# Report B7 HW03

## Что нужно сделать

1. Создать ВМ в Я.Облаке (минимальная конфигурация: ***2vCPU***, ***2GB RAM***, ***20GB HDD***).
2. Установить и запустить на ВМ ***Docker***.
3. Установить и запустить на ВМ СУБД ***Postgres***.
4. В ***Postgres*** создать БД и пользователя с произвольным именем на ваш выбор и дать этому пользователю полные права на созданную БД.
5. Создать ***Docker***-образ:
   - Содержащий ***Python 3***, а также библиотеки для него: ***Flask***, ***Psycopg2*** (для работы с СУБД ***Postgres***, хранящей данные) и ***ConfigParser***.
   - Содержащий код приложения на ***Python*** (копирующий его с локальной файловой системы). На хостовой ФС код будет лежать по пути: ***/srv/app/web.py***.
   - Содержащий конфигурационный файл приложения (копирующий его с локальной ФС). На хостовой ФС конфигурационный файл будет лежать по пути: ***/srv/app/conf/web.conf***.
   - При запуске контейнера, он должен запускать описанный выше код.
   - Образ должен быть оптимизирован с учетом лучших практик.
   - Пришлите ментору свой ***Dockerfile***, скриншот с работающим приложением и размер образа.

Для проверки работоспособности образа можно использовать приложение из репозитория (не забудьте поправить конфиг-файл).

Для этого потребуется склонировать репозиторий из ***GitHub*** (заодно вспомним, как работать с ***Git***), создать директорию ***/srv/app/conf*** и расположить файлы из склонированного репозитория так:

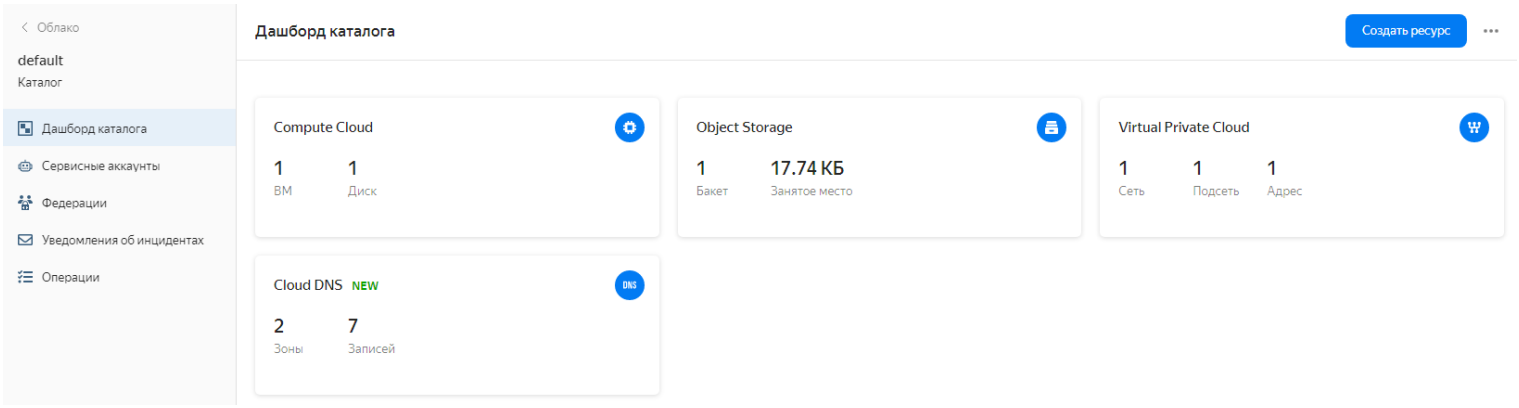
- ***web.py*** расположить в ***/srv/app/***;
- ***web.conf*** расположить в ***/srv/app/conf/***.

