



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 №2
Тенденції розвитку інформаційних систем та технологій
Інфраструктура як код. Terraform.

Виконав
студент групи ІТ-41ф

Новиков Д. М.

Перевірив:

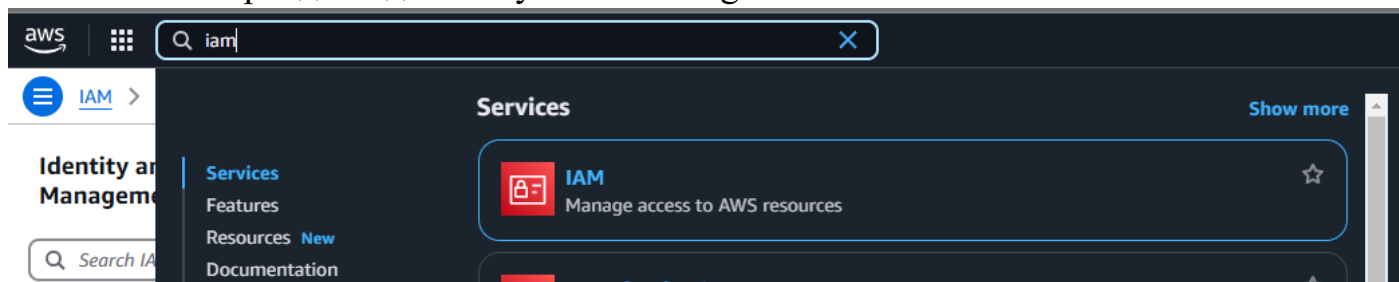
ас. Цимбал С. І.

Мета роботи: перенести створення інфраструктури з минулої лабораторної роботи в код.

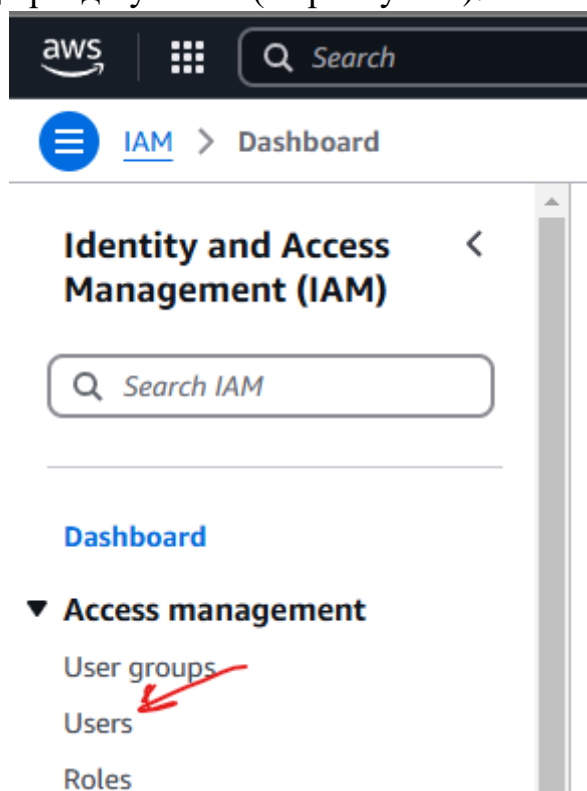
Хід роботи:

1. Створити ключ-файл, за допомогою якого будете підключатися до інфраструктури (можна використати з минулої лабораторної роботи): Використаємо ключ з попередньої ЛР.
2. Створити IAM ключ, за допомогою якого буде доступ до створення інстансів, та зберегти ID та KEY згідно з інструкцією, з яким ви ознайомились під час опрацювання матеріалу:

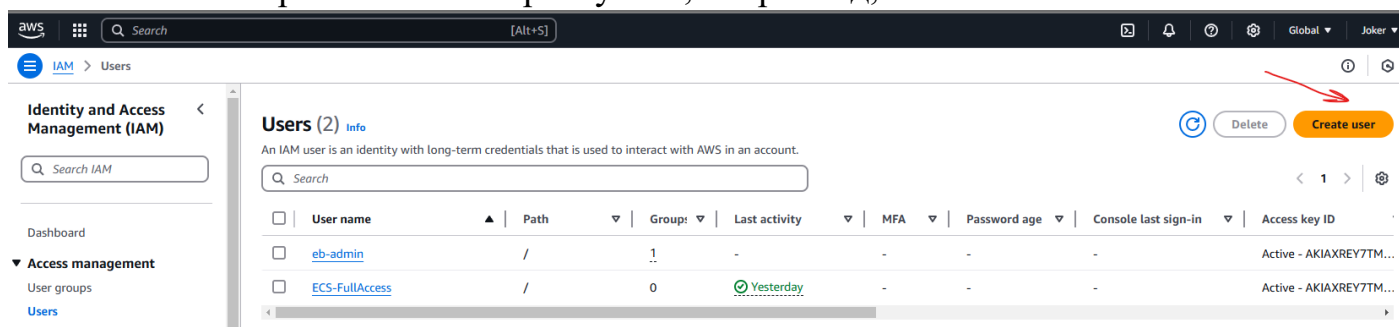
а. Перейдемо до IAM у AWS Management Console:



б. Перейдемо до розділу Users (Користувачі):



с. Створимо нового користувача, наприклад, terraform-user:



aws [Search] [Alt+S]

IAM > Users > Create user

Step 1 **Specify user details**
Step 2 Set permissions
Step 3 Review and create

Specify user details

User details

User name
terraform-user

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , _ - (hyphen)

☐ Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Info If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

- d. Щоб уникнути надання повного доступу (навіть для лабораторної роботи, оскільки це суперечить best practices), створимо нову policy із мінімально необхідними правами:

aws [Search] [Alt+S]

IAM > Users > Create user

Step 1 Specify user details
Step 2 **Set permissions**
Step 3 Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ **Attach policies directly**
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1317)
Choose one or more policies to attach to your new user.

Create policy

Filter by Type
ec2 44 matches

aws [Search] [Alt+S]

