Выполнение домашнего задания для DevOps-тренировок в Яндексе, лекция "Облако. Кто виноват и что делать."

Описание задания и все ссылки есть у меня в репозитории на github

Для начала нужно создать две виртуальные машины на Яндекс облаке и клонировать себе в каждую машину репозиторий с гитхаба. В рамках проекта тренировок предоставляется возможность бесплатно использовать виртуальные машины по запросу.

Машины созданы

Подключаемся удобным способом к ним по SSH, я использую Windows Terminal. Клонируем репозиторий

```
git clone https://github.com/igkfellk/y-y-devops-trainings-cloud-1/
```

Сборка образа для catGPT

Создаем каталог

```
sudo mkdir /etc/docker/catGPT
```

Копируем нужные файлы в созданный каталог

```
sudo cp -r /home/ikfellk/y-y-devops-trainings-cloud-1/catgpt/.
/etc/docker/catGPT/
```

Переходим в каталог

```
cd /etc/docker/catGPT
```

Создаем Dockerfile

```
sudo nano Dockerfile
```

```
# Используем образ Golang рекомендуемый в задании
FROM golang:1.21 AS builder

# Указываем автора файла
MAINTAINER Igor Kucherenko <i.kucherenko@test.ru>

# Устанавливаем и переходим в рабочую директорию
WORKDIR /go/src/app
```

```
# Копируем go.mod и go.sum для загрузки зависимостей
COPY go.mod go.sum ./

# Загружаем зависимости
RUN go mod download

# Копируем исходный код приложения
COPY . .

# Собираем приложение
RUN CGO_ENABLED=0 go build -o /go/bin/app
RUN chmod +x /go/bin/app

# Создаем runtime образ на основе gcr.io/distroless/static-debian12:latest-
amd64
FROM gcr.io/distroless/static-debian12:latest-amd64

# Копируем собранный бинарный файл из builder образа
COPY --from=builder /go/bin/app /go/bin/app

# Определяем команду, которая будет запускаться при запуске контейнера
CMD ["/go/bin/app"]
```

Запуск сборки

```
sudo docker build -t ikfellk/catgpt:v1 .
```

Запуск контейнера

```
docker run -d -p 8080:8080 ikfellk/catgpt:v1
```

Тестируем

Для проверки созданных образов выполняем команду

```
docker images
```

и получаем подобный вывод

Для проверки созданных контейнеров выполняем команду

```
docker ps -a
```

и получаем подобный вывод

идем в браузер и открываем страницу

```
http://ваш_ip:8080/
```

Получаем такой вывод

Коннектимся к яндекс облаку через Oauth

Для полноценного выполнения задания еще потребуется настроить связь через oauth яндекса с яндекс облаком. Тут ничего сложного, нужно получить токен через свой аккаунт яндекса перейдя по ссылке Далее в ранее созданной виртуальной машине или машинах выполнить подключение в несколько этапов:

- Скачать и установить яндекс CLI

```
curl -sSL https://storage.yandexcloud.net/yandexcloud-yc/install.sh | bash
```

- Выполнить настройку подключения через созданный токен

```
yc init
```

Создаем реестр

Выполнив одну команду, получаем созданный реестр

```
yc container registry create --name registry-ikfellk
```

Где:

- **registry-ikfellk** - любое имя на ваше усмотрение

Загружаем образ в реестр

Для начала смотрим на образы

```
docker images
```

Выбираем нужный нам образ для загрузки и присваиваем тег яндекса

```
docker tag ikfellk/catgpt:v1 \
```

```
cr.yandex/crp3b70qbcrknbg22qbk/ikfellkcatgpt:v1
```

Где:

- **crp3b70qbcrknbg22qbk** - идентификатор вашего реестра на яндекс облаке

Загружаем образ в реестр

```
docker push cr.yandex/crp3b70qbcrknbg22qbk/ikfellkcatgpt:v1
```

Проверяем что образ загрузился командой

```
yc container image list
```

И получаем такую картину

Автосборка с помощью Github CI и отправка в Яндекс облако

Для начала нужно создать ключи доступа к яндекс облаку из Github

```
yc iam access-key create --service-account-name ikfellk
```

Где:

- **ikfellk** - имя вашего аккаунта в яндекс облаке

Получаем такой вывод

Создаем токен авторизации

```
yc iam create-token
```

Создаем в репозитории на гитхабе файл

`.github/workflows/docker-build-catgpt.yml`

```
name: Docker Build and Deploy catgpt

on:
  push:
    tags:
   - '*'  # Любой тег

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2
      - name: Login to Yandex
        run: docker login -u ${{ secrets.YANDEX_USERNAME }} -p ${{
secrets.YANDEX_CLOUD_SERVICE_ACCOUNT_KEY }}  cr.yandex/crj3b74hbhretbg25qbk
      - name: Build and push Docker image
        run: |
          docker build -t cr.yandex/crj3b74hbhretbg25qbk/catgpt:v1 ./catgpt
          docker push cr.yandex/crj3b74hbhretbg25qbk/catgpt:v1
```

Где:

- **YANDEX_USERNAME** - логин от яндекс облака
- **YANDEX_CLOUD_SERVICE_ACCOUNT_KEY** - Токен авторизации созданный ранее
- **crj3b74hbhretbg25qbk** - Идентификатор вашего реестра на яндекс облаке

Создаем секреты описанные в предыдущем шаге для создания вы должны находится в своем репозитории и нажать кнопку **Settings**

Далее перейти в **Secrets and variables** и далее в **Actions**

Теперь создаем секреты для авторизации Нужно нажать **New repository secret** Ввести названия что упоминались выше или можете посмотреть свой файл **docker-build-catgpt.yml** и там найти строки с секретами и изменить на ваши названия, которые вы создали

Важное напоминание

У вас должен лежать Dockerfile в каталоге, который использует команда docker build.

В случае данного задания у меня лежит файл в каталоге **./catgpt**

Тестируем автосборку и отправку в Яндекс облако

В консоли переходим в каталог клонированной ветки репозитория гитхаба

```
cd y-y-devops-trainings-cloud-1/
```

Выполняем команды

```
git pull
```

```
git tag -a v1.2.4 -m "Тест автосборки"
```

```
git push origin v1.2.4
```

Вводим логин и пароль от гитхаба и получаем такой вывод

Переходим в браузере в данный репозиторий во складку **Actions** И видим что процесс автосборки пошел

Буквально через несколько минут там же вы увидите такую картину

Процесс сборки завершился успешно

Теперь переходим в свое яндекс облако и проверяем действительно ли там появился образ

Вуаля и он действительно там

Автосборка с помощью Github CI и отправка в Docker Hub

Для начала нужно создать токен авторизации на docker hub:

- Переходим на сайт
- Логинимся под своей учеткой
- Переходим в **Account Settings**

- Далее **Security**
- Нажимаем **New Access Token**
- Даем ему имя и права
- Получаем готовый токен

Токен копируем и не теряем, он пригодится для создания секретов на гитхабе Создаем в репозитории на гитхабе файл

```
.github/workflows/docker-publish.yml
```

```
name: Publish Docker image

on:
  push:
    tags:
   - '*'  # Любой тег

jobs:
  push_to_registry:
    name: Push Docker image to Docker Hub
    runs-on: ubuntu-latest
    steps:
      - name: Check out the repo
        uses: actions/checkout@v4
      - name: Log in to Docker Hub
        uses: docker/login-action@f4ef78c080cd8ba55a85445d5b36e214a81df20a
        with:
          username: ${{ secrets.USERNAME }}
          password: ${{ secrets.PASSWORD }}
      - name: Extract metadata (tags, labels) for Docker
        id: meta
        uses: docker/metadata-
action@9ec57ed1fcdbf14dcef7dfbe97b2010124a938b7
        with:
          images: ikfellk/catgpt
      - name: Build and push Docker image
        uses: docker/build-push-
action@3b5e8027fcad23fda98b2e3ac259d8d67585f671
        with:
          context: ./catgpt
          file: ./Dockerfile
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

Где:

- **USERNAME** - логин от docker hub

- **PASSWORD** - Токен авторизации созданный ранее

Создаем секреты описанные в предыдущем шаге для создания вы должны находится в своем репозитории и нажать кнопку **Settings**

Далее перейти в **Secrets and variables** и далее в **Actions**

Теперь создаем секреты для авторизации Нужно нажать **New repository secret** Ввести названия что упоминались выше или можете посмотреть свой файл **docker-publish.yml** и там найти строки с секретами и изменить на ваши названия, которые вы создали

Важное напоминание У вас должен лежать Dockerfile в корневом каталоге

Тестируем автосборку и отправку в Docker Hub

В консоли переходим в каталог клонированной ветки репозитория гитхаба

```
cd y-y-devops-trainings-cloud-1/
```

Выполняем команды

```
git pull
```

```
git tag -a v1.2.4 -m "Тест автосборки"
```

```
git push origin v1.2.4
```

Получаем выполненный процесс как и в предыдущем тестировании Переходим в свой репозиторий на **Docker hub** и убеждаемся что файлы попали туда

Разворачиваем инфраструктуру с помощью Terraform

Установка

Для начала нужно установить Terraform, а т.к. мы в России, то делать будем это через

репозиторий яндекса Заходим на сайт **https://hashicorp-releases.yandexcloud.net/terraform/** и копируем ссылку последней версии Выполняем команды: Скачать архив