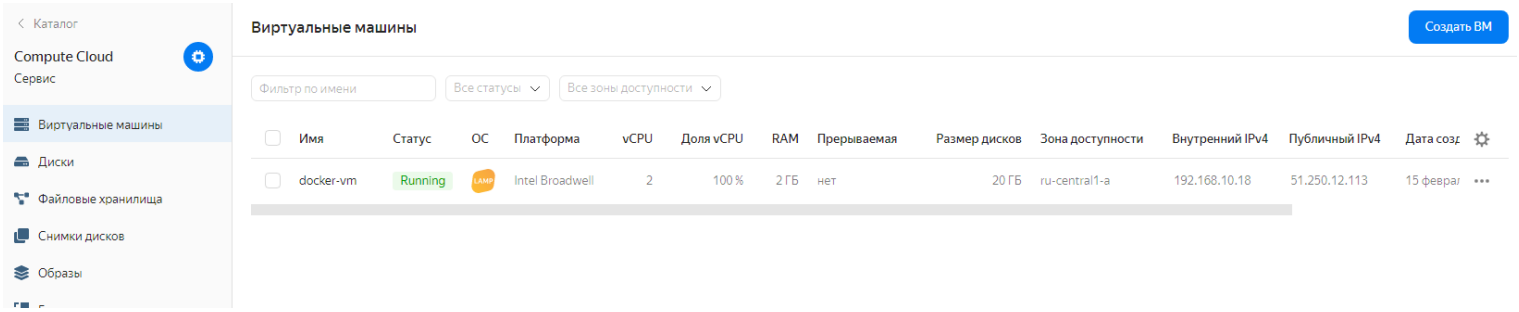
Затем запустить ***Docker***-контейнер, смонтировав ***/srv/app*** с хостовой ФС в контейнерную, а также пробросив порт 80 из контейнера в хостовую сеть.