IAM > Policies > Create policy

Step 1 **Specify permissions**
Step 2 Review and create

Specify permissions

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

Visual JSON Actions

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "ec2:RunInstances",
8         "ec2:TerminateInstances",
9         "ec2:StartInstances",
10        "ec2:StopInstances",
11        "ec2:DescribeInstances",
12        "ec2:DescribeInstanceStatus",
13        "ec2:ModifyInstanceAttribute",
14        "ec2:CreateSecurityGroup",
15        "ec2:DeleteSecurityGroup",
16        "ec2:AuthorizeSecurityGroupIngress",
17        "ec2:RevokeSecurityGroupIngress",
18        "ec2:AuthorizeSecurityGroupEgress",
19        "ec2:RevokeSecurityGroupEgress",
20        "ec2:DescribeSecurityGroups"
21      ],
22       "Resource": "*"
23     },
24     {
25       "Effect": "Allow",
26       "Action": [
27         "ec2:DescribeRegions"
28       ],
29       "Resource": "*"
30     }
31   ]
32 }
```

+ Add new statement

JSON Ln 8, Col 33

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

5609 of 6144 characters remaining

Cancel Next

aws [Search] [Alt+S] Global Joker

IAM > Policies > Create policy

Step 1 Specify permissions
Step 2 Review and create

Review and create

Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

ec2-terraform

Maximum 128 characters. Use alphanumeric and '+', '@', '-' characters.

Description - optional
Add a short explanation for this policy.

Access to manage EC2 in enough amount to be used by Terraform.

Maximum 1,000 characters. Use alphanumeric and '+', '@', '-' characters.

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (1 of 437 services) Show remaining 436 services

Service	Access level	Resource	Request condition
EC2	Limited: List, Write	All resources	None

Add tags - optional

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

Key Value - optional

purpose trist-lab2 Remove

Add new tag

You can add up to 49 more tags.

Cancel Previous Create policy

е. Повернемося до процесу створення користувача IAM, оберемо новостворену policy, перейдемо далі та завершимо створення користувача:

aws [Search] [Alt+S] Global Joker

IAM > Users > Create user

Step 1 Specify user details
Step 2 Set permissions
Step 3 Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1318)

Choose one or more policies to attach to your new user.

Filter by Type

ec2- All types 1 match

Policy name	Type	Attached entities
ec2-terraform	Customer managed	0

Set permissions boundary - optional

Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. [Learn more](#)

☐ Use a permissions boundary to control the maximum permissions
You can select one of the existing permissions policies to define the boundary.

Cancel Previous Next

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name
terraform-user

Console password type
None

Require password reset
No

Permissions summary

< 1 >

Name 



Type



Used as



[ec2-terraform](#)


Customer managed

Permissions policy

Tags - optional


Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

Key

 purpose



Value - optional

 trist-lab2



Remove

Add new tag

You can add up to 49 more tags.

Cancel

Previous

Create user

f. Створимо Access Key для нового користувача:

Users (3) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.




Delete

Create user

 Search

< 1 > 

<input type="checkbox"/>	User name	Path	Group	Last activity	MFA	Password age	Console last sign-in	Access key ID
<input type="checkbox"/>	eb-admin	/	1	-	-	-	-	Active - AKIAHREY7TM...
<input type="checkbox"/>	ECS-FullAccess	/	0	 Yesterday	-	-	-	Active - AKIAHREY7TM...
<input type="checkbox"/>	terraform-user	/	0	-	-	-	-	-

Summary

ARN

 `arn:aws:iam::517864397577:user/terraform-user`

Created

December 04, 2024, 22:07 (UTC+02:00)

Console access

Disabled

Last console sign-in

-

Access key 1

[Create access key](#)

[Permissions](#)

[Groups](#)

[Tags](#)
(1)

[Security credentials](#)

[Last Accessed](#)

Console sign-in

[Enable console access](#)

Console sign-in link

 `https://517864397577.signin.aws.amazon.com/console`

Console password

Not enabled

Multi-factor authentication (MFA) (0)

[Remove](#)

[Resync](#)

[Assign MFA device](#)

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. [Learn more](#)

Type	Identifier	Certifications	Created on
No MFA devices. Assign an MFA device to improve the security of your AWS environment			
Assign MFA device			

Access keys (0)

[Create access key](#)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

[Create access key](#)

Step 1

Access key best practices & alternatives

Step 2 - optional

Set description tag

Step 3

Retrieve access keys

Access key best practices & alternatives [Info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

☐ **Command Line Interface (CLI)**

You plan to use this access key to enable the AWS CLI to access your AWS account.

☐ **Local code**

You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ **Application running on an AWS compute service**

You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☒ **Third-party service**

You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ **Application running outside AWS**

You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

☐ **Other**

Your use case is not listed here.

⚠ Alternative recommended

As a best practice, use temporary security credentials (IAM roles) instead of creating long-term credentials like access keys, and don't create AWS account root user access keys. [Learn more](#)

Confirmation

☒ I understand the above recommendation and want to proceed to create an access key.

[Cancel](#)

[Next](#)

Set description tag - optional [Info](#)

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

trist-lab2

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

[Cancel](#)

[Previous](#)

[Create access key](#)

(ВАЖЛИВО: на цьому кроці обов'язково збережемо отримані значення Access Key ID та Secret Access Key, оскільки їх неможливо переглянути повторно!)

✓ Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

- Step 1
Access key best practices & alternatives
- Step 2 - optional
Set description tag
- Step 3
Retrieve access keys

Retrieve access keys [Info](#)

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAHREY7TMEXLPTFZPZ	***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

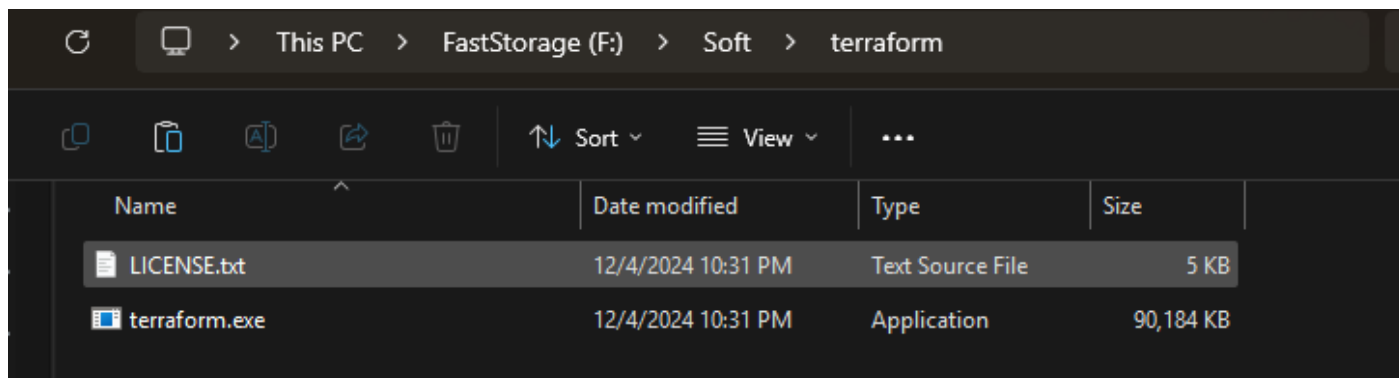
[Download .csv file](#)
[Done](#)

3. Створити main.tf файл, в якому буде описано всю інфраструктуру застосунку (створення інстансу та Security group з минулої л/р). Використовуйте репозиторій з минулої л/р для розміщення в ньому необхідних файлів.

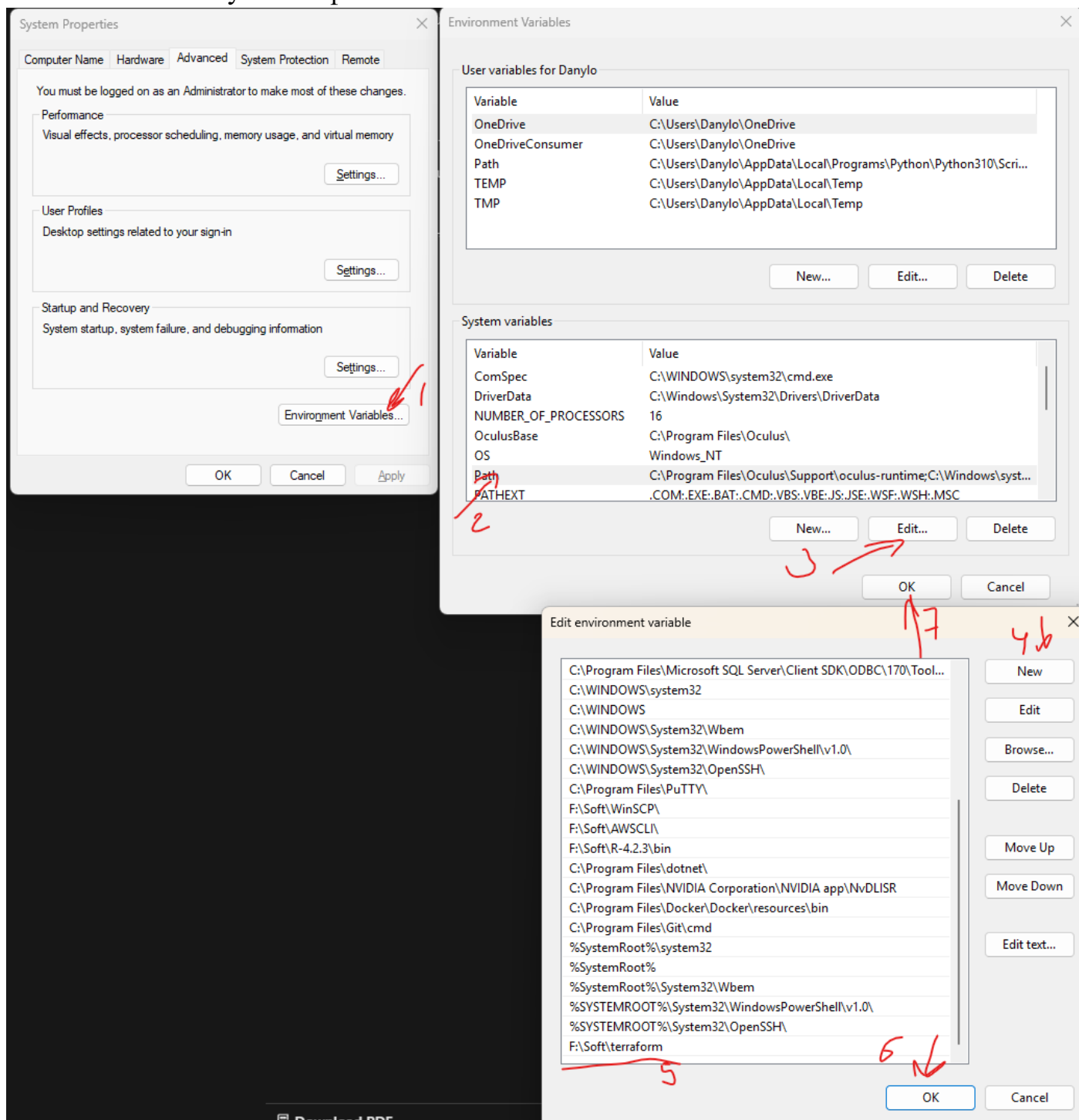
а. Для виконання дій із Terraform потрібно виконати попередні налаштування:

- Встановити Terraform CLI (версія 1.2.0+);
- Встановити AWS CLI;
- Мати AWS обліковий запис та облікові дані, що дозволяють створювати ресурси (виконано на попередньому кроці).

б. Завантажимо та встановимо Terraform із офіційного сайту <https://developer.hashicorp.com/terraform/install>:



с. Оновимо змінну середовища PATH, додавши шлях до каталогу з виконуваним файлом Terraform:



d. Перевіримо встановлення Terraform за допомогою PowerShell, виконавши команду «terraform -version» (запустіть новий інстанс PowerShell після змінення змінних середовища):

