ГУАП КАФЕДРА № 44

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший прподаватель |  |  |  | А.В. Аксенов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3 |
| КОНТЕЙНЕРЕЗАЦИЯ И DOCKET. АВТОМАТИЧЕСКОЕ МАСШТАБИРОВАНИЕ |
| по курсу: ОБЛАЧНЫЕ ТЕХНОЛОГИИ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |
| --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4941 |  | Н. С. Горбунов |
|  |  | подпись, дата | инициалы, фамилия |

Санкт-Петербург 2022

1. **Цель работы:** создать и настроить облачную инфраструктуру, запустить группу виртуальных машин, развернуть на них демонстрационное приложение, подключить группу к балансировщику сетевой нагрузки. Проверить работу балансировщика сетевой нагрузки.
2. **Задание**:

**Вариант 7.**

1. Создать Container Registry: Имя – Ваша фамилия транслитом Собрать докер образ с тегом: cr.yandex/<registry-id>/lab3-app:latest
2. Опубликовать образ в Yandex Container Registry

# Создание облачной инфраструктуры:

1. Зайти в консоль управления.
2. Создать VPC со следующими параметрами:
   1. Имя – Ваша фамилия транслитом
   2. Описание – lab3
3. Создать две публичные подсети в новой VPC со следующими параметрами:
   1. Имя – public-subnet-1, public-subnet-2
   2. AZ – ru-central1-a, ru-central1-b
   3. 3.3. IPv4 CIDR block – 10.n.1.0/24, 10.n.2.0/24, где n – Ваш порядковый номер по списку
4. Настроить таблицу маршрутизации: 4.1.Создать шлюз с именем igw
   1. Создать таблицу маршрутизации с именем internet и провести маршрут к новому шлюзу
   2. Привязать таблицу маршрутизации к публичным подсетям
5. Создать группу виртуальных машин: Параметры запуска:
   1. Имя – Ваше имя транслитом
   2. Сервисный аккаунт – создать новый с именем lab3 и ролью editor
   3. Распределение – зоны доступности, в которых были созданы публичные подсети
   4. Шаблон ВМ:
      1. Container Solution – Docker-контейнер на основе
         1. Docker-образ – ранее опубликованный Вами в реестр Docker-образ
         2. Переменные окружения – APP\_PORT = 8000 + n, где n – Ваш порядковый номер по списку
      2. Resources:
         1. Платформа – Inter Cascade Lake
         2. vCPU – 2
         3. Доля vCPU – 5%
         4. RAM – 1GB
         5. Прерываемая – Yes
      3. Network:
         1. Сеть – созданная в пункте 2
         2. Подсети – созданные в пункте 3
         3. Публичный адрес – автоматически
      4. Access
         1. Логин – student
         2. SSH-ключ – сгенерировать SSH-ключ типа ed25519
   5. Масштабирование:
      1. Тип – Фиксированный
      2. Размер – 2
   6. Интеграция с Network Load Balancer:
      1. Целевая группа:
         1. Имя – tg-lab3
6. Балансировщик сетевой нагрузки:
   1. Имя – nlb-lab3
   2. Тип – внешний
   3. Публичный адрес – автоматически
   4. Обработчик:
      1. Имя – listener-lab3
      2. Внешний порт – 80
      3. Внутренний порт – 8000 + n, где n – Ваш порядковый номер по списку
   5. Целевая группа – созданная в пункте 5.6.1
   6. Проверка состояния – HTTP, должна выполняться на порт приложения
7. **Ход выполнения работы**

Аналогично лабораторной работе создается VPC, с таблицей маршрутизации и шлюзом. (Рис. 1-2)

