



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №1
Тенденції розвитку інформаційних систем та технологій
CI/CD. Хмарні обчислення.

Виконав
студент групи ІТ-41ф

Новиков Д. М.

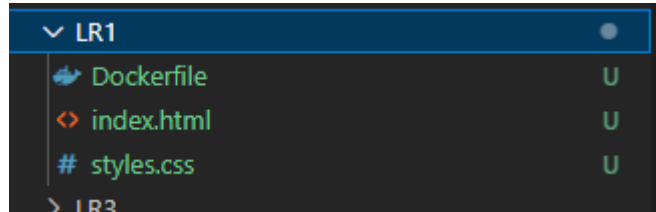
Перевірив:

ас. Цимбал С. І.

Мета роботи: створити CI/CD, який буде доставляти в хмару оновлені зміни до застосунку при коміті коду в GitHub репозиторій.

Хід роботи:

1. Обрати хмарну платформу і створити аккаунт на ній (наприклад, AWS, GCP). Обрано AWS як хмарну платформу, і аккаунт вже існує.
2. Створити простий веб-застосунок html + css, створити для нього Dockerfile на базі nginx (можна використати матеріали з минулої лабораторної роботи):



index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Web App</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>I'm not a fan of front-end development, so I apologize for the simplicity of the app :(</h1>
  <p>This is a simple HTML and CSS application deployed using Docker on AWS.</p>
</body>
</html>
```

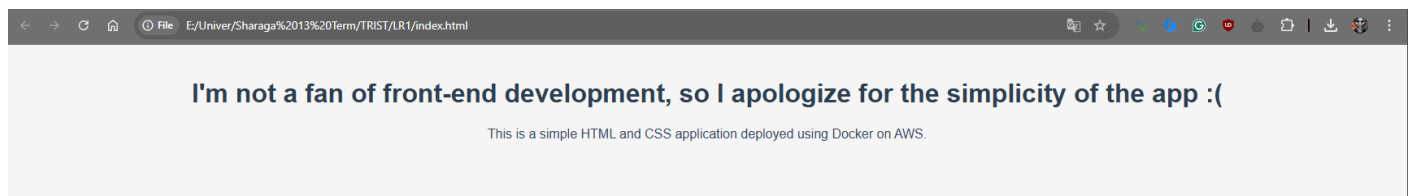
styles.css:

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background-color: #f5f5f5;
  margin: 0;
  padding: 20px;
}
h1 {
  color: #2c3e50;
}
p {
  color: #34495e;
}
```

Dockerfile:

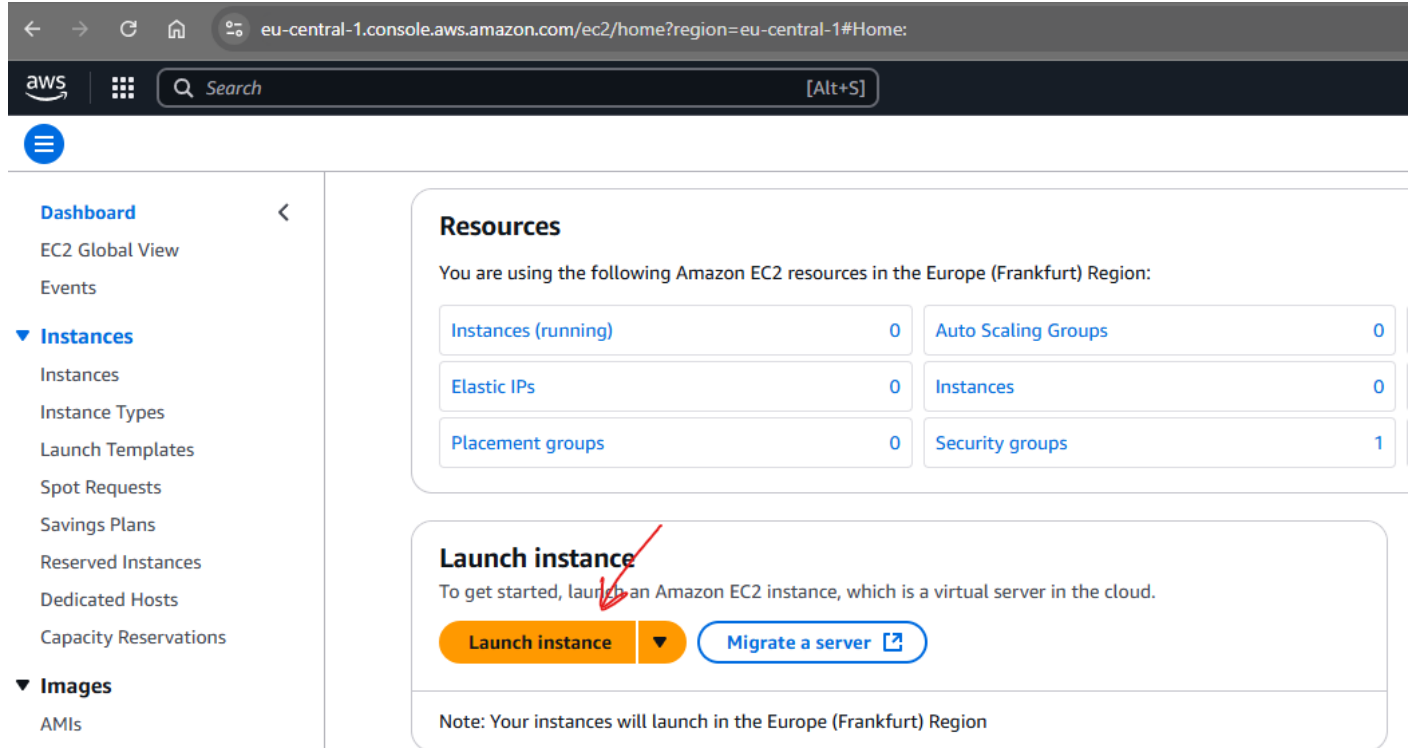
```
FROM nginx:alpine
COPY . /usr/share/nginx/html
EXPOSE 80
```

Застосунок має наступний вигляд:



3. Створити VM instance та security group (застосунок має бути доступний на 80 порті, HTTP), встановити на нього docker.

а) Створимо інстанс EC2:



aws

Search

[Alt+S]

EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

test-lr1

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Debian

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-0084a47cc718c111a (64-bit (x86)) / ami-099a546c02844706e (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture

AMI ID

Username

64-bit (x86)

ami-0084a47cc718c111a

ubuntu

Verified provider

▼ Instance type Info | [Get advice](#)

Instance type

t2.micro
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.018 USD per Hour On-Demand SUSE base pricing: 0.0134 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0152 USD per Hour On-Demand Linux base pricing: 0.0134 USD per Hour
On-Demand RHEL base pricing: 0.0278 USD per Hour

Free tier eligible

All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

б) Згенеруємо пару ключів для підключення, завантажимо приватний ключ на локальний пристрій:

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Select

Create new key pair

Create key pair ✕

Key pair name
Key pairs allow you to connect to your instance securely.

keys-lr1

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

☒ **RSA**
RSA encrypted private and public key pair

☐ **ED25519**
ED25519 encrypted private and public key pair

Private key file format

☒ **.pem**
For use with OpenSSH

☐ **.ppk**
For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

[Cancel](#) [Create key pair](#)

- в) Створимо групу безпеки з наступними налаштуваннями:
- а. Дозволимо HTTP-трафік, щоб забезпечити доступ до веб-застосунку;
 - б. Дозволимо SSH-підключення для управління інстансом.

▼ Network settings Info

Network | Info
vpc-05b5b4786229a5f4e

Subnet | Info
No preference (Default subnet in any availability zone)

Auto-assign public IP | Info
Enable
Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

☒ Allow SSH traffic from
Helps you connect to your instance
My IP
212.110.138.79/32
☒ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. ✕

г) Інші налаштування залишаємо за замовчуванням і натискаємо Launch Instance:



✔ Success
Successfully initiated launch of instance (i-01bc42345113f8120)

► Launch log

д) Після створення перевіримо інформацію для підключення:

Connect to your instance

Once your instance is running, log into it from your local computer.

Connect to instance ↗

[Learn more ↗](#)

Connect to instance [Info](#)

Connect to your instance i-01bc42345113f8120 (test-lr1) using any of these options

EC2 Instance Connect Session Manager **SSH client** EC2 serial console

Instance ID

i-01bc42345113f8120 (test-lr1)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is keys-lr1.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
☐ `chmod 400 "keys-lr1.pem"`
4. Connect to your instance using its Public DNS:
☐ `ec2-3-121-227-25.eu-central-1.compute.amazonaws.com`