```
PS C:\WINDOWS\system32> terraform -version
Terraform v1.10.1
on windows_amd64
PS C:\WINDOWS\system32>
```

- е. Перед виконанням Terraform-команд потрібно налаштувати доступ до AWS. Для того щоб взаємодіяти з AWS з локального пристрою встановимо AWS CLI з офіційного сайту <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>:

The screenshot shows the AWS Command Line Interface (CLI) installation guide for Windows. The page is titled "AWS Command Line Interface" and "User Guide for Version 2". The left sidebar contains a navigation menu with the following items: "About the AWS CLI", "Get started" (expanded), "Prerequisites", "Install/Update" (highlighted), "Past releases", "Build and install from source", "Amazon ECR Public/Docker", "Setup", "Configure the AWS CLI", "Authentication and access credentials", "Using the AWS CLI", "Code examples", "Security", "Troubleshoot errors", "Migration guide", "Uninstall", and "Document History". The main content area is titled "Windows" and "Install and update requirements". It lists two requirements: "We support the AWS CLI on Microsoft-supported versions of 64-bit Windows." and "Admin rights to install software". Below this, the section "Install or update the AWS CLI" explains that to update the current installation, a new installer should be downloaded each time an update is released. It provides a link to the "AWS CLI version 2 Changelog" on GitHub. The first step is to download and run the AWS CLI MSI installer for Windows (64-bit), with a link to <https://awscli.amazonaws.com/AWSCLIV2.msi>. Alternatively, the `msiexec` command can be used to run the MSI installer. A code block shows the command: `C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi`. For various parameters that can be used with `msiexec`, it refers to the `msiexec` on the Microsoft Docs website. For example, the `/qn` flag can be used for a silent installation. A second code block shows the command: `C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi /qn`. The second step is to confirm the installation by opening the Start menu, searching for `cmd` to open a command prompt window, and at the command prompt using the `aws --version` command. A code block shows the output: `C:\> aws --version`
`aws-cli/2.19.1 Python/3.11.6 Windows/10 exe/AMD64 prompt/off`. At the bottom, it notes that if Windows is unable to find the program, the user might need to close and reopen the command prompt window to refresh the path, or follow the troubleshooting in [Troubleshooting errors for the AWS CLI](#).

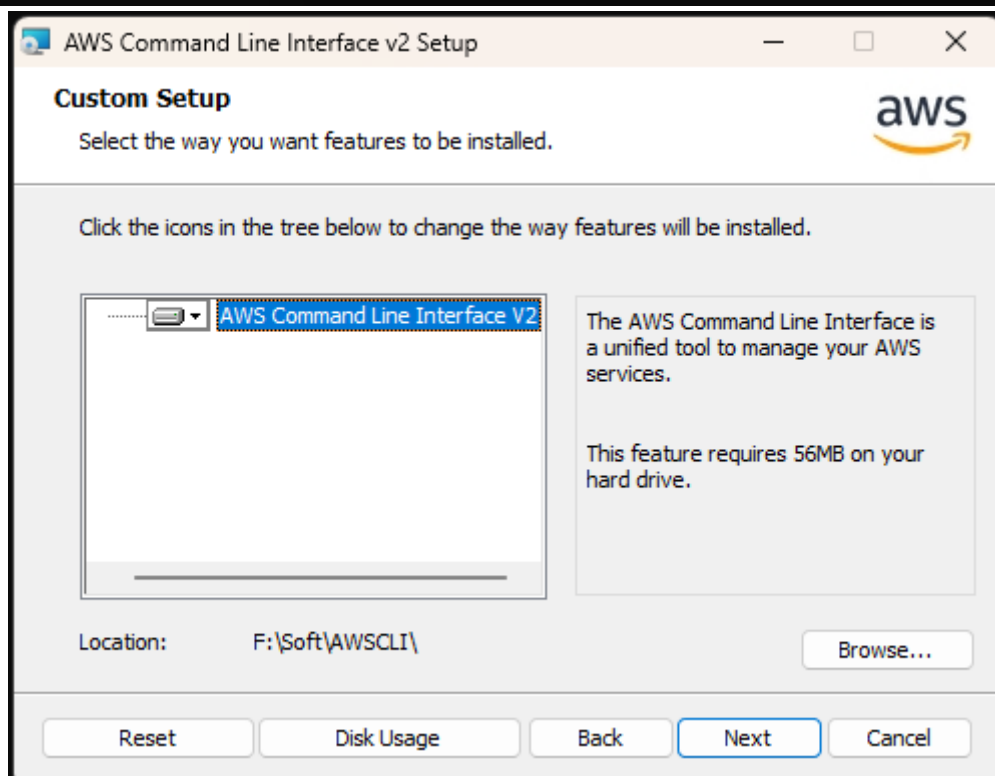
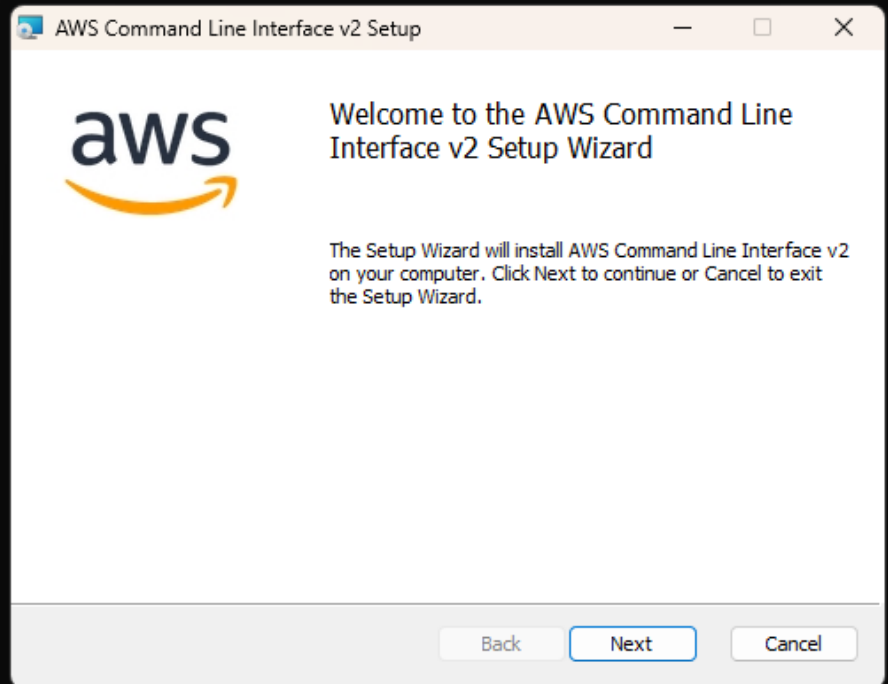
Administrator: Windows PowerShell

Terraform v1.10.1

on windows_amd64

PS C:\WINDOWS\system32> msixexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi

PS C:\WINDOWS\system32>




