**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

**Інститут комп’ютерних технологій, автоматики та метрології**

**Кафедра "Комп'ютеризовані системи автоматики"**



**ЗВІТ**

про виконання **лабораторної роботи №2**

з навчальної дисципліни**: «Хмарні технології»**

**Виконав:**

студент групи ІР-31

Борецький Б. І.

**Прийняв:**

к.т.н., асистент

Рибак В. І.

Львів — 2023

**Завдання**

Обовязкові

* (2б) Створити докер образи для розгортання REST сервісу з лабораторної роботи 1 (якщо це ще не зроблено)
* (2б) Завантажити контейнери на хмару відповідно до контейнерних сервісів хмарного провайдера.
* (2б) Налаштувати контейнери на автоматичне масштабування (збільшення і зменшення) в залежності від навантаження на CPU чи Memory.
* (2б) Написати сценарій для автоматичного генерування навантаження на REST сервіс та емулятор генерації запитів до веб сайту.
* (2б) Запустити сценарій і прослідкувати масштабування контейнерів. В журналі подій відслідкувати масштабування і навантаження на сервіси.

Бонусні

* (2б) Запустити виконання сценарію автоматичного генерування з окремого контейнера
* (2б) Завантажити контейнер з сценарієм автоматичного генерування в окремий сервіс на хмарі

**Виконання**

Написав dockerfile для свого проекту

FROM amazoncorretto:17

COPY target/\*.jar /app.jar

EXPOSE 80

ENTRYPOINT ["java", "-jar", "/app.jar"]

Та створеного стрес-тест сервісу

FROM locustio/locust

COPY locustfile.py /home/locust

EXPOSE 80

from locust import HttpUser, TaskSet, task

import logging

class UserBehavior(TaskSet):

    def \_\_init\_\_(self, parent):

        super(UserBehavior, self).\_\_init\_\_(parent)

        self.token = self.login()

        self.headers = {}

        print(self.token)

    def on\_start(self):

        self.token = self.login()

        self.headers = {'Authorization': 'Bearer {}'.format(self.token)}

        self.login()

    def login(self):

        response = self.client.post("auth/login",

                                    json={"email": "bogdanbor.etc@gmail.com",

                                          "password": "password"})

        logging.info(f"Status code: {response.status\_code}")

        logging.info(f"Json response: {response.json()}")

        return response.json().get("token")

    @task

    def get\_medical\_infos(self):

        self.client.get("api/medicalInfos", headers=self.headers)

    @task

    def get\_fatigue\_monitorings(self):

        self.client.get("api/fatigueMonitorings", headers=self.headers)

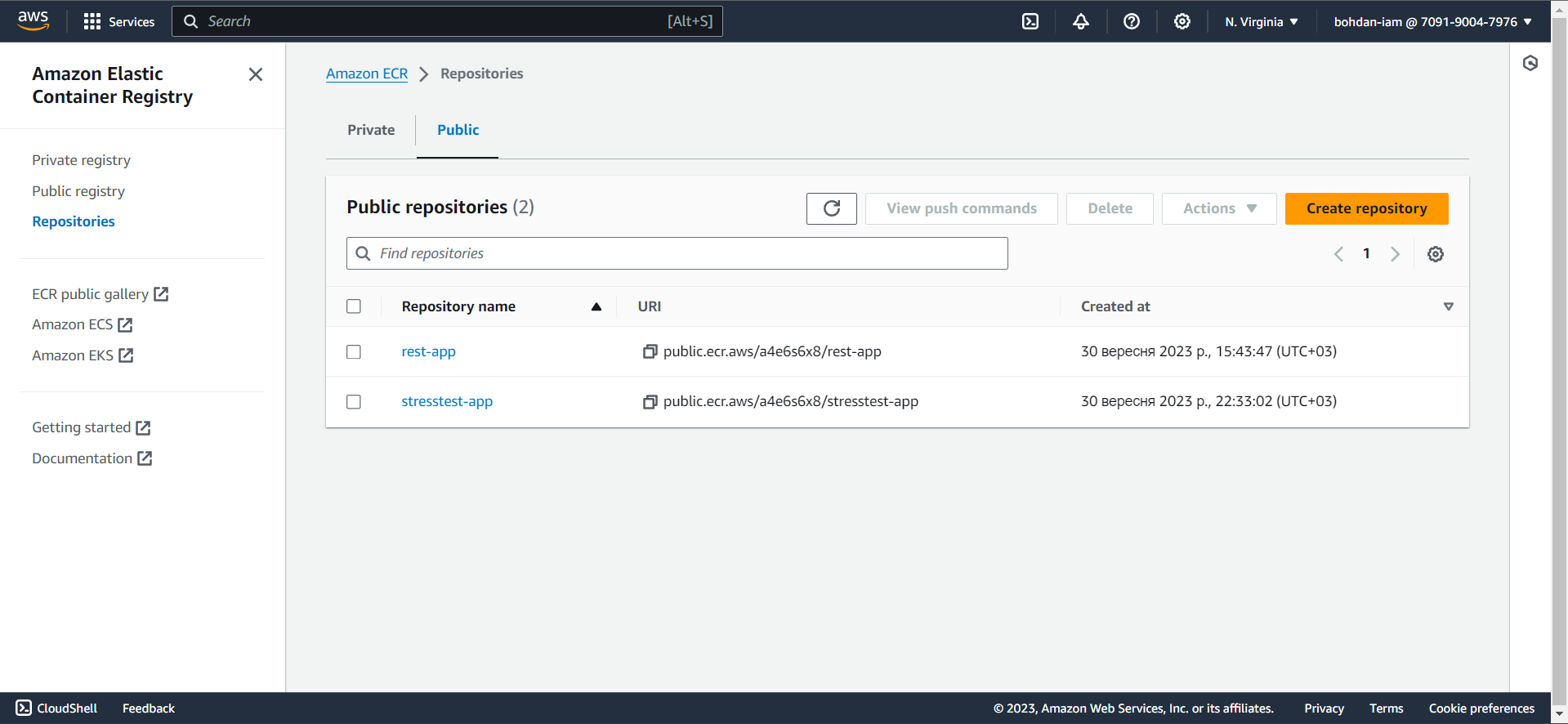
class WebsiteUser(HttpUser):

    tasks = [UserBehavior]

    min\_wait = 5000

    max\_wait = 9000

Завантажив контейнери у ECR.



Запустив сценарій і прослідкував масштабування контейнерів. При запуску навантаження на сервіс, підвищелось навантаження на CPU, спрацював CloudWatch Alarm що збільшив кількість Tasks у моєму ECS service. Після цього ASG ініціалізувала створення ще однієї віртуальної машини, для виконання ініціалізованого task-у.

**Висновок**

Виконавши цю лабораторну роботу, я навчився працювати з сервісами ECS, ECR, ASG, LB.