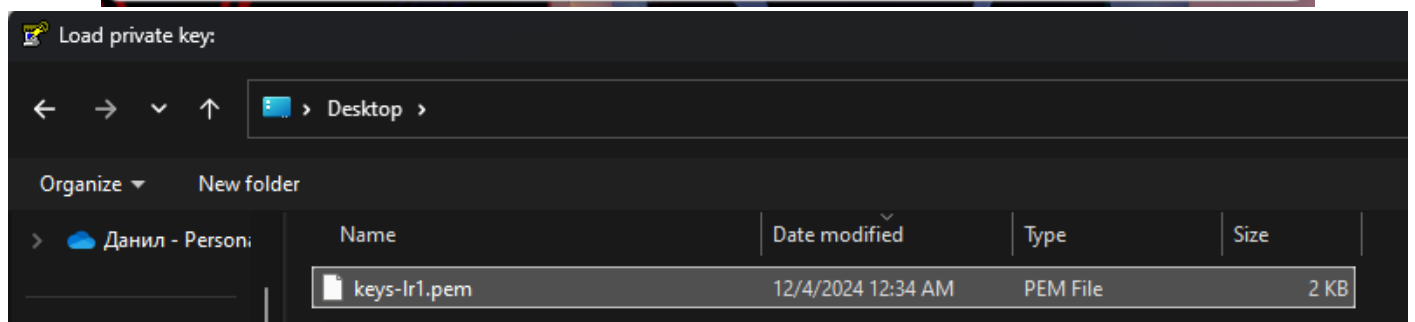
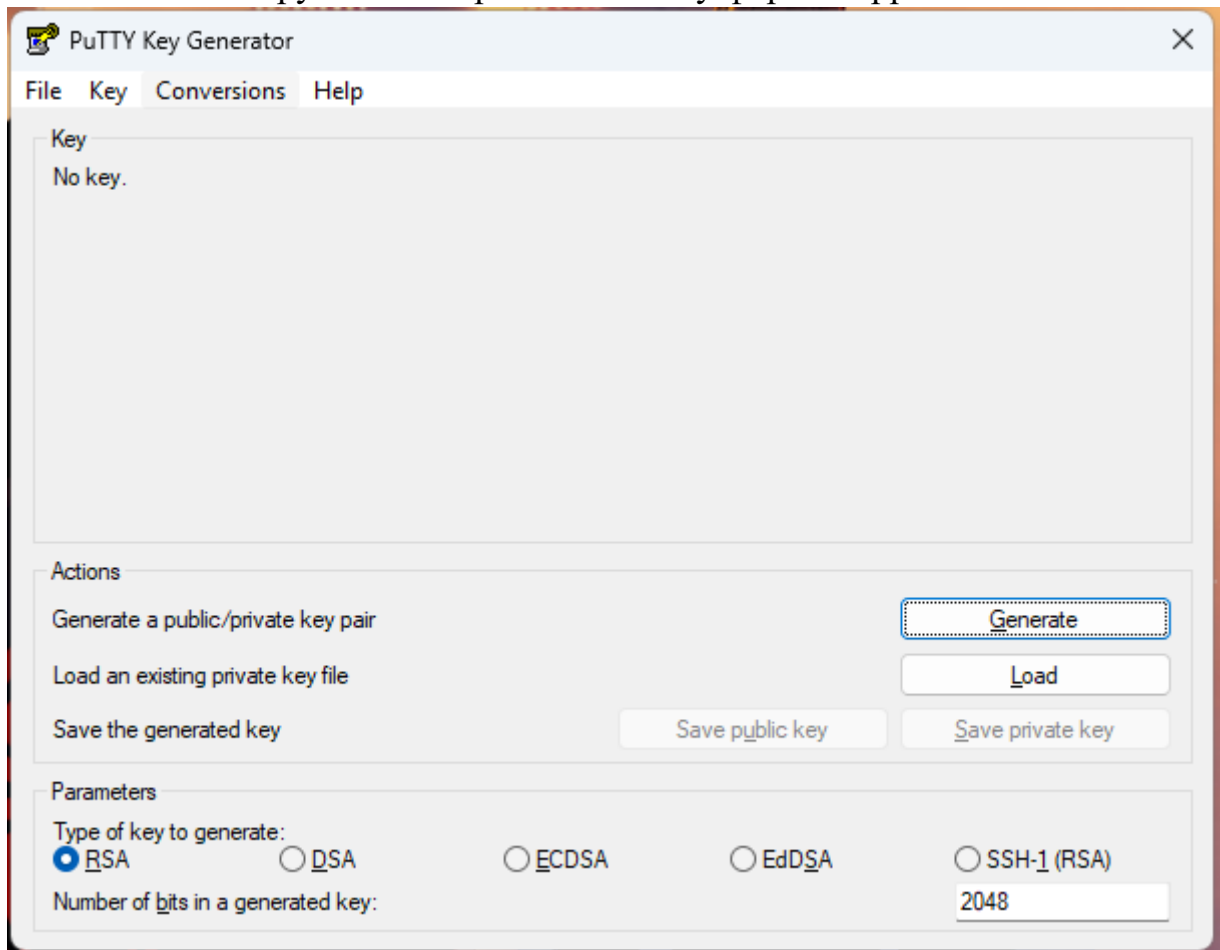
Example:

`ssh -i "keys-lr1.pem" ubuntu@ec2-3-121-227-25.eu-central-1.compute.amazonaws.com`

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#)

- е) Підключимося через SSH до створеного EC2-інстансу, використовуючи PuTTY. Оскільки PuTTY вимагає ключ у форматі .ppk, для його генерації скористаємося утилітою PuTTY Key Generator:
- Відкриваємо PuTTY Key Generator;
 - Завантажуємо приватний ключ у форматі .pem;
 - Генеруємо та зберігаємо ключ у форматі .ppk.



PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDQ0fMZqKyKZ5SOBRBJHOlhxQN0AFXCpCesTz0wLx1M0pPsMBN9
LSXGUlb4IG4rs7RPZltVYJzRmwRR7V1QSAyloWslYD35fYWxwT5nOymr9LYIGRXEaExH7U4ektulEQP8SsqcJK
2KEDg3bvvyAbOVpbDjzFxDKsDxaD1k3anpz1X16CIQpL10XpqNKDCfkXv/0ZXnC
+5ZjmrXDYJF16AAsnBIKf
```

Key fingerprint: ssh-rsa 2048 SHA256:H1y5Fs8ISLsa4Pj7OyCv5uZdh8UUwBythScuhRwQZS0

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

Save the generated key Save public key Save private key

Parameters

Type of key to generate:

☒ RSA ☐ DSA ☐ ECDSA ☐ EdDSA ☐ SSH-1 (RSA)

Number of bits in a generated key: 2048

PuTTYgen Notice

Successfully imported foreign key (OpenSSH SSH-2 private key (old PEM format)). To use this key with PuTTY, you need to use the "Save private key" command to save it in PuTTY's own format.

OK

PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDQ0fMZqKyKZ5SOBRBJHOlhxQN0AFXCpCesTz0wLx1M0pPsMBN9
LSXGUlb4IG4rs7RPZltVYJzRmwRR7V1QSAyloWslYD35fYWxwT5nOymr9LYIGRXEaExH7U4ektulEQP8SsqcJK
2KEDg3bvvyAbOVpbDjzFxDKsDxaD1k3anpz1X16CIQpL10XpqNKDCfkXv/0ZXnC
+5ZjmrXDYJF16AAsnBIKEHsl/IA7eeXbGAuer/aEETSTZQMCYx
```

Key fingerprint: ssh-rsa 2048 SHA256:H1y5Fs8ISLsa4Pj7OyCv5uZdh8UUwBythScuhRwQZS0

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

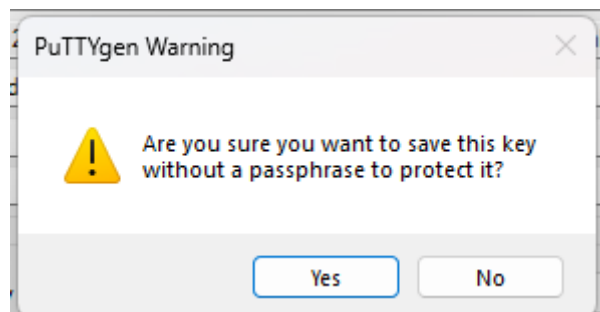
Save the generated key Save public key Save private key

Parameters

Type of key to generate:

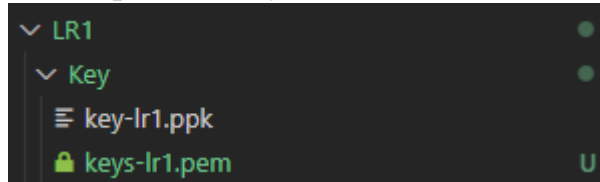
☒ RSA ☐ DSA ☐ ECDSA ☐ EdDSA ☐ SSH-1 (RSA)

Number of bits in a generated key: 2048



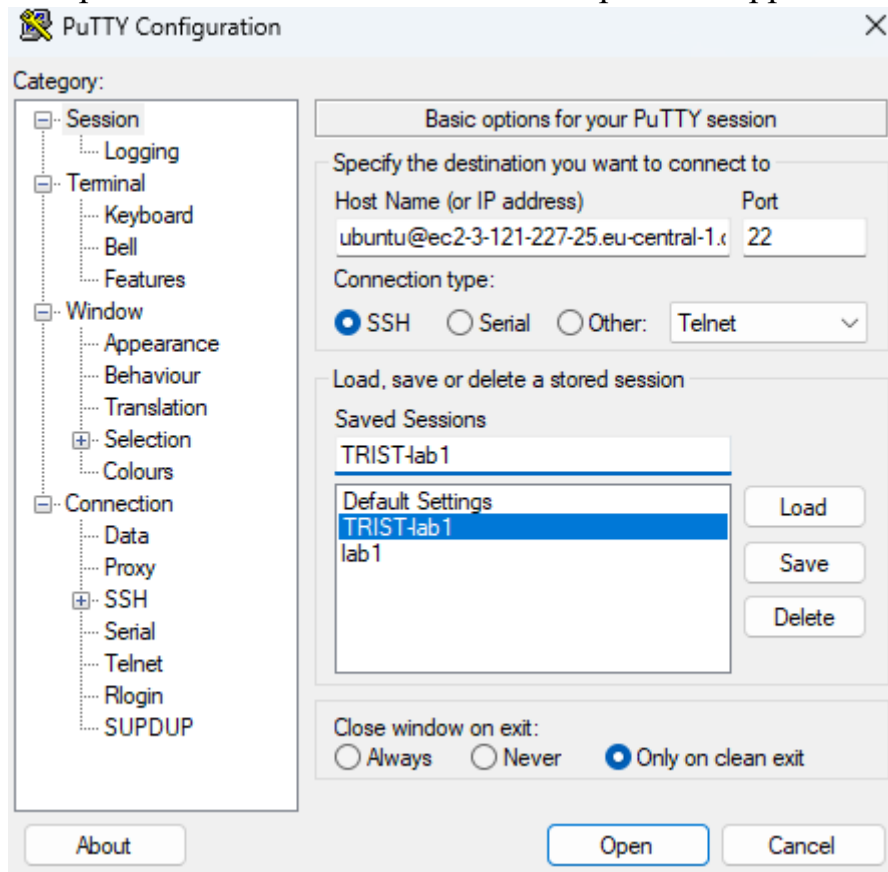
ж) У результаті маємо пару ключів:

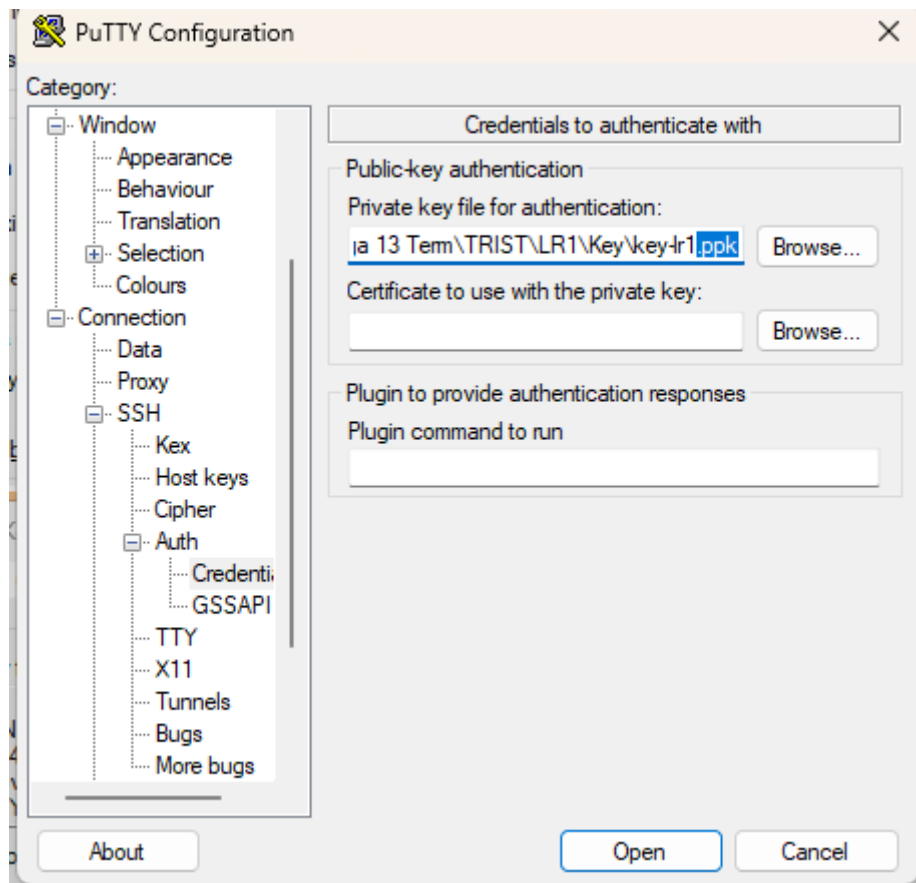
- a. .pem для використання у CLI або інших інструментах;
- b. .ppk для використання у PuTTY.