```

# Вхідний трафік (ingress rules).
ingress {
    description = "Allow SSH traffic from your IP"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["212.110.138.79/32"] # Мій IP адрес (якщо вже брати аналогію з минулою ЛР).
}

ingress {
    description = "Allow HTTPS traffic from anywhere"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"] # HTTPS з будь-якого місця.
}

ingress {
    description = "Allow HTTP traffic from anywhere"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"] # HTTP з будь-якого місця.
}

# Вихідний трафік (egress rules).
egress {
    description = "Allow all outbound traffic"
    from_port   = 0
    to_port     = 0
    protocol    = "-1" # Всі протоколи.
    cidr_blocks = ["0.0.0.0/0"] # Вихідний трафік до будь-якого місця.
}

tags = {
    Name = "trist-lab2"
}
}

# Створення EC2 інстансу.
resource "aws_instance" "trist_lab2_web" {
    ami           = "ami-0084a47cc718c111a" # ubuntu-noble-24.04-amd64-server-20240927.
    instance_type = "t2.micro"
    key_name      = "keys-lr1"

    security_groups = [aws_security_group.trist_lab2_sg.name]

    # Встановлення Docker, запуск застосунку з ЛР1 + watchtower.
    user_data = <<EOF
#!/bin/bash
sudo apt update
sudo apt install -y docker.io
sudo usermod -aG docker $USER
newgrp docker
docker run -d -p 80:80 --name rist-lr1 joker759/trist-lr1:latest

```

```

docker ps -a
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
docker ps -a
docker logs watchtower
EOF
}

tags = {
  Name = "trist-lab2"
}

# Вихідні дані: Public IP
output "instance_ip" {
  description = "Public IP of EC2 instance"
  value       = aws_instance.trist_lab2_web.public_ip
}

# Вихідні дані: Application URL
output "nginx_url" {
  description = "Address of the deployed application"
  value       = "http://${aws_instance.trist_lab2_web.public_ip}"
}

```

- j. Виконаємо ініціалізацію Terraform плагінів за допомогою команди «terraform init»:

```

Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.80.0...
- Installed hashicorp/aws v5.80.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

```

- k. Переглянемо план розгортання за допомогою команди «terraform plan»:

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

```
# aws_instance.trist_lab2_web will be created
+ resource "aws_instance" "trist_lab2_web" {
  + ami                  = "ami-0084a47cc718c111a"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count        = (known after apply)
  + cpu_threads_per_core  = (known after apply)
  + disable_api_stop      = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized         = (known after apply)
  + get_password_data     = false
  + host_id               = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile   = (known after apply)
  + id                    = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle     = (known after apply)
  + instance_state         = (known after apply)
  + instance_type          = "t2.micro"
  + ipv6_address_count     = (known after apply)
  + ipv6_addresses        = (known after apply)
  + key_name               = "keys-lr1"
  + monitoring             = (known after apply)
  + subnet_id              = (known after apply)
  + tags                   = {}
  + vpc_id                 = (known after apply)
}
```

1. Застосуємо конфігурацію за допомогою команди «terraform apply»:

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbol
+ create

Terraform will perform the following actions:

```
# aws_instance.trist_lab2_web will be created
+ resource "aws_instance" "trist_lab2_web" {
  + ami                  = "ami-0084a47cc718c111a"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count        = (known after apply)
  + cpu_threads_per_core  = (known after apply)
  + disable_api_stop      = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized         = (known after apply)
  + get_password_data     = false
  + host_id               = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile   = (known after apply)
  + id                    = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle     = (known after apply)
}
```

...

```
+ Name = trist-lab2
}
+ vpc_id = (known after apply)
}
```

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:

```
+ instance_ip = (known after apply)
```

Do you want to perform these actions?

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

```

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_security_group.trist_lab2_sg: Creating...
aws_security_group.trist_lab2_sg: Creation complete after 2s [id=sg-0e9cf38798bb04842]
aws_instance.trist_lab2_web: Creating...
aws_instance.trist_lab2_web: Still creating... [10s elapsed]

Error: reading EC2 Instance Type (t2.micro): operation error EC2: DescribeInstanceTypes, https response error StatusCode: 403, RequestID: 38fc7621-e7ea-47bd-a8cb-2d8b09331e62, api error UnauthorizedOperation: You are not authorized to perform this operation. User: arn:aws:iam::517864397577:user/terraform-user is not authorized to perform: ec2:DescribeInstanceTypes because no identity-based policy allows the ec2:DescribeInstanceTypes action