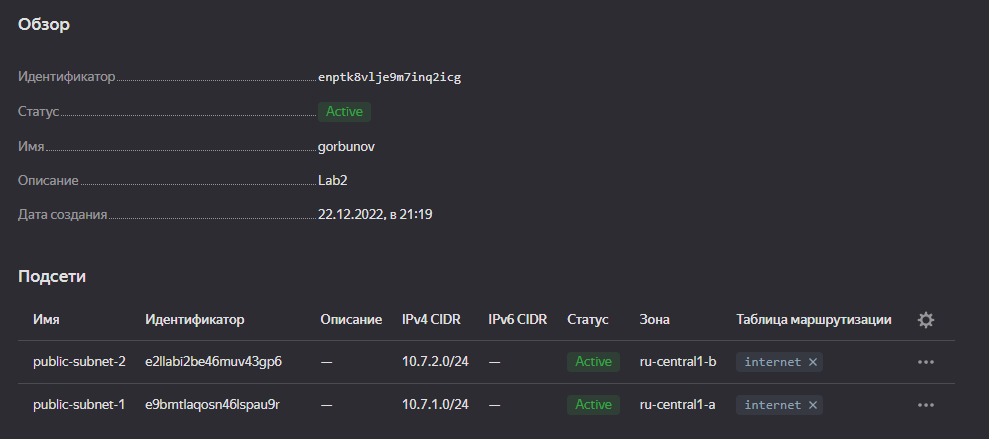
****

Рисунок 1 –Параметры VPC.

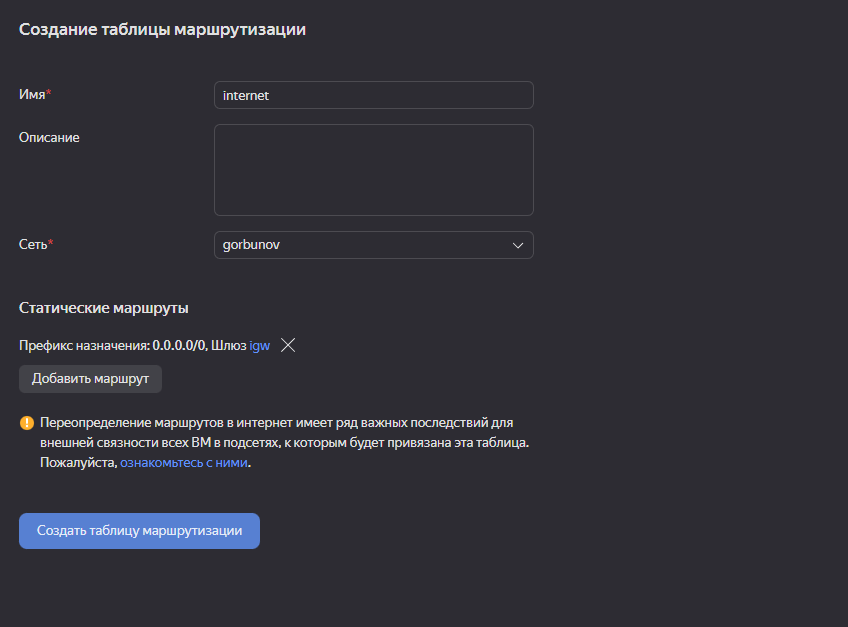
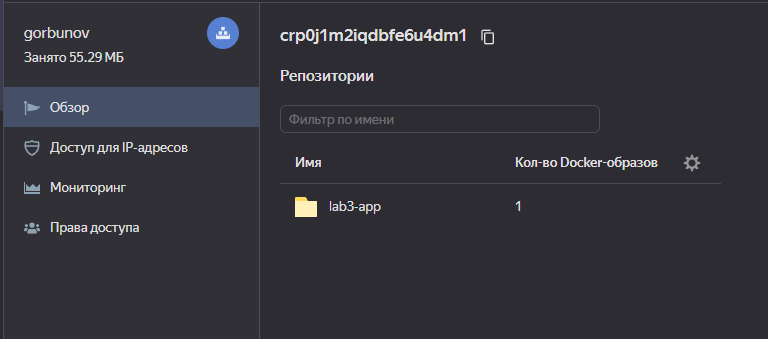


Рисунок 2 – Создание таблицы маршрутизации

Создается Container Registry, в который загружается docker образ. (Рис. 3)

  
Рисунок 3 – Container Registry

Создается кластер виртуальных машин с шаблоном Container Solution (Рис. 4-7)