з) Налаштуємо підключення в PuTTY:

- a. У полі Host Name (or IP address) вкажемо публічну IP-адресу інстансу, отриману з розділу "Інформація про підключення";
- b. У розділі SSH > Auth додамо згенерований .ppk ключ:





и) Виконаємо підключення до EC2-інстансу через SSH:

```
ubuntu@ip-172-31-34-233: ~  
Unable to use certificate file "E:\Univer\Sharaga 13 Term\TRIST\LR1\Key\keys-  
lr1.pem" (OpenSSH SSH-2 private key (old PEM format))  
Using username "ubuntu".  
Authenticating with public key "imported-openssh-key"  
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/pro  
  
System information as of Tue Dec  3 22:53:25 UTC 2024  
  
System load:  0.0          Processes:      106  
Usage of /:   22.8% of 6.71GB  Users logged in:  0  
Memory usage: 20%          IPv4 address for enX0: 172.31.34.233  
Swap usage:   0%  
  
Expanded Security Maintenance for Applications is not enabled.  
  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-34-233:~$
```

к) Встановимо Docker на EC2-інстанс через SSH для запуску контейнерів із додатком і Watchtower:

```
sudo apt update  
sudo apt install -y docker.io  
sudo usermod -aG docker $USER  
newgrp docker
```

```
ubuntu@ip-172-31-34-233:~$ sudo apt update
Hit:1 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [498 kB]
Get:8 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://eu-central-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 C-n-f Metadata [301 kB]
```

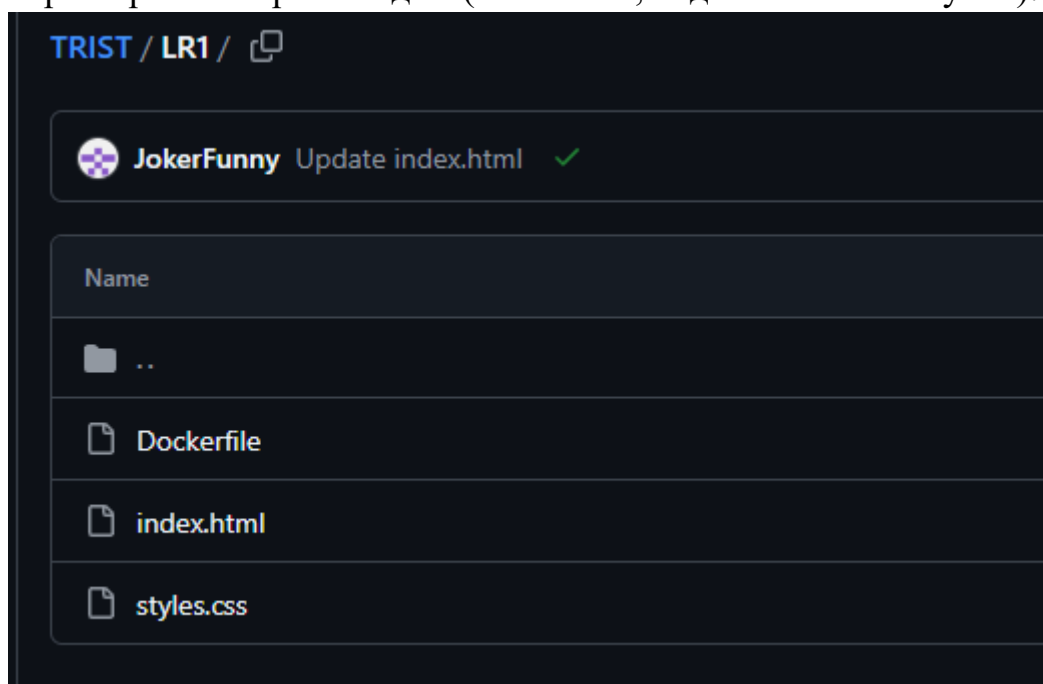
```
...
Building dependency tree... Done
Reading state information... Done
58 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-34-233:~$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
...
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-34-233:~$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-34-233:~$ newgrp docker
ubuntu@ip-172-31-34-233:~$
```

4. Створити репозиторій з кодом (Dockerfile, код вашого застосунка):



5. Запустити два контейнери – створений образ та watchtower:

а) Для запуску контейнерів спершу необхідно створити образ у публічному Docker-реєстрі (оскільки використання приватного не вимагається). Використаємо DockerHub. Увійдемо в обліковий запис за допомогою команди «docker login»:

```
PS E:\Univer\Sharaga 13 Term\TRIST\LR1> docker login
Authenticating with existing credentials...
Login did not succeed, error: error during connect: Post "http://%2F%2Fpipe%2FdockerDesktopLinuxEngine/v1.47/auth": open //./pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.

USING WEB-BASED LOGIN
To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: NQmZ-XJHF
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser...
Login Succeeded
```



Sign in

Using Docker for work? We recommend signing in with your work email address.

Username or email address*

Joker759

Continue

OR



Continue with Google



Continue with GitHub

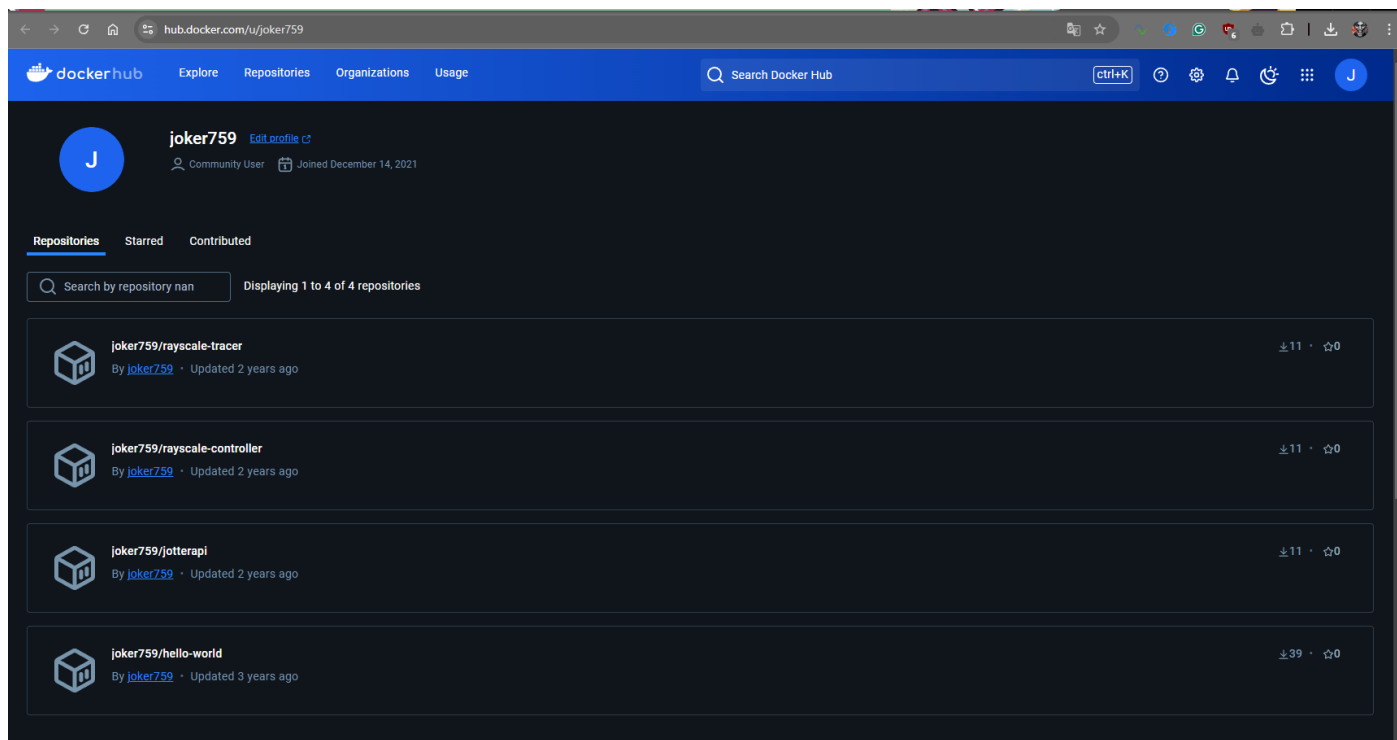
Don't have an account? [Sign Up](#)



Congratulations, you're all set!

Your device is now connected.