# Terraform Machine creation

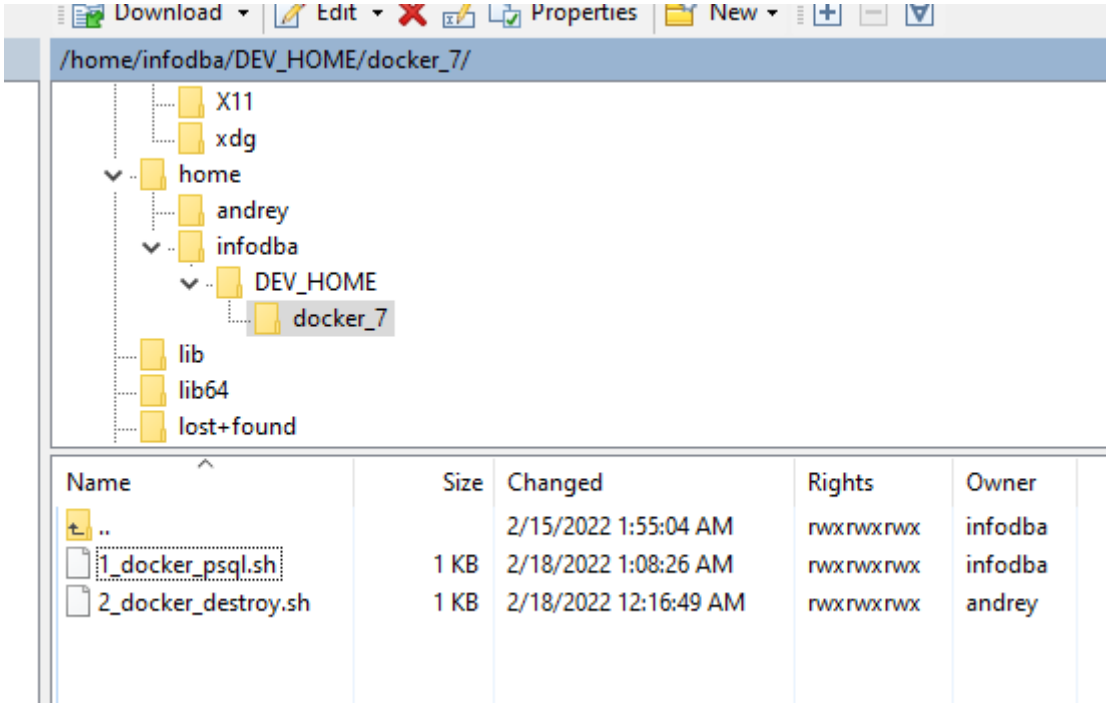
```
# ------------------------------------------------------------ VARIABLES
variable "zone" {                          # Используем переменную для передачи в конфиг инфраструктуры
  description = "Use specific availability zone" # Опционально описание переменной
  type        = string                     # Опционально тип переменной
  default     = "ru-central1-a"              # Опционально значение по умолчанию для переменной
}
variable "cloud_id" {
  type        = string                     # Опционально тип переменной
  default     = "b1gfdopk51c4d5reva85"         # Опционально значение по умолчанию для переменной
}
variable "folder_id" {
  type        = string                     # Опционально тип переменной
  default     = "b1gug0h1o834u3niipmr"         # Опционально значение по умолчанию для переменной
}
variable "cloud_key_file" {
  type        = string                     # Опционально тип переменной
  default     = "F:/DEV_HOME/Terraform_Projects/key_experiments/andrey_key.json"         # Опционально значение по умолчанию для переменной
}
variable "ssh_key_file" {
  type        = string                     # Опционально тип переменной
  default     = "F:/DEV_HOME/Terraform_Projects/key_experiments/andrey_key.pub"
}
variable "config_file" {
  type        = string                     # Опционально тип переменной
  default     = "F:/DEV_HOME/Terraform_Projects/key_experiments/andrey_config.yml"
}

# ------------------------------------------------------------ PROVIDER
terraform {
  required_providers {
    yandex = {
      source  = "yandex-cloud/yandex"
      version = "0.70.0" # Фиксируем версию провайдера
    }
  }
}

# Документация к провайдеру тут https://registry.terraform.io/providers/yandex-cloud/yandex/latest/docs#configuration-reference
# Настраиваем the Yandex.Cloud provider
provider "yandex" {
  service_account_key_file = var.cloud_key_file
  cloud_id  = var.cloud_id
  folder_id = var.folder_id
  zone      = var.zone # зона, в которая будет использована по умолчанию
}

# ------------------------------------------------------------ WORKING CODE
data "yandex_compute_image" "lamp" {
  family = "lamp"
}

resource "yandex_compute_instance" "docker_vm" {
  name              = "docker-vm"

  resources {
    cores  = 2
    memory = 2
  }

  boot_disk {
    initialize_params {
      image_id = data.yandex_compute_image.lamp.id
                 size = 20
                 type = "network-hdd"
    }
  }

  network_interface {
    subnet_id = yandex_vpc_subnet.subnet-1.id
    nat       = true
  }

  metadata = {
    ssh-keys = "${file(var.ssh_key_file)}"
               user-data = file(var.config_file)
  }
}

resource "yandex_vpc_network" "network-1" {
```

```
  name = "network1"
}

resource "yandex_vpc_subnet" "subnet-1" {
  name           = "subnet1"
  zone           = "ru-central1-a"
  network_id     = yandex_vpc_network.network-1.id
  v4_cidr_blocks = ["192.168.10.0/24"]
}


output "external_ip_address_docker_vm" {
  value = yandex_compute_instance.docker_vm.network_interface.0.nat_ip_address
}

output "internal_ip_address_docker_vm" {
  value = yandex_compute_instance.docker_vm.network_interface.0.ip_address
}
```