with aws_instance.trist_lab2_web
on main.tf line 47, in resource "aws_instance" "trist_lab2_web":
47: resource "aws_instance" "trist_lab2_web" {

```

m. Модифікуємо створену раніше політику ec2-terraform-policy, додавши необхідні права:

- i. ec2:DescribeInstanceTypes;
- ii. ec2:CreateTags;
- iii. ec2:DescribeTags;
- iv. ec2:DescribeInstanceAttribute;
- v. ec2:DescribeVolumes;
- vi. ec2:DescribeInstanceCreditSpecifications;
- vii. ec2:DescribeNetworkInterfaces.

IAM > Policies > ec2-terraform

entity and Access management (IAM)

Search IAM

shboard

cess management

pr groups

rs

es

licies

ntity providers

ount settings

ct access management [New](#)

cess reports

ess Analyzer

External access

Unused access

Analyzer settings

idential report

ec2-terraform [Info](#)

Access to manage EC2 in enough amount to be used by Terraform.

[Edit](#) [Delete](#)

Policy details

Type	Creation time	Edited time	ARN
Customer managed	December 04, 2024, 22:05 (UTC+02:00)	December 04, 2024, 22:05 (UTC+02:00)	arn:aws:iam::517864397577:policy/ec2-terraform

[Permissions](#) [Entities attached](#) [Tags](#) [Policy versions \(1\)](#) [Last Accessed](#)

Permissions defined in this policy [Info](#)

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Q Search

Allow (1 of 437 services) [Show remaining 436 services](#)

Service	Access level	Resource	Request condition
EC2	Limited: List, Write	All resources	None

[Edit](#) [Summary](#) [JSON](#)

Modify permissions in ec2-terraform [Info](#)

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

[Visual](#) [JSON](#) [Actions](#) [Add](#)

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "ec2:RunInstances",
8         "ec2:TerminateInstances",
9         "ec2:StartInstances",
10        "ec2:StopInstances",
11        "ec2:DescribeInstances",
12        "ec2:DescribeInstanceStatus",
13        "ec2:ModifyInstanceAttribute",
14        "ec2:CreateSecurityGroup",
15        "ec2>DeleteSecurityGroup",
16        "ec2:AuthorizeSecurityGroupIngress",
17        "ec2:RevokeSecurityGroupIngress",
18        "ec2:AuthorizeSecurityGroupEgress",
19        "ec2:RevokeSecurityGroupEgress",
20        "ec2:DescribeSecurityGroups",
21        "ec2:DescribeInstanceTypes",
22        "ec2:CreateTags",
23        "ec2:DescribeTags",
24        "ec2:DescribeInstanceAttribute",
25        "ec2:DescribeVolumes",
26        "ec2:DescribeInstanceCreditSpecifications",
27        "ec2:DescribeNetworkInterfaces"
28      ],
29    }
30  ]
31 }

```

[+ Add new statement](#)

Edit statement [Remove](#)

Add actions

Choose a service

Q Filter services

Included

EC2

Available

AI Operations

AMP

API Gateway

API Gateway V2

ARC Zonal Shift

ASC

Access Analyzer

Add a resource [Add](#)

Add a condition (optional) [Add](#)

JSON Ln 21, Col 0

5416 of 6144 characters remaining

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

[Check for new access](#)

Cancel

Next

IAM

Policies

ec2-terraform

Edit policy

Step 1

Modify permissions in ec2-terraform

Step 2

Review and save

Review and save

Review the permissions, specify details, and tags.

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (1 of 437 services)

Show remaining 436 services

Service	Access level	Resource	Request condition
EC2	Limited: List, Write, Tagging	All resources	None

☒ Set this new version as the default.

Permissions defined in this version will be applied to all the entities this policy is attached to.

Cancel

Previous

Save changes

Policy ec2-terraform updated.

ec2-terraform

Edit

Delete

Access to manage EC2 in enough amount to be used by Terraform.

Policy details

Type	Creation time	Edited time	ARN
Customer managed	December 04, 2024, 22:05 (UTC+02:00)	December 04, 2024, 23:14 (UTC+02:00)	arn:aws:iam::517864397577:policy/ec2-terraform

- Permissions
- Entities attached
- Tags
- Policy versions (2)
- Last Accessed

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (1 of 437 services)

Show remaining 436 services

Service	Access level	Resource	Request condition
EC2	Limited: List, Write, Tagging	All resources	None

п. Повторно застосуємо конфігурацію за допомогою команди «terraform apply»:

aws

Search

[Alt+S]

Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Instances (7)

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

All states

	Name	Instance ID	Instance state	Instance type	Status check
<input type="checkbox"/>	test-lr1	i-01bc42345113f8120	Running	t2.micro	2/2 checks passed
<input type="checkbox"/>	trist-lr2	i-0e0c7aff845f9333	Terminated	t2.micro	-
<input type="checkbox"/>	trist-lr2	i-0c269cf5101f8a692	Terminated	t2.micro	-
<input type="checkbox"/>	trist-lr2	i-0cc78bd75427d3103	Terminated	t2.micro	-
<input type="checkbox"/>		i-0731feec25115c9fe	Terminated	t2.micro	-
<input type="checkbox"/>	trist-lr2	i-0f3aaf72a76b6531f	Terminated	t2.micro	-
<input type="checkbox"/>	trist-lr2	i-023e0c6cb878d6c01	Terminated	t2.micro	-

Search

[Alt+S]

☰

Dashboard

EC2 Global View

Events

▼ Instances

Instances

Instance Types

Launch Templates

Spot Requests

Security Groups (2) Info

Find resources by attribute or tag

<input type="checkbox"/>	Name	Security group ID	Security group name
<input type="checkbox"/>	-	sg-01d6c739b6547ca4d	launch-wizard-1
<input type="checkbox"/>	default-allow-all	sg-0f47b7fdccd167991	default

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ instance_ip = (known after apply)
+ nginx_url = (known after apply)

Do you want to perform these actions?

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ instance_ip = (known after apply)
+ nginx_url = (known after apply)

Do you want to perform these actions?

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_security_group.trist_lab2_sg: Creating...

aws_security_group.trist_lab2_sg: Creation complete after 2s [id=sg-0a2fde36c5e0112bb]

aws_instance.trist_lab2_web: Creating...

aws_instance.trist_lab2_web: Still creating... [10s elapsed]

aws_instance.trist_lab2_web: Creation complete after 13s [id=i-0211471b431705252]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

instance_ip = "52.59.98.181"

nginx_url = "http://52.59.98.181"

- Перевіримо результат розгортання, переконавшись, що ресурси створено коректно:

The screenshot displays two screenshots of the AWS Management Console. The top screenshot shows the 'Instances (8)' page, which lists eight EC2 instances. The instances are organized into a table with columns for Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. The instances are: test-lr1 (Running), trist-lr2 (Running), and five instances named trist-lr2 (Terminated). The bottom screenshot shows the 'Security Groups (3)' page, which lists three security groups: launch-wizard-1, trist-lab2-sg, and default. The table includes columns for Name, Security group ID, Security group name, VPC ID, and Description. The browser's address bar shows the URL 'https://console.aws.amazon.com/ec2/home?region=us-east-1:instances'. The browser's taskbar at the bottom shows several open tabs, including 'lab1', 'lab3', 'Instances | EC2', 'Security groups', 'My Web App', 'terraform-user', 'ec2-terraform', 'Installing or update', and 'Install | Terraform'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
test-lr1	i-01bc42345113f8120	Running	t2.micro	2/2 checks passed	View alarms +
trist-lr2	i-0211471b431705252	Running	t2.micro	Initializing	View alarms +
trist-lr2	i-0e0c7aff845f9333	Terminated	t2.micro	-	View alarms +
trist-lr2	i-0c269cf5101f8a692	Terminated	t2.micro	-	View alarms +
trist-lr2	i-0cc78bd75427d3103	Terminated	t2.micro	-	View alarms +
	i-0731feec25115c9fe	Terminated	t2.micro	-	View alarms +
trist-lr2	i-0f3aaf72a76b6531f	Terminated	t2.micro	-	View alarms +
trist-lr2	i-023e0c6cb878d6c01	Terminated	t2.micro	-	View alarms +

Name	Security group ID	Security group name	VPC ID	Description
-	sg-01d6c739b6547ca4d	launch-wizard-1	ypc-05b5b4786229a5f4e	launch-wizard-1 created 2024-12-
trist-lr2	sg-0a2fde36c5e0112bb	trist-lab2-sg	ypc-05b5b4786229a5f4e	Allow SSH, HTTPS, and HTTP traff
default-allow-all	sg-0f47b7fdcd167991	default	ypc-05b5b4786229a5f4e	default VPC security group

р. Видалимо створені ресурси за допомогою команди «terraform destroy»:

Plan: 0 to add, 0 to change, 2 to destroy.