6) Перейдемо на DockerHub (<https://hub.docker.com/u/joker759>) та перевіримо існуючі репозиторії:



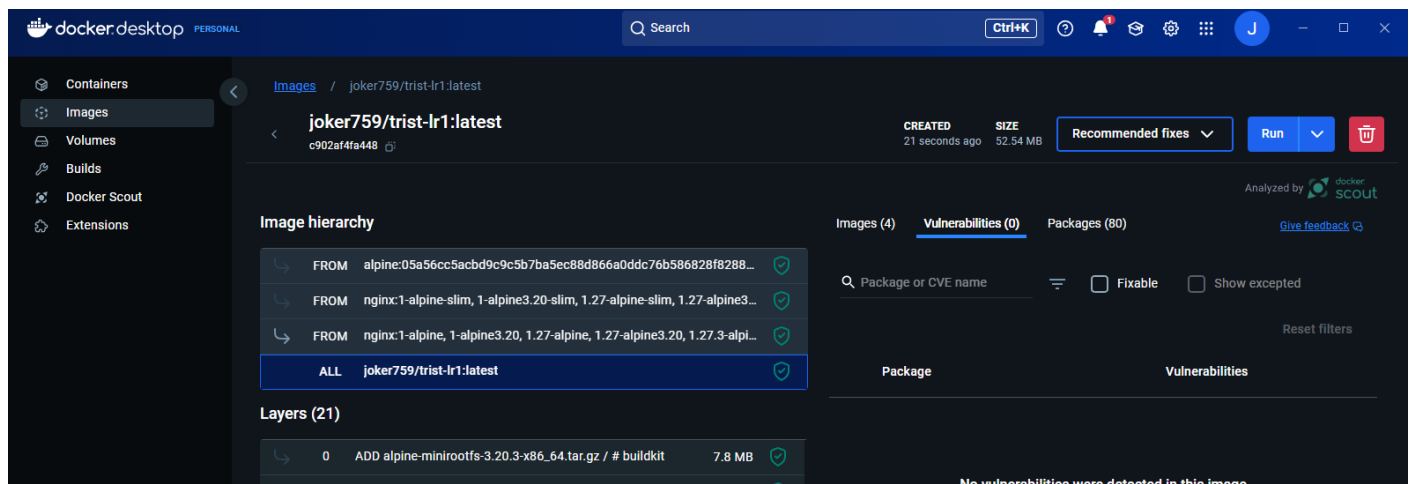
в) Зблдімо Dockerfile для створення Docker-образу, використовуючи команду «docker build -t joker759/trist-lr1:latest .»:

```
PS E:\Univer\Sharaga 13 Term\TRIST\LR1> docker build -t joker759/trist-lr1:latest .
[+] Building 5.0s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 95B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 4.08kB
=> [1/2] FROM docker.io/library/nginx:alpine@sha256:41523187cf7d7a2f2677a80609d9caa14388bf5c1fbca9c410ba3de602aaaab4
=> => resolve docker.io/library/nginx:alpine@sha256:41523187cf7d7a2f2677a80609d9caa14388bf5c1fbca9c410ba3de602aaaab4
=> => sha256:91ca84b4f57794f97f70443afccff26aed771e36bc48bad1e26c2ce66124ea66 11.25kB / 11.25kB
=> => sha256:41523187cf7d7a2f2677a80609d9caa14388bf5c1fbca9c410ba3de602aaaab4 10.36kB / 10.36kB
=> => sha256:b1f7437a6d0398a47a5d74a1e178ea6fff3ea692c9e41d19c2b3f7ce52cdb371 2.50kB / 2.50kB
=> => sha256:af9c0e53c5a430c700d068066f35cb313945c9917bee94108bae13a933f6b6b4 628B / 628B
=> => sha256:e10e486de1ab216956a771c782ef1adabef10b1bfd9a3765e14f79484784e9cd 4.06MB / 4.06MB
=> => sha256:b2eb2b8af93a0c4d2b5f5a70ed620869b406658462aba70b03f12f442aa40cc1 956B / 956B
=> => sha256:e351ee5ec3d4f55b4e3fce972c2a34a5632ede02602dfbcad85afc539b486131 405B / 405B
=> => extracting sha256:e10e486de1ab216956a771c782ef1adabef10b1bfd9a3765e14f79484784e9cd
=> => sha256:fbfb7d28be71101773e4440c75dbbe7ed12767763fbb2e9c85a32a31f611169a 1.21kB / 1.21kB
=> => sha256:471412c08d15ee3b0c86b86fe91a6dd0e17d1f4d1b6d83a7f68e9b709328bf3d 1.40kB / 1.40kB
=> => sha256:a2eb5282fbec00fa3d13849dafbfd7f416b69059e527e5653b84f1d9245b8eb0 15.10MB / 15.10MB
=> => extracting sha256:af9c0e53c5a430c700d068066f35cb313945c9917bee94108bae13a933f6b6b4
=> => extracting sha256:b2eb2b8af93a0c4d2b5f5a70ed620869b406658462aba70b03f12f442aa40cc1
=> => extracting sha256:e351ee5ec3d4f55b4e3fce972c2a34a5632ede02602dfbcad85afc539b486131
=> => extracting sha256:fbfb7d28be71101773e4440c75dbbe7ed12767763fbb2e9c85a32a31f611169a
=> => extracting sha256:471412c08d15ee3b0c86b86fe91a6dd0e17d1f4d1b6d83a7f68e9b709328bf3d
=> => extracting sha256:a2eb5282fbec00fa3d13849dafbfd7f416b69059e527e5653b84f1d9245b8eb0
=> [2/2] COPY . /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:c902af4fa448ae6d56c852e9ce82e1f0ddf08a87d3d51a32e2921890a641ea21
=> => naming to docker.io/joker759/trist-lr1:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/t2c2701bb5gdgz24xpx0bkdim

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
```

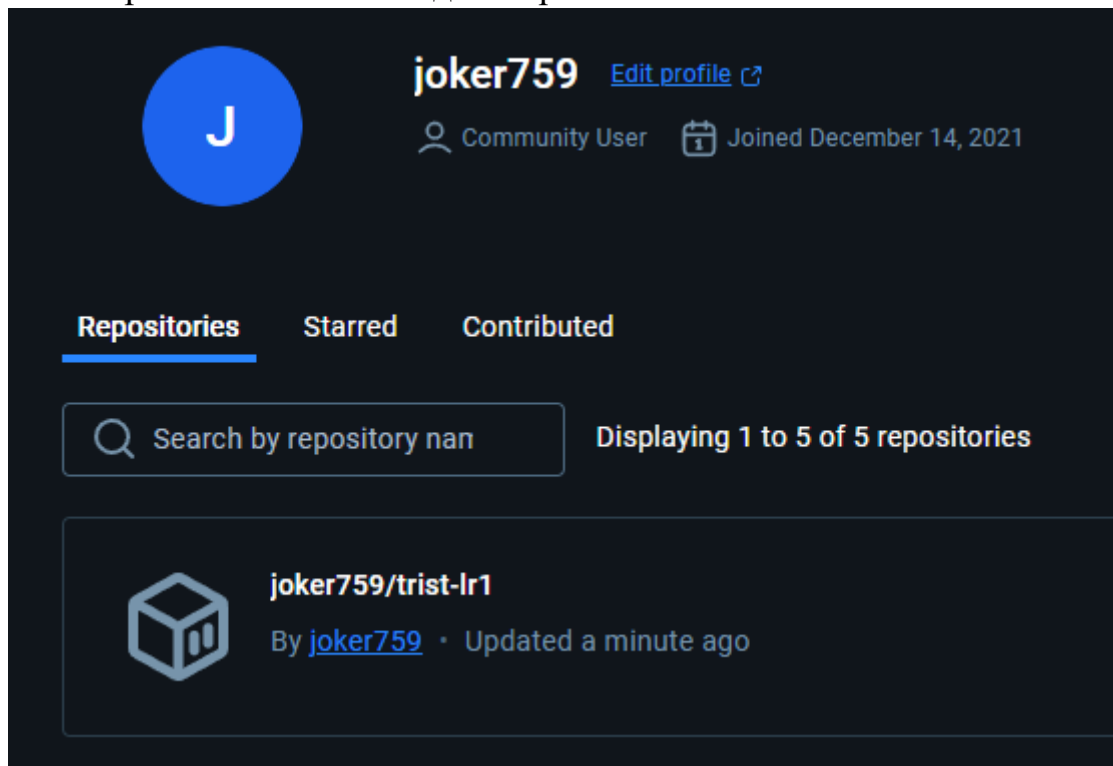
г) Перевіримо через Docker Desktop або команду docker images, що образ було створено успішно:



- д) Завантажимо створений образ у DockerHub, щоб використати його для розгортання на EC2 за допомогою команди «`docker push joker759/trist-lr1:latest`»:

```
PS E:\Univer\Sharaga 13 Term\TRIST\LR1> docker push joker759/trist-lr1:latest
The push refers to repository [docker.io/joker759/trist-lr1]
e8dba7b7b0a8: Pushed
2430c01bea64: Mounted from library/nginx
b11b58162504: Mounted from library/nginx
8b5ce426f73d: Mounted from library/nginx
884b72c14f15: Mounted from library/nginx
4a37d1b49911: Mounted from library/nginx
4e8a0009474a: Mounted from library/nginx
287563f25f8b: Mounted from library/nginx
75654b8eeebd: Mounted from library/nginx
latest: digest: sha256:2e286fe78ba80c6a3179980c58930cfd43c825eaa2d403eda20a4e9396687f0f size: 2197
```

- е) Перевіримо на DockerHub, що репозиторій із назвою `joker759/trist-lr1` створено і він містить один образ із тегом `latest`:



Explore / joker759/trist-lr1

joker759/trist-lr1
By [joker759](#) · Updated 1 minute ago

IMAGE
☆ 0 ↓ 0

Overview **Tags**

Sort by **Newest** Filter tags

| TAG | Digest | OS/ARCH | Last pull | Compressed size |
|--------|--------------|-------------|-------------------|-----------------|
| latest | 2e286fe78ba8 | linux/amd64 | a few seconds ago | 21.73 MB |

docker pull joker759/trist-lr1:latest [Copy](#)

ж) Запустимо застосунок як Docker-контейнер на EC2 за допомогою команди «`docker run -d -p 80:80 --name rist-lr1 joker759/trist-lr1:latest`»:

```
ubuntu@ip-172-31-34-233:~$ docker run -d -p 80:80 --name rist-lr1 joker759/trist-lr1:latest
Unable to find image 'joker759/trist-lr1:latest' locally
latest: Pulling from joker759/trist-lr1
da9db072f522: Pull complete
e10e486delab: Pull complete
af9c0e53c5a4: Pull complete
b2eb2b8af93a: Pull complete
e351ee5ec3d4: Pull complete
fbbf7d28be71: Pull complete
471412c08d15: Pull complete
a2eb5282fbec: Pull complete
f51a5cbb5e93: Pull complete
Digest: sha256:2e286fe78ba80c6a3179980c58930cfd43c825eaa2d403eda20a4e9396687f0f
Status: Downloaded newer image for joker759/trist-lr1:latest
8dfa3bcf36118ble12d8a330f79e313f3afab82539b0cb9f4159a6a56e6784ca
ubuntu@ip-172-31-34-233:~$ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|---------------------------|-------------------------|----------------|--------------|-----------------------------------|----------|
| 8dfa3bcf3611 | joker759/trist-lr1:latest | "/docker-entrypoint..." | 10 seconds ago | Up 9 seconds | 0.0.0.0:80->80/tcp, :::80->80/tcp | rist-lr1 |

```
ubuntu@ip-172-31-34-233:~$
```

з) Запустимо контейнер із Watchtower для автоматичного оновлення контейнерів за допомогою команди «`docker run -d --name watchtower -v /var/run/docker.sock:/var/run/docker.sock containrrr/watchtower`»:

```
ubuntu@ip-172-31-34-233:~$ docker run -d --name watchtower -v /var/run/docker.sock:/var/run/docker.sock containrrr/watchtower
Unable to find image 'containrrr/watchtower:latest' locally
latest: Pulling from containrrr/watchtower
57241801ebfd: Pull complete
3d4f475b92a2: Pull complete
1f05004da6d7: Pull complete
Digest: sha256:6dd50763bbd632a83cb154d5451700530d1e44200b268a4e9488fedfcdcf2b038
Status: Downloaded newer image for containrrr/watchtower:latest
8d0d37244bb820aade6453c2d59f51fa39f275aa825bc515830b70d0e365b83f
ubuntu@ip-172-31-34-233:~$ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS |
|--------------|---------------------------|-------------------------|--------------------|----------------------------------|-----------------------------------|
| 8d0d37244bb8 | containrrr/watchtower | "/watchtower" | 23 seconds ago | Up 22 seconds (health: starting) | 8080/tcp |
| 8dfa3bcf3611 | joker759/trist-lr1:latest | "/docker-entrypoint..." | About a minute ago | Up About a minute | 0.0.0.0:80->80/tcp, :::80->80/tcp |

```
ubuntu@ip-172-31-34-233:~$
```

и) Перевіримо, що застосунок працює коректно. На сторінці AWS у вкладці створеного EC2-інстансу знайдемо поле «Public IPv4 address» та перейдемо за відповідним посиланням:

aws

EC2 > Instances > i-01bc42345113f8120

Dashboard
EC2 Global View
Events

Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations

Images

Instance summary for i-01bc42345113f8120 (test-lr1) Info

Updated 2 minutes ago

Instance ID
i-01bc42345113f8120

IPv6 address
-

Hostname type
IP name: ip-172-31-34-233.eu-central-1.compute.internal

Answer private resource DNS name
IPv4 (A)

Auto-assigned IP address
3.121.227.25 [Public IP]

Public IPv4 address
3.121.227.25 | [open address](#)

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-34-233.eu-central-1.compute.internal

Instance type
t2.micro

VPC ID
vpc-05b5b4786229a5f4e

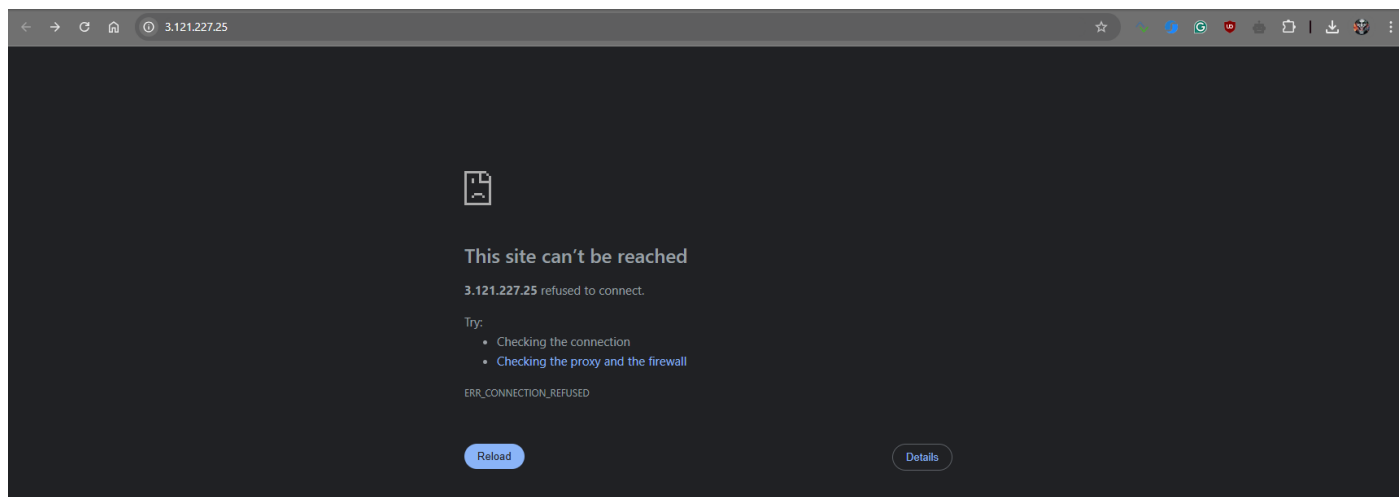
Private IPv4 addresses
172.31.34.233

Public IPv4 DNS
ec2-3-121-227-25.eu-central-1.compute.amazonaws.com | [open address](#)

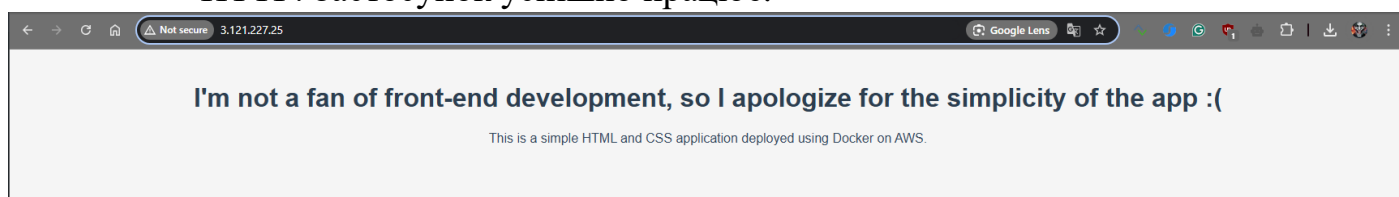
Elastic IP addresses
-

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.

[Connect](#) [Instance state](#) [Actions](#)

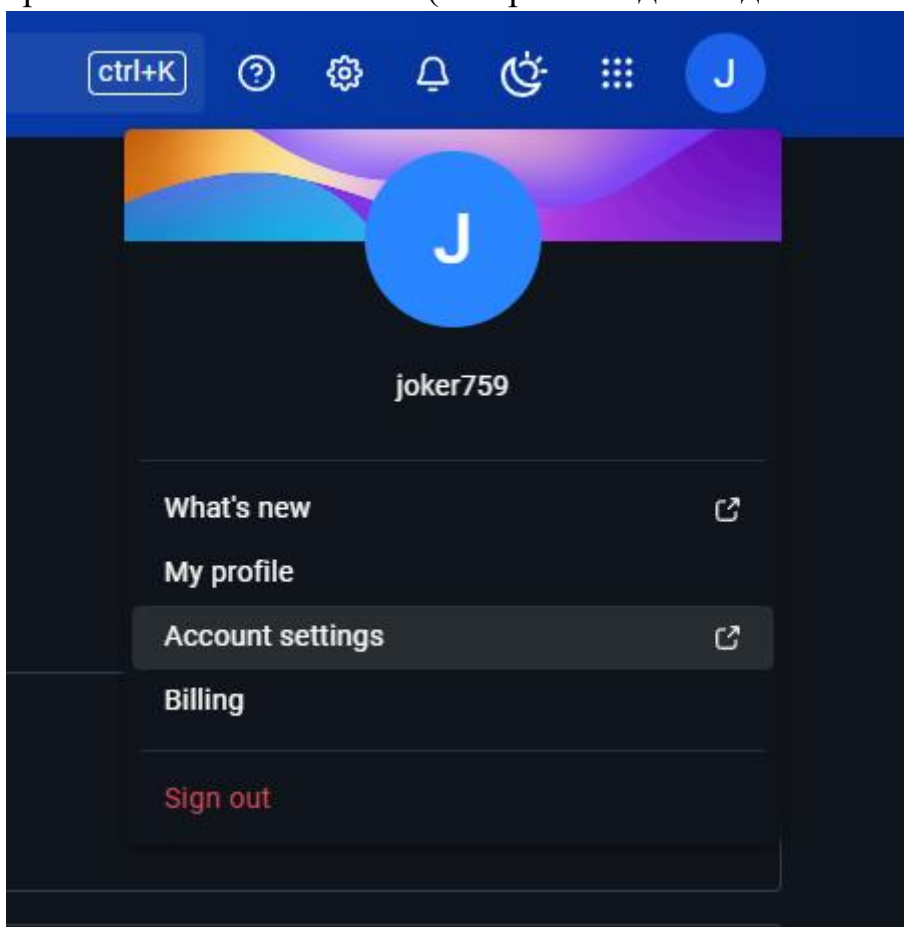


к) Якщо адреса за замовчуванням перенаправляє на HTTPS, виправимо її на HTTP. Застосунок успішно працює:



6. Написати github actions pipeline, який збиратиме docker образ і пушитиме його в docker hub при коміті в гілку main.

а) Для взаємодії з Dockerhub'ом, нам необхідно отримати токен. Для цього перейдемо в налаштування аккаунта -> Security -> Personal access tokens -> створимо новий Access token (і збережемо для подальшого використання):





joker759

Joined December 14, 2021

General

Account information

Add your account information.



Email

danilnovikov759@gmail.com ✓ VERIFIED



Password

You can change your password by initiating a reset via email. [Reset password](#)

Security

Two-factor authentication

Two factor authentication is disabled.



Personal access tokens

There is 1 personal access token associated with your account.



Connected accounts

Connect your Docker account to Google or GitHub to sign in using this account.



Account management

Convert account

Convert your account to an organization.



Deactivate account

Permanently deactivate your user account.



[Settings](#) / Personal access tokens

Personal access tokens

You can use a personal access token instead of a password for Docker CLI authentication. Create multiple tokens, control their scope, and delete tokens at any time. [Learn more](#)

[Generate new token](#)

| Description | Scope | Status | Source ⓘ | Created | Last used | Expiration date | |
|-----------------------------|---------------------|--------|----------------|--------------------------|--------------------------|-----------------|---|
| Generated through deskto... | Read, Write, Delete | Active | Auto-generated | Dec 04, 2024 at 00:59:57 | Dec 04, 2024 at 01:08:08 | Never | ⋮ |

Rows per page: 10 ▾ 1-1 of 1 < >

Create access token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#) ↗

Access token description

trist-labs

Expiration date

30 days



Optional

Access permissions

Read, Write, Delete



Read, Write, Delete tokens allow you to manage your repositories.

Cancel

Generate

Copy access token

Use this token as a password when you sign in from the Docker CLI client. [Learn more](#)

Make sure you copy your personal access token now. Your personal access token is only displayed once. It isn't stored and can't be retrieved later.

Access token description

trist-labs

Expires on

Jan 03, 2025 at 23:59:59

Access permissions

Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run

```
$ docker login -u joker759
```

Copy

2. At the password prompt, enter the personal access token.

[REDACTED]

Copy

[Back to access tokens](#)

[Settings](#) / Personal access tokens

✓ Successfully created personal access token trist-labs.