# Solution




**/etc/app/web.py**

```
from psycopg2 import connect
from flask import Flask
from configparser import ConfigParser


app = Flask(__name__)



def get_db_config(db_option):
        config = ConfigParser()
        config.read('/srv/app/conf/web.conf')
        try:
                result = config.get("database", db_option)
        except configparser.NoSectionError:
                print ('Cannot get {}. There is no such section or config file is unavailable/does not
exist').format(db_option)
                exit ()
        return result


def get_db_time():
        conn = connect("host= {0} dbname={1} user={2} password={3}".format(get_db_config('db_host'),
get_db_config('db_name'), get_db_config('db_user'), get_db_config('db_password')))
        conn.autocommit = True
        cur = conn.cursor()
        cur.execute('SELECT current_user;;')
        return cur.fetchone()


text = """<h1 style='color:blue'>Hello there!</h1>
Everything is OK! DB Query was completed by {} user""".format(get_db_time()[0])



@app.route("/")


def hello():
    return text

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

**/etc/app/conf/web.conf**

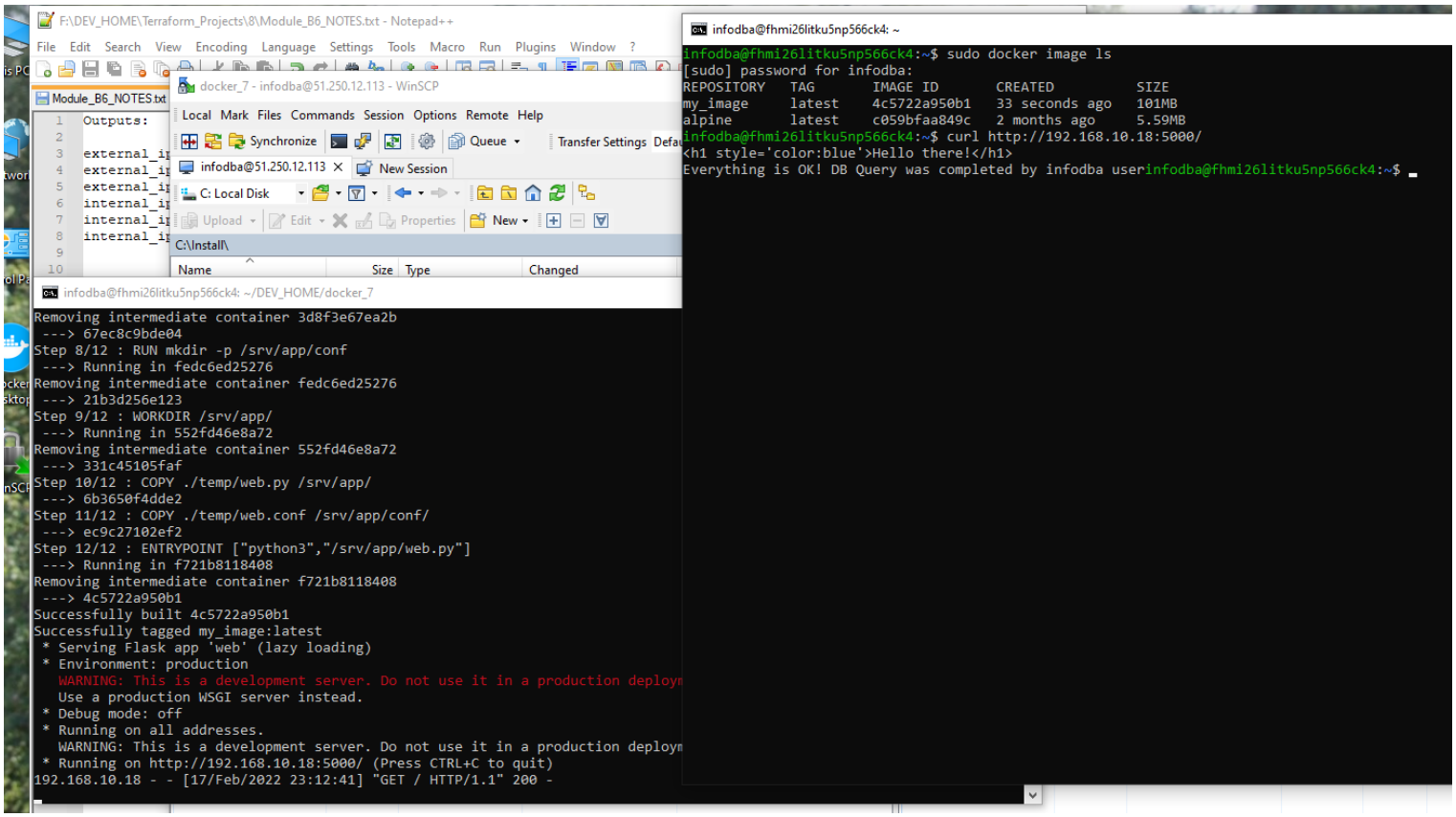
```
[database]
db_host = '192.168.10.18'
db_port = '5432'
db_user = 'infodba'
db_password = 'infodba'
db_name = 'my_psql_database'
```

**2_docker_destroy.sh**