Changes to Outputs:

- instance_ip = "52.59.98.181" -> null
- nginx_url = "http://52.59.98.181" -> null

Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

```
aws_instance.trist_lab2_web: Destroying... [id=i-0211471b431705252]
aws_instance.trist_lab2_web: Still destroying... [id=i-0211471b431705252, 10s elapsed]
aws_instance.trist_lab2_web: Still destroying... [id=i-0211471b431705252, 20s elapsed]
aws_instance.trist_lab2_web: Still destroying... [id=i-0211471b431705252, 30s elapsed]
aws_instance.trist_lab2_web: Still destroying... [id=i-0211471b431705252, 40s elapsed]
aws_instance.trist_lab2_web: Still destroying... [id=i-0211471b431705252, 50s elapsed]
aws_instance.trist_lab2_web: Destruction complete after 51s
aws_security_group.trist_lab2_sg: Destroying... [id=sg-0a2fde36c5e0112bb]
aws_security_group.trist_lab2_sg: Destruction complete after 0s
```

Destroy complete! Resources: 2 destroyed.

д. Перевіримо результат знищення, щоб переконатися, що всі ресурси успішно видалено:



Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

Instances (8) Info

Last updated less than a minute ago

Connect

Instance state

Find Instance by attribute or tag (case-sensitive)

All states

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Z
<input type="checkbox"/>	test-lr1	i-01bc42345113f8120	Running	t2.micro	2/2 checks passed	View alarms	eu-central-1b
<input type="checkbox"/>	trist-lr2	i-0e0c7aff845f9333	Terminated	t2.micro	-	View alarms	eu-central-1b
<input type="checkbox"/>	trist-lr2	i-0c269cf5101f8a692	Terminated	t2.micro	-	View alarms	eu-central-1b
<input type="checkbox"/>	trist-lr2	i-0cc78bd75427d3103	Terminated	t2.micro	-	View alarms	eu-central-1b
<input type="checkbox"/>		i-0731feec25115c9fe	Terminated	t2.micro	-	View alarms	eu-central-1b
<input type="checkbox"/>	trist-lr2	i-0211471b431705252	Terminated	t2.micro	-	View alarms	eu-central-1b
<input type="checkbox"/>	trist-lr2	i-0f3aaf72a76b6531f	Terminated	t2.micro	-	View alarms	eu-central-1b
<input type="checkbox"/>	trist-lr2	i-023e0c6cb878d6c01	Terminated	t2.micro	-	View alarms	eu-central-1b