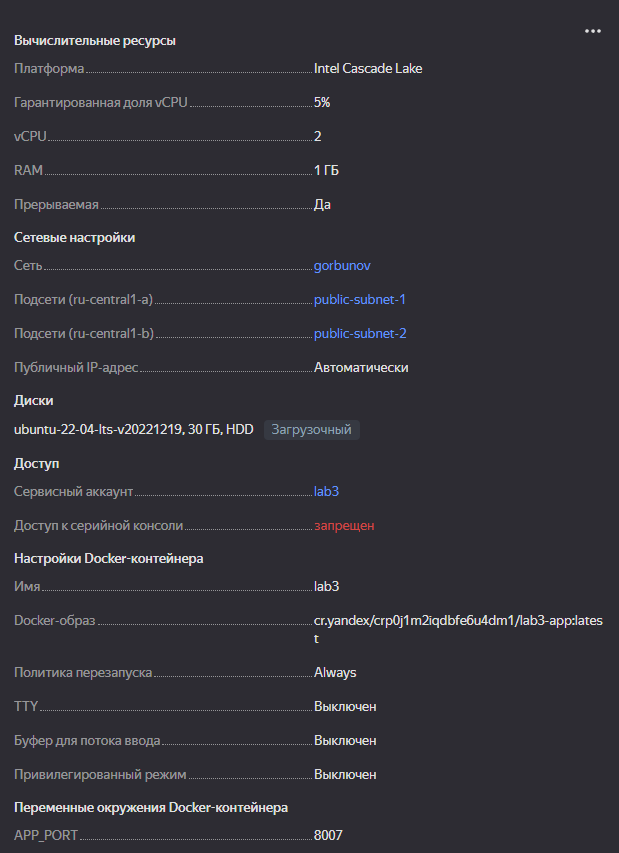


Рисунок 4 – Шаблон VM

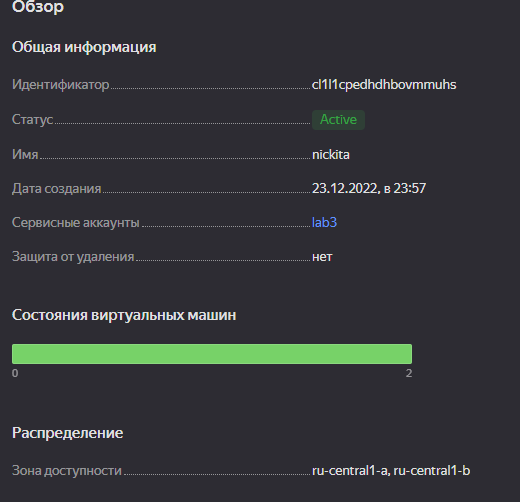


Рисунок 5 – Параметры кластера (1)

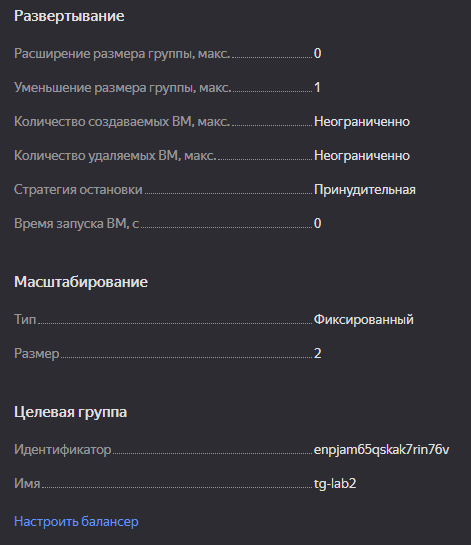


Рисунок 6 – Параметры кластера (2)

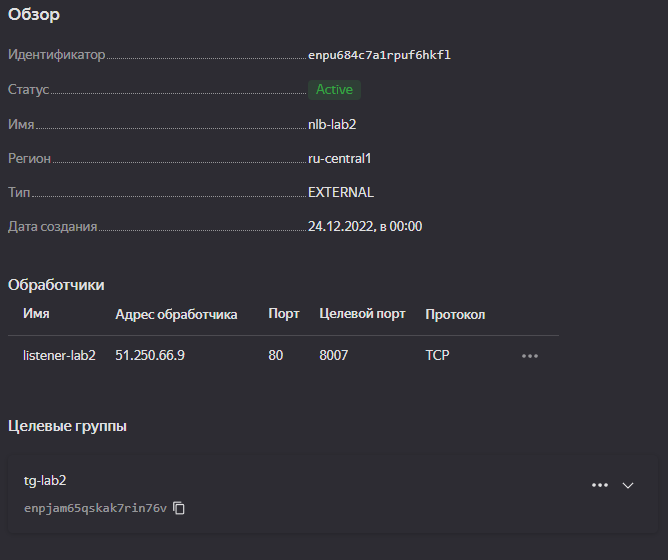


Рисунок 7 – Параметры балансера

Приложение развертывается автоматически:

Dockerfile:

FROM alpine:latest  
  
RUN apk update  
  
RUN apk add --no-cache python3 py3-pip  
  
RUN pip3 install fastapi uvicorn  
COPY app/main.py ./app/main.py  
WORKDIR /app  
  
CMD ["sh", "-c", "uvicorn main:app --host 0.0.0.0 --port $APP\_PORT"]

Где main.py – программа приложения:

from fastapi import FastAPI  
  
app = FastAPI()  
  
  
@app.get("/")  
def read\_root():  
 return {"lastname":"Gorbunov"}

Правильность проделанной работы проверяется с помощью JSON запроса. (Рис. 8)

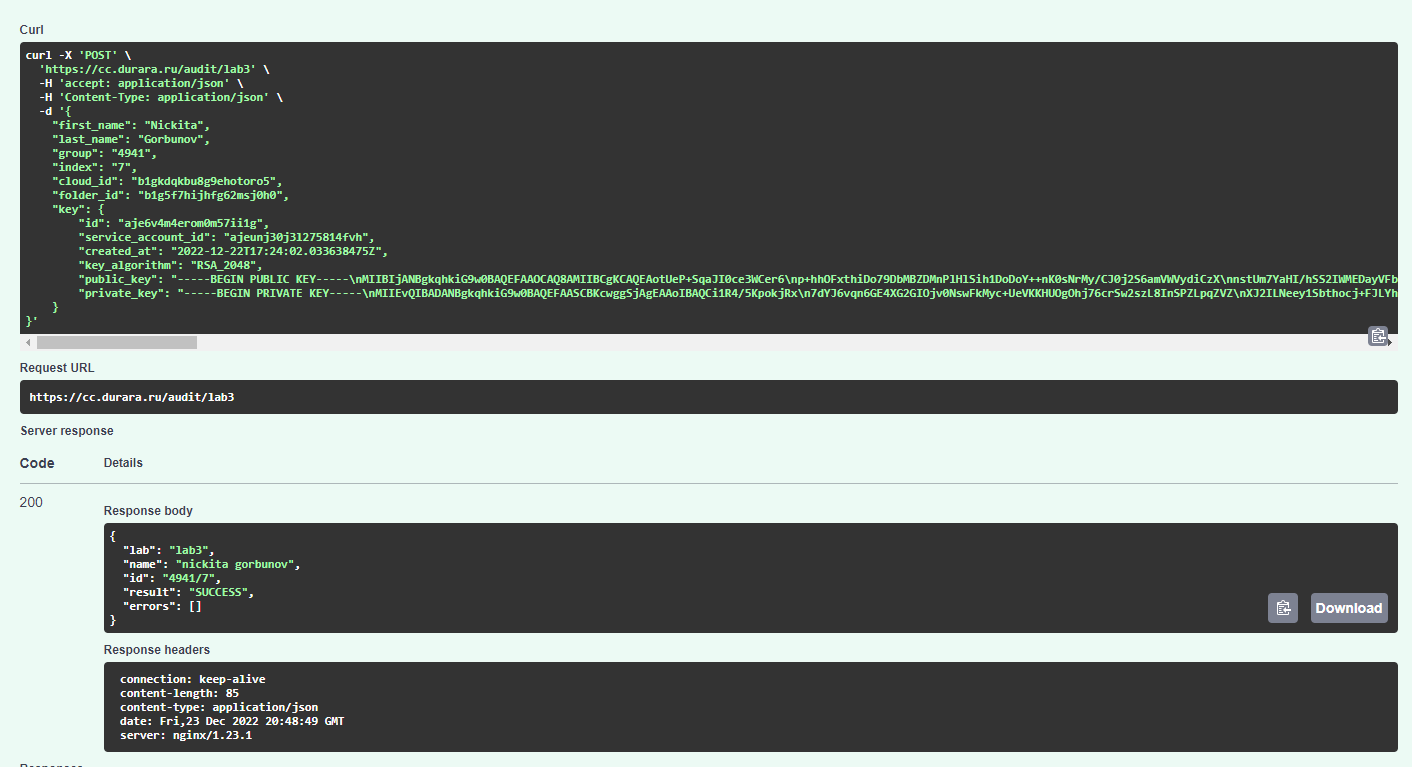


Рисунок 8 – Проверка выполнения работы

**Вывод:** в ходе выполнения работы создал и настроил облачную инфраструктуру, запустил группу виртуальных машин, развернул на них демонстрационное приложение, подключил группу к балансировщику сетевой нагрузки. Проверил работу балансировщика сетевой нагрузки.