Personal access tokens

You can use a personal access token instead of a password for Docker CLI authentication. Create multiple tokens, control their scope, and delete tokens at any time. [Learn more](#)

[Generate new token](#)

| Description | Scope | Status | Source ⓘ | Created | Last used | Expiration date | |
|---------------------------|---------------------|--------|----------------|--------------------------|--------------------------|-----------------------|---|
| Generated through desk... | Read, Write, Delete | Active | Auto-generated | Dec 04, 2024 at 00:59:57 | Dec 04, 2024 at 01:08:08 | Never | ⋮ |
| trist-labs | Read, Write, Delete | Active | Manual | Dec 04, 2024 at 01:43:20 | Never | Jan 03, 2025 at 23:59 | ⋮ |

Rows per page: 10 1-2 of 2

6) Додайте DockerHub username та згенерований токен у GitHub Secrets для подальшого використання під час виконання Github actions:

- DOCKERHUB_USERNAME;
- DOCKERHUB_TOKEN.

Actions secrets / New secret

Name *

DOCKERHUB_USERNAME

Secret *

joker759



Add secret

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets

Variables

Environment secrets

This environment has no secrets.

Manage environment secrets

Repository secrets

New repository secret

Name

Last updated



DOCKERHUB_TOKEN

1 minute ago

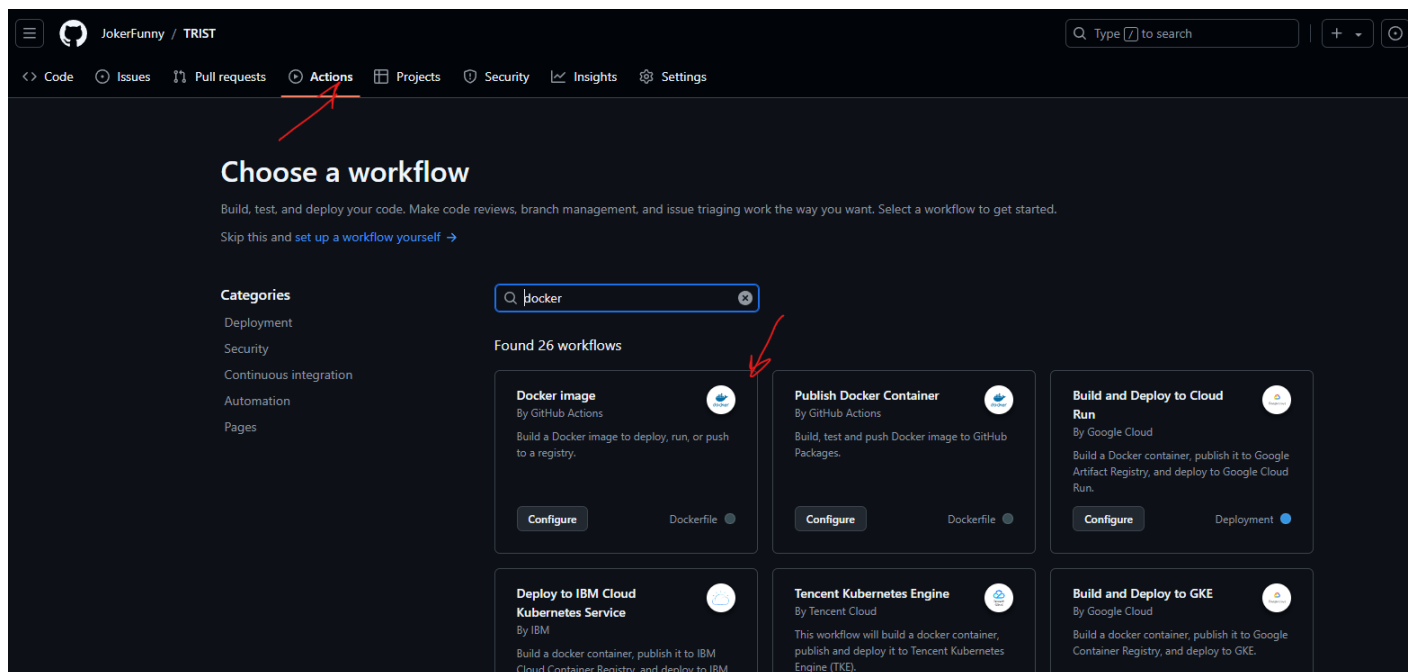


DOCKERHUB_USERNAME

now

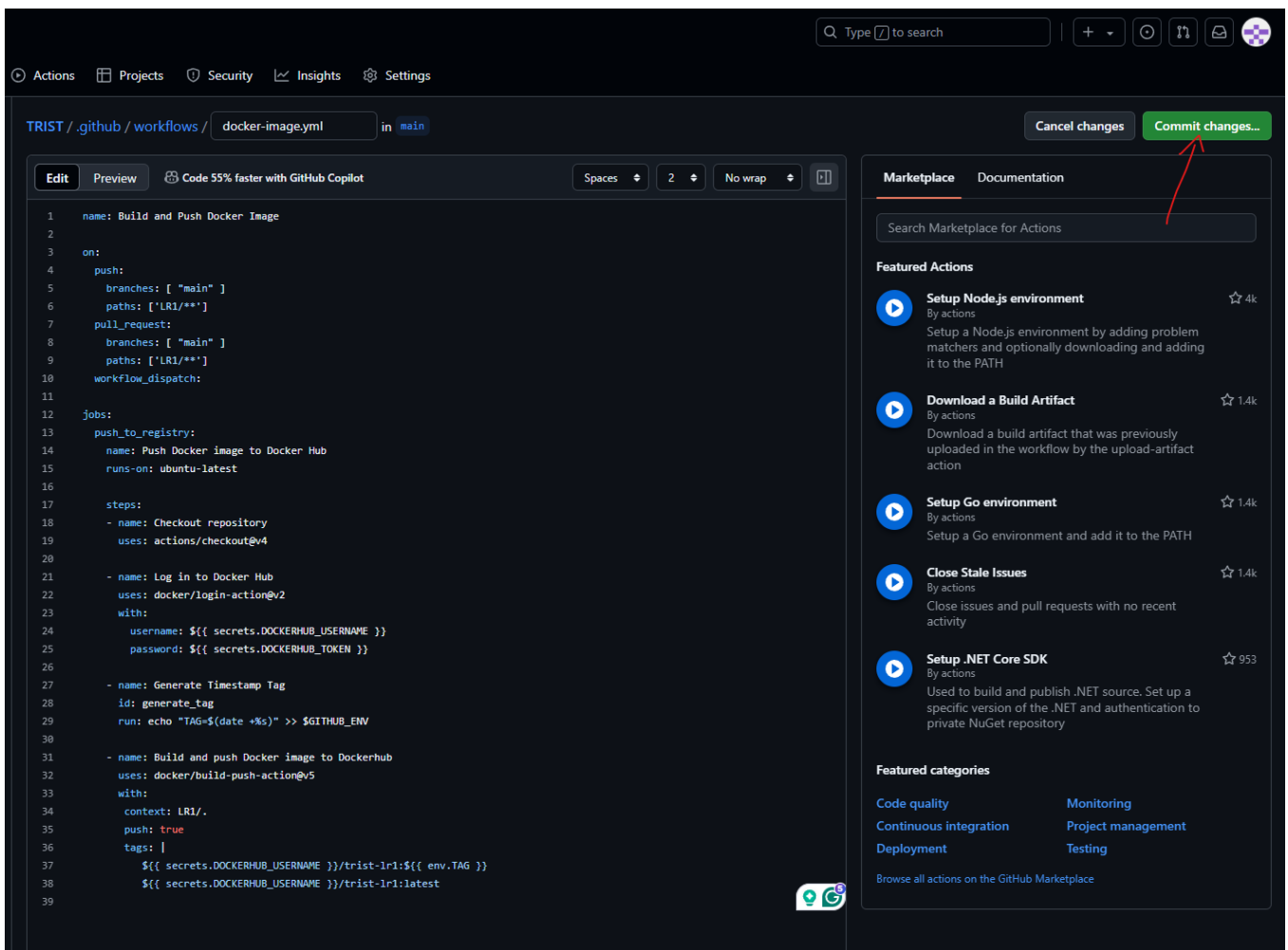


в) Створимо GitHub Actions для автоматизації процесу:

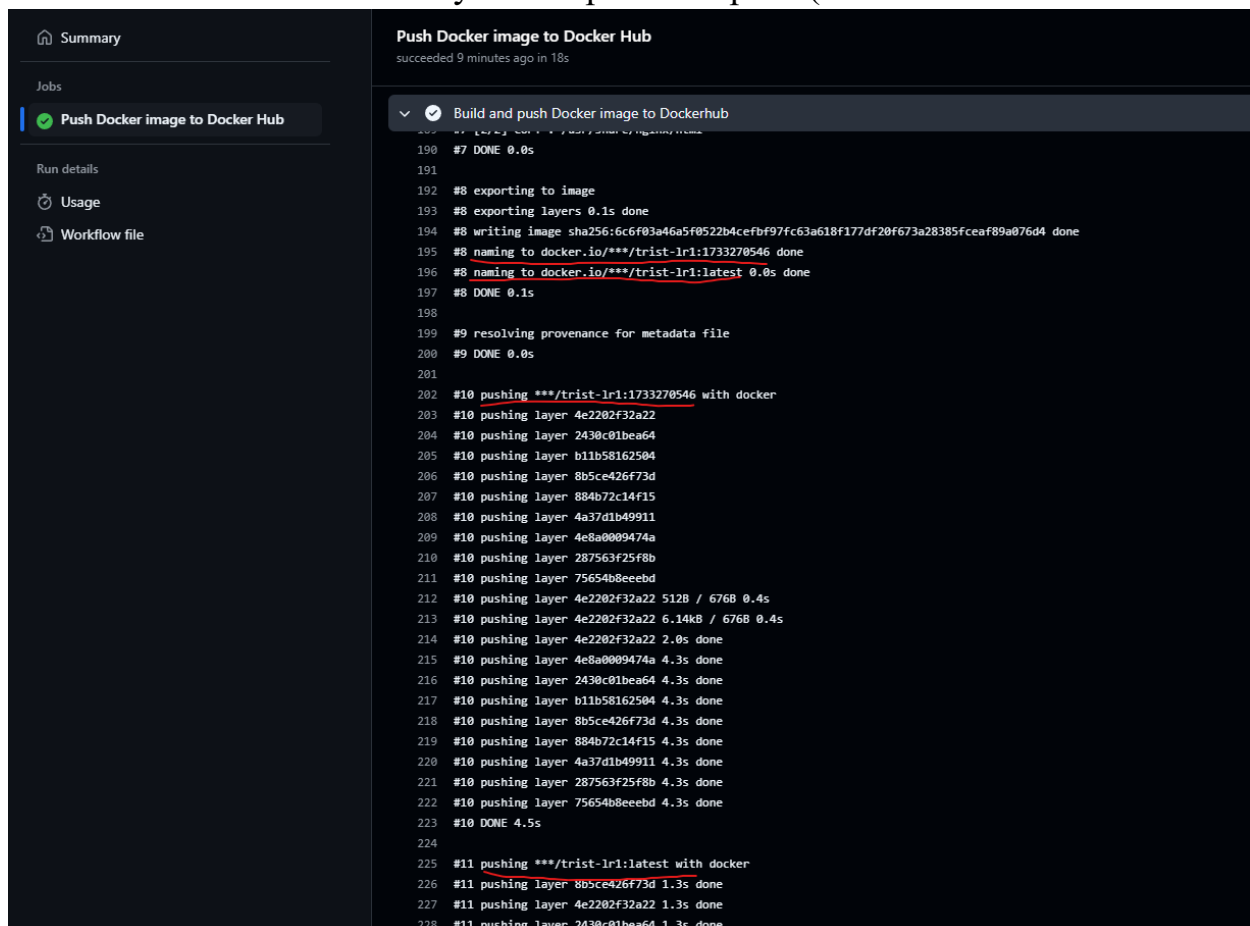


г) Додамо такі кроки у workflow:

- i. Перевірка репозиторію: використовується дія `actions/checkout`, яка завантажує код з GitHub-репозиторію, в якому знаходиться додаток та гарантує, що вся структура проекту (включно з папкою `LR1/`, де зберігаються `Dockerfile`, `index.html` та `styles.css`) буде доступна для наступних етапів;
- ii. Аутентифікація в DockerHub: використовується дія `docker/login-action`, яка виконує авторизацію в обліковому записі DockerHub, використовуючи секрети `DOCKERHUB_USERNAME` та `DOCKERHUB_TOKEN` (додані раніше). Це забезпечує можливість публікувати образи;
- iii. Генерація мітки часу для тегу Docker-образу: виконує команду `echo`, щоб створити динамічний тег для Docker-образу на основі поточного часу (у форматі Unix timestamp);
- iv. Збірка та пуш Docker-образу: використовується дія `docker/build-push-action`, яка виконує наступне:
 1. Збірка Docker-образу: використовуючи папку `LR1/` як контекст, у якій розташовані `Dockerfile` і веб-додаток (HTML + CSS), на основі `Dockerfile` створює образ, що містить веб-додаток, розміщений у контейнері NGINX;
 2. Тегування Docker-образу: додає два теги до одного й того ж образу - тег з міткою часу (наприклад, `trist-lr1:1698961321`) та тег `latest`. В результаті маємо 2 образи, що будемо публікувати в DockerHub;
 3. Публікація Docker-образу: пушить зібраний образ у DockerHub з обома тегами.



- д) Після виконання workflow переглянемо результат у розділі GitHub Actions:
- Переконаємося, що процес завершився успішно;
 - У DockerHub було створено 2 образи (із тегами latest та з міткою часу).



е) Перевіримо DockerHub, щоб переконатися, що нові образи додано з відповідними тегами:

The screenshot shows the DockerHub profile of user **joker759**. The repository **joker759/trist-lr1** is displayed with the following tags:

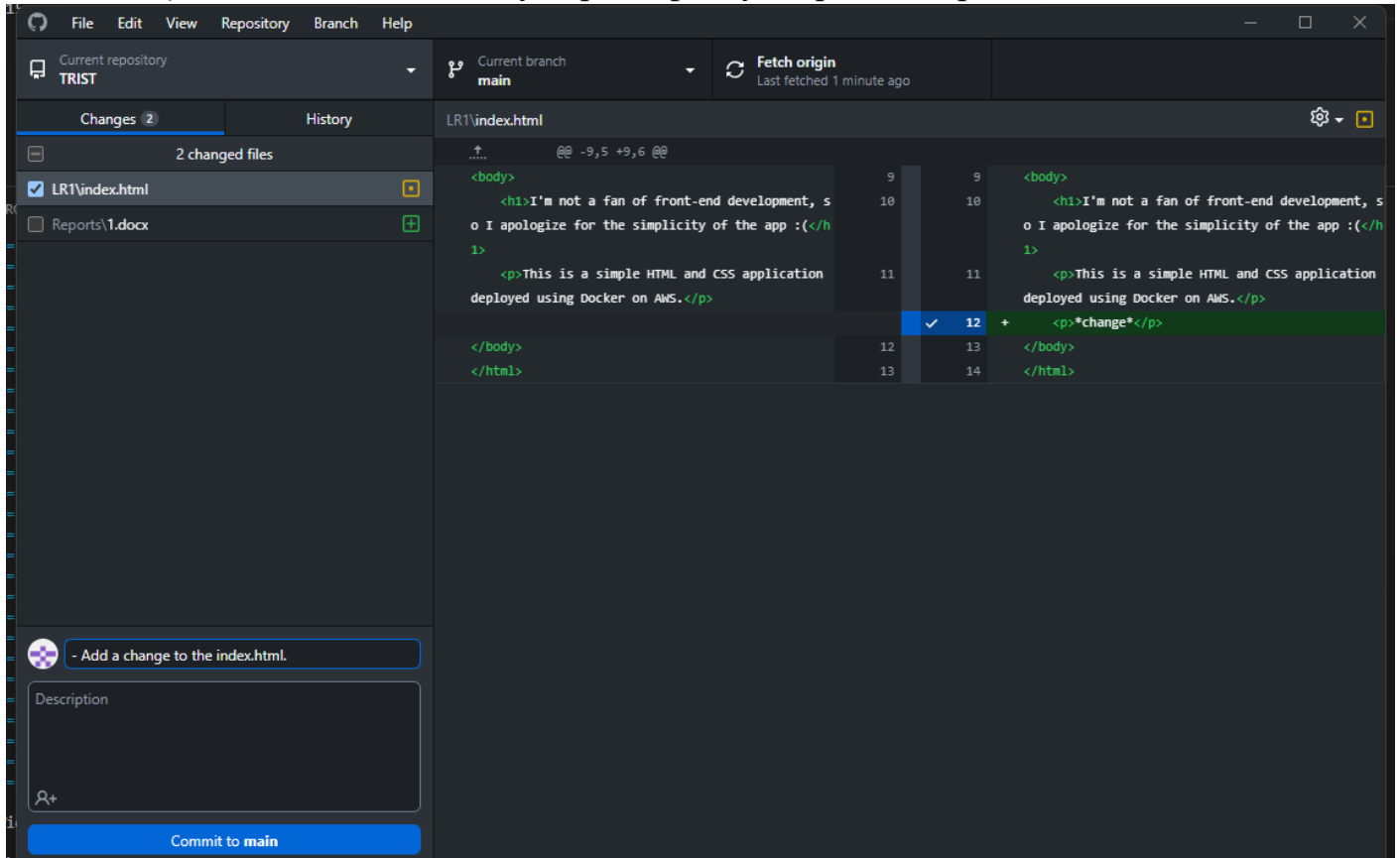
| TAG | OS/ARCH | Compressed size |
|------------|-------------|-----------------|
| latest | linux/amd64 | 21.73 MB |
| 1733270546 | linux/amd64 | 21.73 MB |
| 1733270326 | linux/amd64 | 21.73 MB |

7. Валідація результату:

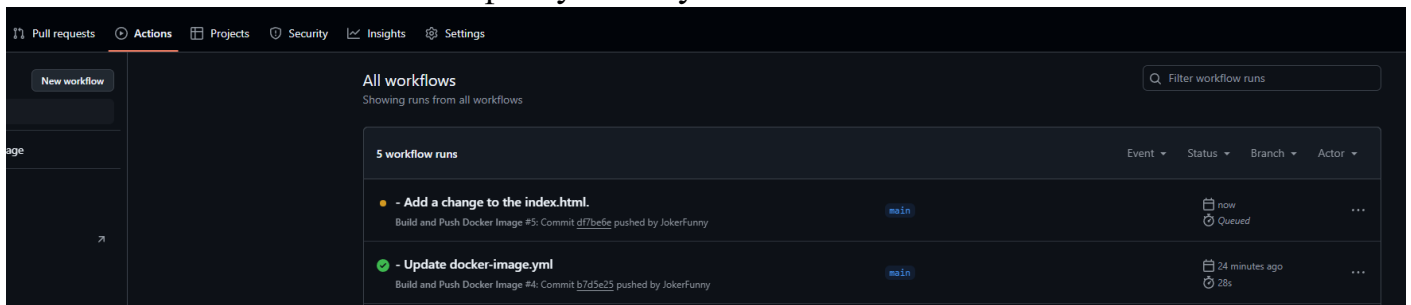
а) Зробимо зміну в застосунку та перевіримо роботу GitHub Actions, оновлення образів у DockerHub, а також відпрацювання Watchtower для автоматичного оновлення контейнера на EC2. Наприклад, додамо текст **change** до файлу `index.html`:

```
index.html M X # styles.css Dockerfile key
LR1 > <> index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device
6   <title>My Web App</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10   <h1>I'm not a fan of front-end development,
11   <p>This is a simple HTML and CSS applicatio
12   <p>*change*</p>
13 </body>
14 </html>
15
```


б) Закомітимо оновлену версію файлу до репозиторію:



в) У нас автоматично запуститься Github actions, який створить нові образи на основі зміненого файлу та запусить їх в DockerHub:



Summary

Jobs

Push Docker image to Docker Hub

Run details

Usage

Workflow file

Push Docker image to Docker Hub

succeeded 1 minute ago in 12s

Build and push Docker image to Dockerhub

186

187 #8 exporting to image

188 #8 exporting layers

189 #8 exporting layers 0.2s done

190 #8 writing image sha256:9d3718973f85c1b64701e3b37f946e6bfd3d2b09857d65bbf502cb463434321b done

191 #8 naming to docker.io/***/trist-lr1:1733272032 done

192 #8 naming to docker.io/***/trist-lr1:latest done

193 #8 DONE 0.2s

194

195 #9 resolving provenance for metadata file

196 #9 DONE 0.0s

197

198 #10 pushing ***/trist-lr1:1733272032 with docker

199 #10 pushing layer 62a15162703e

200 #10 pushing layer 2430c01bea64

201 #10 pushing layer b11b58162504

202 #10 pushing layer 8b5ce426f73d

203 #10 pushing layer 884b72c14f15

204 #10 pushing layer 4a37d1b49911

205 #10 pushing layer 4e8a0009474a

206 #10 pushing layer 287563f25f8b

207 #10 pushing layer 75654b8eeebd

208 #10 pushing layer 62a15162703e 6.14kB / 696B 0.3s

209 #10 pushing layer 62a15162703e 1.3s done

210 #10 pushing layer 8b5ce426f73d 2.9s done

211 #10 pushing layer 2430c01bea64 2.9s done

212 #10 pushing layer b11b58162504 2.9s done

213 #10 pushing layer 884b72c14f15 2.9s done

214 #10 pushing layer 4a37d1b49911 2.9s done

215 #10 pushing layer 4e8a0009474a 2.9s done

216 #10 pushing layer 287563f25f8b 2.9s done

217 #10 pushing layer 75654b8eeebd 2.9s done

218 #10 DONE 3.0s

219


220 #11 pushing ***/trist-lr1:latest with docker

221 #11 pushing layer 62a15162703e 0.6s done

222 #11 pushing layer 2430c01bea64 0.6s done

223 #11 pushing layer b11b58162504 0.6s done

224 #11 pushing layer 8b5ce426f73d 0.6s done



joker759/trist-lr1

By [joker759](#) · Updated 1 minute ago

IMAGE

☆0 ↓10

Overview

Tags

Sort by

Newest

Filter tags

TAG

[latest](#)