Dashboard

EC2 Global View

Events

▼ Instances

Instances

Instance Types

Launch Templates

Spot Requests

Security Groups (2) Info

⌕ Find resources by attribute or tag

⌵

Name

⌵

Security group ID

⌵

Security group name

⌵

VPC ID

⌵

Description

-

sg-01d6c739b6547ca4d

launch-wizard-1

vpc-05b5b4786229a5f4e

launch-wizard-1 created 2024-12-

default-allow-all

sg-0f47b7fdcd167991

default

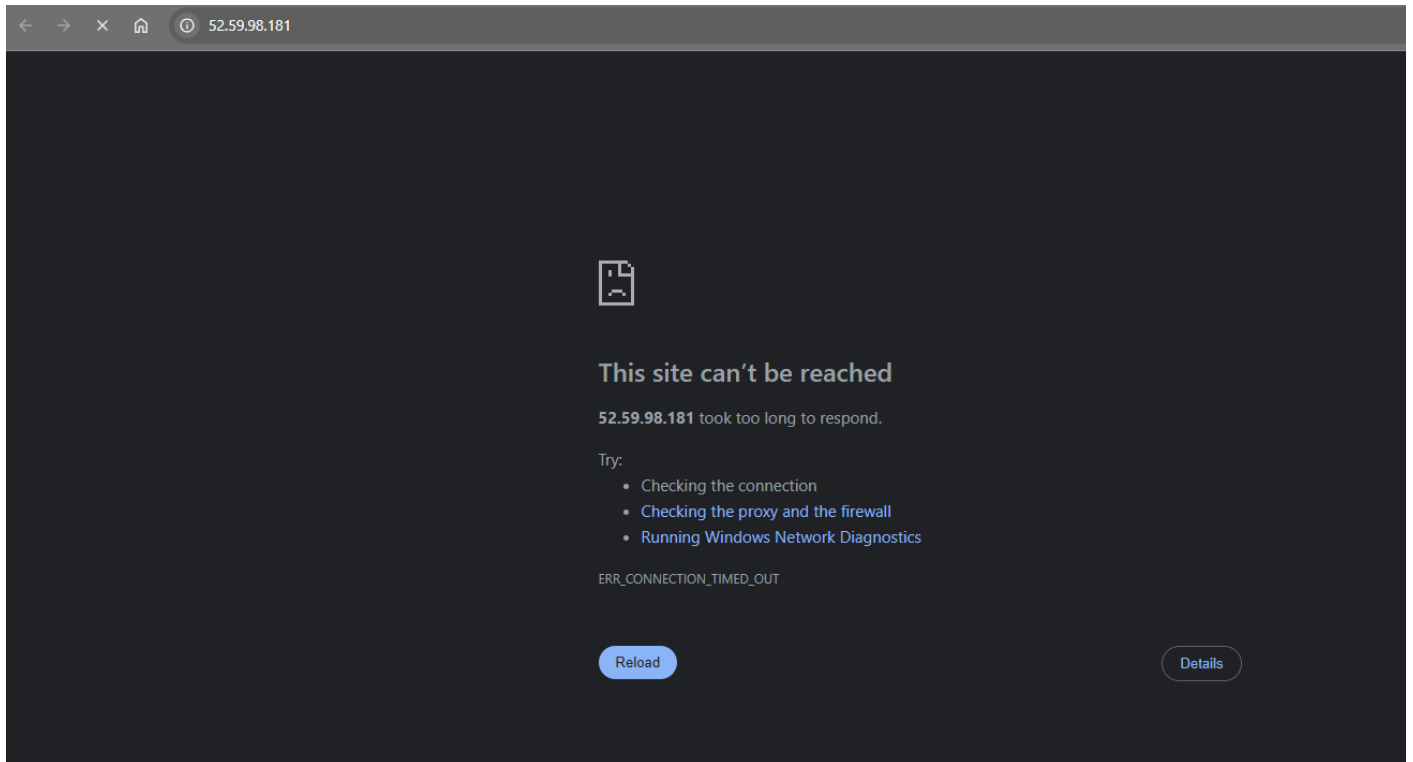
vpc-05b5b4786229a5f4e

default VPC security group

⌵

1

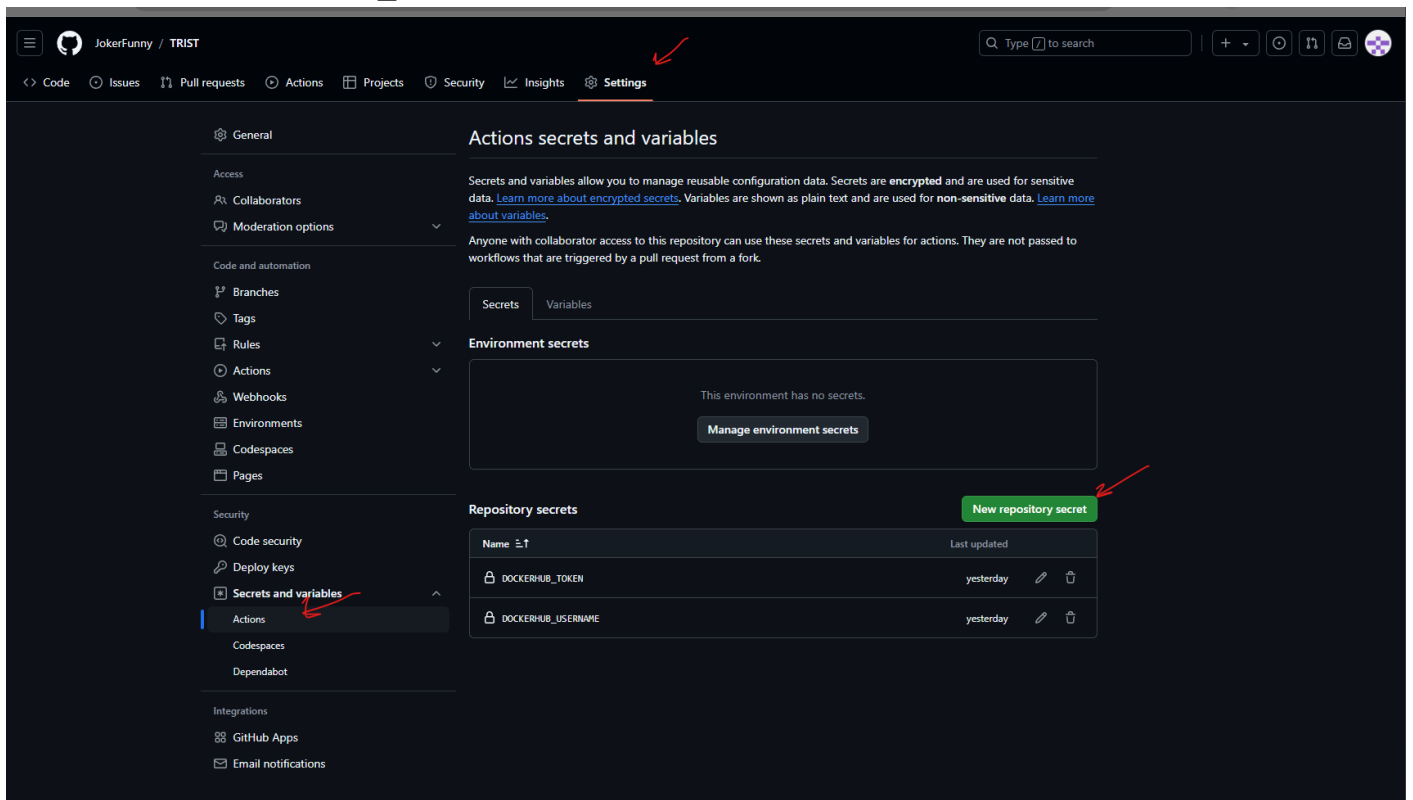
⌵



4. (Додатково) Створити GitHub Actions pipeline, який буде виконувати terraform init, plan, apply замість ручного виконання. Передбачити коректне виконання джобів у різних станах інфраструктури:

а) Додамо токен із другого кроку в GitHub Secrets для подальшого використання у GitHub Actions:

- i. AWS_ACCESS_KEY_ID;
- ii. AWS_SECRET_ACCESS_KEY;
- iii. AWS_REGION.



Actions secrets / New secret

Name *

AWS_ACCESS_KEY_ID

Secret *

AKIAXREY7TMEXLPTFZPZ



Add secret

Actions secrets / New secret

Name *

AWS_SECRET_ACCESS_KEY

Secret *

SECRET



Add secret

Actions secrets / New secret

Name *

AWS_REGION

Secret *

eu-central-1



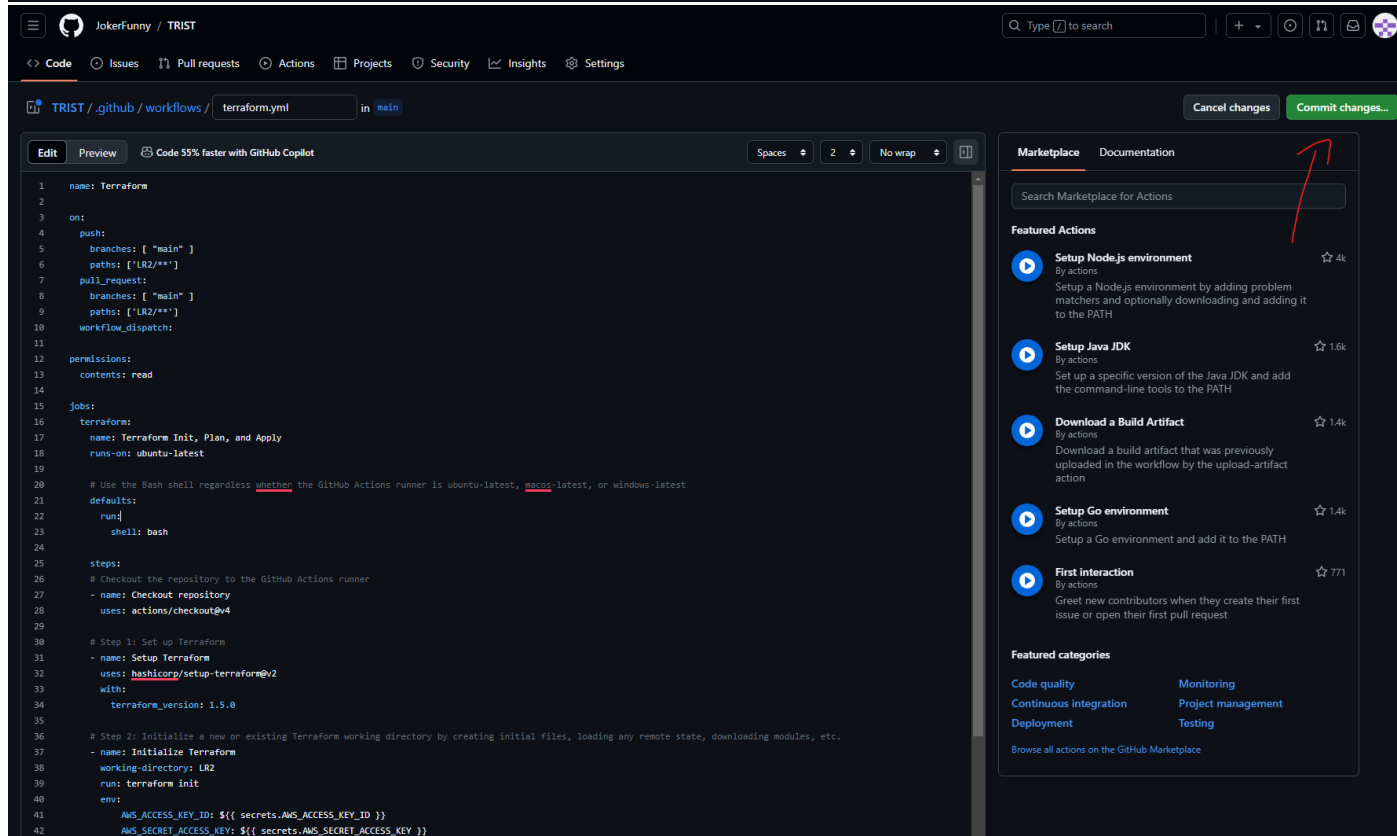
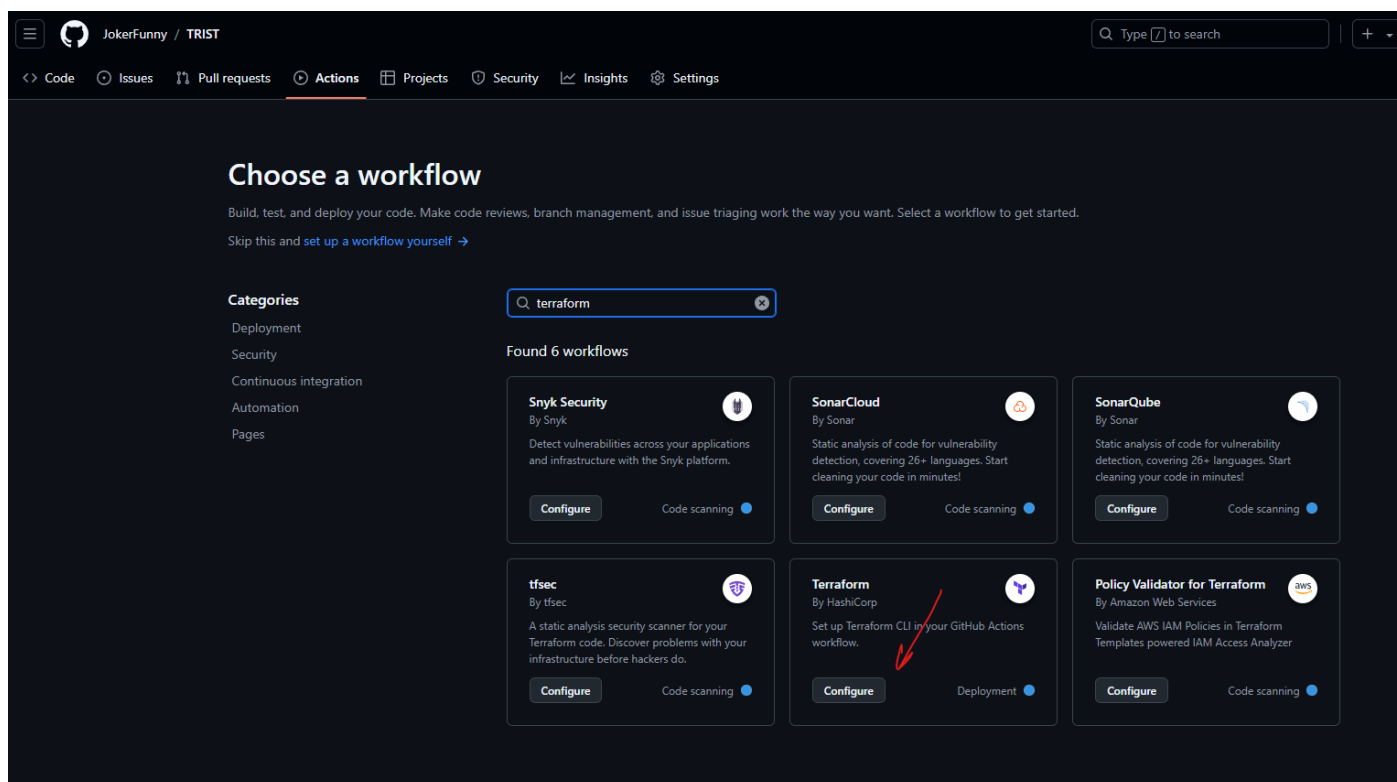
Add secret