```
#!/bin/sh
echo Destroy Andreys Docker Machine

rm -rf temp
rm -rf ./Dockerfile

docker rm $(docker ps -a -q)

sleep 1

docker image rm $(docker images -q)
```

**1_docker_psql.sh**

```
#!/bin/sh

mkdir ./temp
chmod 777 ./temp
cp /etc/app/web.py ./temp
cp /etc/app/conf/web.conf ./temp

cat << EOF > ./Dockerfile
# ----------------------------- Dockerfile START --------------------
FROM alpine:latest

RUN apk add --update \
py-pip \
linux-headers \
bash

RUN pip install --upgrade pip
RUN pip install Flask
RUN pip install psycopg2-binary
RUN pip install configparser==5.2.0

RUN mkdir -p /srv/app
RUN mkdir -p /srv/app/conf
WORKDIR /srv/app/

COPY ./temp/web.py /srv/app/
COPY ./temp/web.conf /srv/app/conf/

ENTRYPOINT ["python3","/srv/app/web.py"]
# ----------------------------- Dockerfile END  --------------------

EOF

cat ./Dockerfile

docker build . -t my_image --no-cache
docker run -it --name infodba_DEV -v vol:/infodba_VOL --network=host my_image -p 80:80
```


F:\DEV_HOME\Terraform_Projects\8\Module_B6_NOTES.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Module_B6_NOTES.txt
1 Outputs:
2
3 external_ip
4 external_ip
5 external_ip
6 internal_ip
7 internal_ip
8 internal_ip
9
10
docker_7 - infodba@51.250.12.113 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings
infodba@51.250.12.113
New Session
C: Local Disk
Upload
Edit
Properties
New
C:\Install\
Name
Size
Type
Changed
infodba@fhmi26litku5np566ck4: ~/DEV_HOME/docker_7
Removing intermediate container 3d8f3e67ea2b
---> 67ec8c9bde04
Step 8/12 : RUN mkdir -p /srv/app/conf
---> Running in fedc6ed25276
Removing intermediate container fedc6ed25276
---> 21b3d256e123
Step 9/12 : WORKDIR /srv/app/
---> Running in 552fd46e8a72
Removing intermediate container 552fd46e8a72
---> 331c45105faf
Step 10/12 : COPY ./temp/web.py /srv/app/
---> 6b3650f4dde2
Step 11/12 : COPY ./temp/web.conf /srv/app/conf/
---> ec9c27102ef2
Step 12/12 : ENTRYPOINT ["python3","/srv/app/web.py"]
---> Running in f721b8118408
Removing intermediate container f721b8118408
---> 4c5722a950b1
Successfully built 4c5722a950b1
Successfully tagged my_image:latest
* Serving Flask app 'web' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deploym
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deploym
* Running on http://192.168.10.18:5000/ (Press CTRL+C to quit)
192.168.10.18 - - [17/Feb/2022 23:12:41] "GET / HTTP/1.1" 200 -
infodba@fhmi26litku5np566ck4: ~
infodba@fhmi26litku5np566ck4:~$ sudo docker image ls
[sudo] password for infodba:
REPOSITORY TAG IMAGE ID CREATED SIZE
my_image latest 4c5722a950b1 33 seconds ago 101MB
alpine latest c059bfaa849c 2 months ago 5.59MB
infodba@fhmi26litku5np566ck4:~$ curl http://192.168.10.18:5000/
<h1 style='color:blue'>Hello there!</h1>
Everything is OK! DB Query was completed by infodba userinfodba@fhmi26litku5np566ck4:~$