Last pushed a minute ago by [joker759](#)

Digest

OS/ARCH

Compressed size

[fb583b3e606d](#)

linux/amd64

21.73 MB

TAG

[1733272032](#)

Last pushed a minute ago by [joker759](#)

Digest

OS/ARCH

Compressed size

[fb583b3e606d](#)

linux/amd64

21.73 MB

TAG

[1733270546](#)

Last pushed 26 minutes ago by [joker759](#)

Digest

OS/ARCH

Compressed size

[049cbb6e13b2](#)

linux/amd64

21.73 MB

г) Підключимося до EC2 через SSH і перевіримо статус контейнерів за допомогою команди «docker ps -a»:

```

ubuntu@ip-172-31-34-233:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
8d0d37244bb8   containrrr/watchtower   "/watchtower"          2 hours ago   Up 2 hours   8080/tcp                watchtower
8dfa3bcf3611   joker759/trist-lrl:latest "/docker-entrypoint..." 2 hours ago   Up 2 hours   0.0.0.0:80->80/tcp, :::80->80/tcp   trist-lrl

ubuntu@ip-172-31-34-233:~$ docker logs 8d0d37
time="2024-12-03T23:16:17Z" level=info msg="Watchtower 1.7.1"
time="2024-12-03T23:16:17Z" level=info msg="Using no notifications"
time="2024-12-03T23:16:17Z" level=info msg="Checking all containers (except explicitly disabled with label)"
time="2024-12-03T23:16:17Z" level=info msg="Scheduling first run: 2024-12-04 23:16:17 +0000 UTC"
time="2024-12-03T23:16:17Z" level=info msg="Note that the first check will be performed in 23 hours, 59 minutes, 59 seconds"

ubuntu@ip-172-31-34-233:~$ docker rm 8d0d -f
8d0d

ubuntu@ip-172-31-34-233:~$ docker logs 8d0d37
Error response from daemon: No such container: 8d0d37

ubuntu@ip-172-31-34-233:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
8dfa3bcf3611   joker759/trist-lrl:latest "/docker-entrypoint..." 2 hours ago   Up 2 hours   0.0.0.0:80->80/tcp, :::80->80/tcp   trist-lrl

```

д) Оскільки Watchtower було запущено з дефолтними налаштуваннями, він перевіряє оновлення контейнерів раз на добу (86400 секунд). Щоб пришвидшити процес, перезапустимо контейнер Watchtower із вказанням змінного інтервалу перевірки (вказавши WATCHTOWER_POLL_INTERVAL=120, щоб він перевіряв апдейт образу інших запущених контейнерів 1 раз на 120 секунд):

```

ubuntu@ip-172-31-34-233:~$ docker run -d --name watchtower -v /var/run/docker.sock:/var/run/docker.sock -e WATCHTOWER_CLEANUP=true -e WATCHTOWER_POLL_INTERVAL=120 containrrr/watchtower
e951a1819fadd80efdd9e6147ed70975cled781c6eeea717fd1f306234f8aa5c

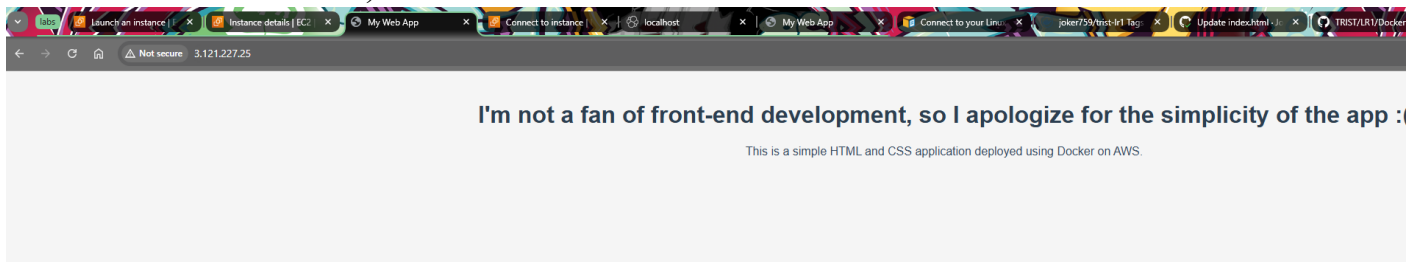
ubuntu@ip-172-31-34-233:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
e951a1819fad   containrrr/watchtower   "/watchtower"          3 seconds ago   Up 2 seconds   8080/tcp                watchtower
8dfa3bcf3611   joker759/trist-lrl:latest "/docker-entrypoint..." 2 hours ago   Up 2 hours   0.0.0.0:80->80/tcp, :::80->80/tcp   trist-lrl

ubuntu@ip-172-31-34-233:~$ docker logs e951a1
time="2024-12-04T00:56:34Z" level=info msg="Watchtower 1.7.1"
time="2024-12-04T00:56:34Z" level=info msg="Using no notifications"
time="2024-12-04T00:56:34Z" level=info msg="Checking all containers (except explicitly disabled with label)"
time="2024-12-04T00:56:34Z" level=info msg="Scheduling first run: 2024-12-04 00:58:34 +0000 UTC"
time="2024-12-04T00:56:34Z" level=info msg="Note that the first check will be performed in 1 minute, 59 seconds"

ubuntu@ip-172-31-34-233:~$

```

е) Перевіримо, що на EC2 поки що використовується стара версія додатку (до оновлення):



ж) Почекаємо декілька хвилин (приблизно 2 хвилини на перевірку Watchtower + час на перезапуск контейнера з новим образом). Після цього перевіримо:

i. Логи контейнеру Watchtower за допомогою команди «docker logs <container_id>»:

```

ubuntu@ip-172-31-34-233:~$ docker logs e951a1
time="2024-12-04T00:56:34Z" level=info msg="Watchtower 1.7.1"
time="2024-12-04T00:56:34Z" level=info msg="Using no notifications"
time="2024-12-04T00:56:34Z" level=info msg="Checking all containers (except explicitly disabled with label)"
time="2024-12-04T00:56:34Z" level=info msg="Scheduling first run: 2024-12-04 00:58:34 +0000 UTC"
time="2024-12-04T00:56:34Z" level=info msg="Note that the first check will be performed in 1 minute, 59 seconds"
time="2024-12-04T00:58:37Z" level=info msg="Found new joker759/trist-lrl:latest image (ef0401b803f0)"
time="2024-12-04T00:58:37Z" level=info msg="Stopping /trist-lrl (8dfa3bcf3611) with SIGTERM"
time="2024-12-04T00:58:38Z" level=info msg="Creating /trist-lrl"
time="2024-12-04T00:58:39Z" level=info msg="Removing image c902af4fa448"
time="2024-12-04T00:58:39Z" level=info msg="Session done" Failed=0 Scanned=2 Updated=1 notify=no

ubuntu@ip-172-31-34-233:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
5e8fbf12142d   joker759/trist-lrl:latest "/docker-entrypoint..." 57 seconds ago   Up 56 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp   trist-lrl
e951a1819fad   containrrr/watchtower   "/watchtower"          3 minutes ago   Up 3 minutes   8080/tcp                watchtower

ubuntu@ip-172-31-34-233:~$

```

ii. Роботу застосунку через браузер або інший клієнт за публічною IP-адресою EC2 (маємо побачити оновлену версію):



з) Перевіримо логи після оновлення контейнера (коли немає нових змін):

```

ubuntu@ip-172-31-34-233:~$ docker logs e951al
time="2024-12-04T00:56:34Z" level=info msg="Watchtower 1.7.1"
time="2024-12-04T00:56:34Z" level=info msg="Using no notifications"
time="2024-12-04T00:56:34Z" level=info msg="Checking all containers (except explicitly disabled with label)"
time="2024-12-04T00:56:34Z" level=info msg="Scheduling first run: 2024-12-04 00:58:34 +0000 UTC"
time="2024-12-04T00:56:34Z" level=info msg="Note that the first check will be performed in 1 minute, 59 seconds"
time="2024-12-04T00:58:37Z" level=info msg="Found new joker759/trist-lrl:latest image (ef0401b803f0)"
time="2024-12-04T00:58:37Z" level=info msg="Stopping /rist-lrl (8dfa3bcf3611) with SIGTERM"
time="2024-12-04T00:58:38Z" level=info msg="Creating /rist-lrl"
time="2024-12-04T00:58:39Z" level=info msg="Removing image c902af4fa448"
time="2024-12-04T00:58:39Z" level=info msg="Session done" Failed=0 Scanned=2 Updated=1 notify=no
time="2024-12-04T01:00:35Z" level=info msg="Session done" Failed=0 Scanned=2 Updated=0 notify=no
ubuntu@ip-172-31-34-233:~$ docker ps -a

```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|---------------------------|-------------------------|---------------|------------------------|-----------------------------------|------------|
| 5e8fbf12142d | joker759/trist-lrl:latest | "/docker-entrypoint..." | 3 minutes ago | Up 3 minutes | 0.0.0.0:80->80/tcp, :::80->80/tcp | rist-lrl |
| e951al919fad | containrrr/watchtower | "/watchtower" | 5 minutes ago | Up 5 minutes (healthy) | 8080/tcp | watchtower |

```

ubuntu@ip-172-31-34-233:~$

```

Висновки: в результаті виконання цієї лабораторної роботи було ознайомлено з базовими концепціями автоматизації розгортання додатків і їх оновлення за допомогою GitHub Actions, Docker, DockerHub та Watchtower.

На основі отриманих знань було реалізовано практичну частину, яка полягала у створенні та автоматизації процесу збірки, публікації й оновлення Docker-образів.

Було успішно налаштовано оновлення застосунку в середовищі AWS EC2 з використанням Watchtower, що забезпечує автоматичне оновлення контейнерів на основі змінених Docker-образів.

Вихідний код застосунку можна знайти за наступним посиланням на [GitHub](#).

Github actions pipeline доступний за посиланням *[посилання](#)*.