```
wget
https://hashicorp-releases.yandexcloud.net/terraform/1.6.3/terraform_1.6.3_linux_amd64.zip
```

Распаковать архив

```
unzip terraform_1.6.3_linux_amd64.zip
```

Переместить файл

```
sudo mv terraform /usr/local/bin
```

Проверка работы Terraform

Переходим в заранее клонированный репозиторий с github в папку terraform И выполняем команду

```
terraform init
```

Если получаете ошибку такого рода

То нужно выполнить следующее: Создать файл

```
nano ~/.terraformrc
```

```
provider_installation {
  network_mirror {
    url = "https://terraform-mirror.yandexcloud.net/"
    include = ["registry.terraform.io/*/*"]
  }
  direct {
    exclude = ["registry.terraform.io/*/*"]
  }
}
```

После этого пробуем еще раз команду

```
terraform init
```

И получаем положительный результат

Запуск

> У вас заранее должны быть созданы два сервисных аккаунта в яндекс облаке с нужными правами, описанными в **main.tf**

Для начала нужно выполнить команду

```
terraform plan
```

Данная команда покажет ошибки в вашем main.tf если таковы имеются. В моем случае ошибки есть и их нужно исправить

Разбор ошибок и устранение

1) Ошибка в названии регистра

> Error: Reference to undeclared resource on output.tf line 2, in output «container_registry_id»: 2: value = yandex_container_registry.registry1.id A managed resource «yandex_container_registry» «registry1» has not been declared in the root module.

Тут все просто, в файле **output.tf** не верно указано название регистра **registry1** В моем случае оно называется **ikfellk-registry** Просто меняю на корректное название и ошибка уходит

2) Отсутствует json файл с данными

> Error: Invalid SA Key with provider[«registry.terraform.io/yandex-cloud/yandex»], on main.tf line 11, in provider «yandex»: 11: service_account_key_file = «./tf_key.json» JSON in «./tf_key.json» are not valid: invalid character '.' looking for beginning of value Error: JSON in «./tf_key.json» are not valid: invalid character '.' looking for beginning of value with provider[«registry.terraform.io/yandex-cloud/yandex»], on main.tf line 11, in provider «yandex»: 11: service_account_key_file = «./tf_key.json»

Для решения нужно сгенерировать json файл с данными для входа Мы это уже делали ранее тут Теперь выполняем команду

```
terraform plan
```

Видим что ошибки ушли, но появилась еще одна

3) Отсутствует SSH ключ для авторизации

> Error: Invalid function argument on main.tf line 119, in resource «yandex_compute_instance_group» «catgpt»: 119: ssh-keys = «ubuntu:${file(»~/.ssh/devops_training.pub«)}» while calling file(path) Invalid value for «path» parameter: no file exists at «~/.ssh/devops_training.pub»; this function works only with files that are distributed as part of the configuration source code, so if this file will be created by a resource in this configuration you must instead obtain this result from an attribute of that resource.

Для решения нужно сгенерировать SSH ключ

```
ssh-keygen -t rsa
```

Указываем путь и название как в ошибке **/root/.ssh/devops_training** Вводим пароль Проверяем полученные сертификаты

```
~/.ssh/
```

Теперь выполняем команду

```
terraform plan
```

И получаем полностью корректный план. Небольшой пример как это выглядит

После этого можно запускать команду, которая настроит все необходимые ресурсы в автоматическом режиме

```
terraform apply
```

В конце, как отработает команда, вы получите такой вывод

Тестируем отработанный Terraform

Переходим в свой личный кабинет яндекс облака и смотрим что там появились машины, сети, балансировщик и регистр

Теперь нужно загрузить актуальную версию приложения в контейнер Для этого выполняем уже знакомые нам команды

```
git pull
```

```
git tag -a v1.2.5 -m "Тест автосборки"
```

```
git push origin v1.2.5
```

Вводим логин и пароль от гитхаба Спустя примерно минуту или полторы, смотрим в личный кабинет яндекс облака и видим там контейнер

Теперь переходим в веб браузер по ip адресам виртуальный машин созданный нашим terraform файлом и видим там рабочее приложение и метрики

Для проверки балансировки нужно отключить одну виртуальную машину и со второй ничего не должно произойти и приложение так же останется в рабочем состоянии

Все работает как нужно.

Задание выполнено.