1.0”, and then push with “docker push 094944678802.dkr.ecr.us-east-1.amazonaws.com/app_dev:1.0”:

```
REPOSITORY                                          TAG         IMAGE ID       CREATED         SIZE
app_dev                                             1.0         f19acc46f61c   3 hours ago     964MB
nginx_dev                                           1.0         3a1f1500dccd   24 hours ago    23.2MB
094944678802.dkr.ecr.us-east-1.amazonaws.com/app_dev   1.0         eb820542a6c2   24 hours ago    964MB
```

```
[sga@sga-work tmp]$ docker tag app_dev:1.0 094944678802.dkr.ecr.us-east-1.amazonaws.com/app_dev:1.0
[sga@sga-work tmp]$ docker push 094944678802.dkr.ecr.us-east-1.amazonaws.com/app_dev:1.0
The push refers to repository [094944678802.dkr.ecr.us-east-1.amazonaws.com/app_dev]
d04898d14735: Pushed
35fc1283b81f: Pushed
3d89e9081968: Pushed
fb755fd0b991: Pushed
ef4a80903153: Pushed
4a5af681fba9: Pushed
8824d89eb00f: Pushed
dfbe6af77ee5: Pushed
b6c95c4dbecc: Pushed
0406c96975d9: Pushed
ed21f2ae1bcb: Pushed
bbd3863206de: Pushed
e8b96c07c2b4: Pushed
eb2b42f7b3cd: Pushed
1c4ace49ea7c: Pushed
d8ca8110db62: Pushed
1a0f5736b51a: Pushed
5355a5449f5a: Pushed
b000eda63315: Pushed
187887a99fc8: Pushed
7b728db2e54c: Pushed
ab1b5f2e7168: Pushed
5ee5ac7bcf33: Pushed
c2599a11c10e: Pushed
12491298a4e6: Pushed
a42ee6a06438: Pushed
42630bf6e3a1: Pushed
9e28eabfde8b: Pushing [==========>                                        ]  106.3MB/510.1MB
52ed97b6b9c6: Pushing [========================================>          ]  116.3MB/145.5MB
9e28eabfde8b: Pushing [===============>                                   ]  158.9MB/510.1MB
52ed97b6b9c6: Pushed
42aff4deb538: Pushed
52ed97b6b9c6: Pushing  117.3MB/145.5MB
d6a325d281f2: Pushing [==================================>                ]  83.78MB/114.1MB
```



By adding some policies, the file terraform/ecr_aws.tf might be:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "3.68.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"
  shared_credentials_file = "./credentials"
}

resource "aws_ecr_repository" "app_dev" {
  name                 = "app_dev"
```

```
  image_scanning_configuration {
    scan_on_push = true
  }
}

resource "aws_ecr_repository_policy" "app_dev_policy" {
  repository = aws_ecr_repository.app_dev.app_dev

  policy = <<EOF
{
    "Version": "2021-12-13",
    "Statement": [
        {
            "Sid": "new policy",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:ListImages",
            ]
        }
    ]
}
EOF
}
```

3 – CI/CD

CI/CD configuration file is located into ./.gitlab-ci.yml.
The stages are organized like that:
- prereq: install all the packages needed.
- build: clone the repo and bring up the development docker-compose.
- test: it runs the tests described inside the repo documentation.
- merge: login to AWS ECR repo via docker and push the development images. This also allows you to keep track of development images.
- deploy-test: bring up production docker-compose to test it before merging.
- deploy-merge: push production images into AWS ECR.
- deploy: initialize a docker swarm with a single node and deploy the stack by pulling the images directly from AWS ECR.