Repository secrets

New repository secret

Name	Last updated		
AWS_ACCESS_KEY_ID	52 minutes ago		
AWS_REGION	now		
AWS_SECRET_ACCESS_KEY	51 minutes ago		
DOCKERHUB_TOKEN	yesterday		
DOCKERHUB_USERNAME	yesterday		

- б) Створимо GitHub Actions для автоматизації процесу деплою за допомогою Terraform. Workflow має включати такі етапи:
- Ініціалізація Terraform (terraform init);
 - Перегляд плану (terraform plan);
 - Застосування конфігурації (terraform apply);
 - Вивід інформації про застосунок (адресу).



terraform.yml:

```
name: Terraform

on:
  push:
    branches: [ "main" ]
    paths: ['LR2/**']
  pull_request:
    branches: [ "main" ]
    paths: ['LR2/**']
  workflow_dispatch:

jobs:
  terraform:
    name: Terraform Init, Plan, and Apply
    runs-on: ubuntu-latest

    # Use the Bash shell regardless whether the GitHub Actions runner is ubuntu-latest, macos-latest, or windows-latest
    defaults:
      run:
        shell: bash

    steps:
      # Checkout the repository to the GitHub Actions runner
      - name: Checkout repository
        uses: actions/checkout@v4

      # Step 1: Set up Terraform
      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2
        with:
          terraform_version: 1.5.0

      # Step 2: Initialize a new or existing Terraform working directory by creating initial files, loading any remote state, downloading modules, etc.
      - name: Initialize Terraform
        working-directory: LR2
        run: terraform init

      env:
        AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
        AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
```



```

permissions:
  contents: read

jobs:
  terraform:
    name: Terraform Init, Plan, and Apply
    runs-on: ubuntu-latest

    # Use the Bash shell regardless whether the GitHub Actions runner is ubuntu-latest, macos-latest, or
    windows-latest
    defaults:
      run:
        shell: bash

    steps:
      # Checkout the repository to the GitHub Actions runner
      - name: Checkout repository
        uses: actions/checkout@v4

      # Step 1: Set up Terraform
      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2
        with:
          terraform_version: 1.5.0

      # Step 2: Initialize a new or existing Terraform working directory by creating initial files, loading
      any remote state, downloading modules, etc.
      - name: Initialize Terraform
        working-directory: LR2
        run: terraform init
        env:
          AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
          AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
          AWS_REGION: ${ secrets.AWS_REGION }

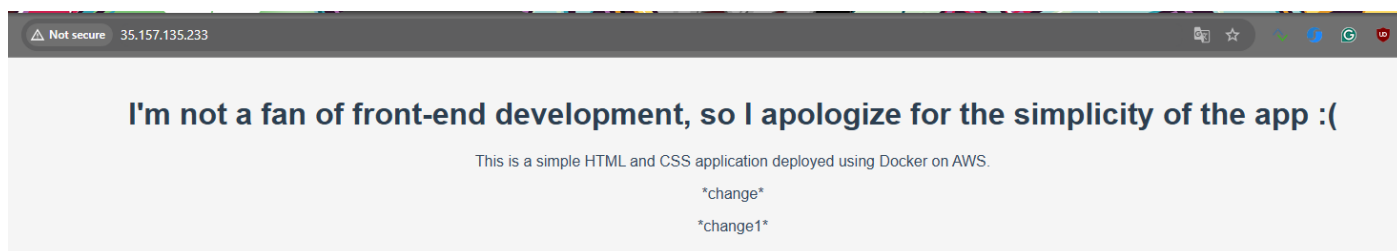
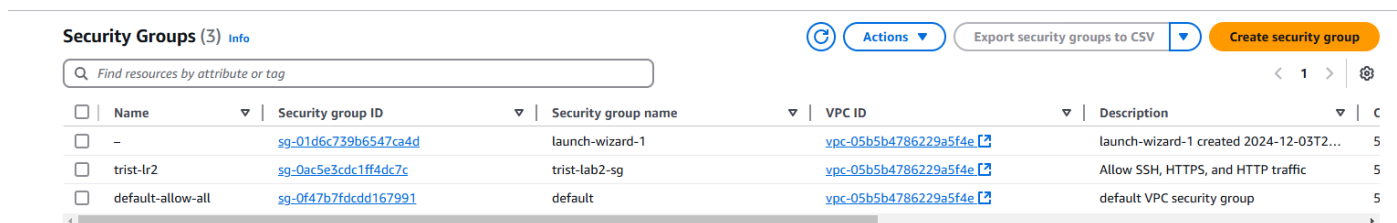
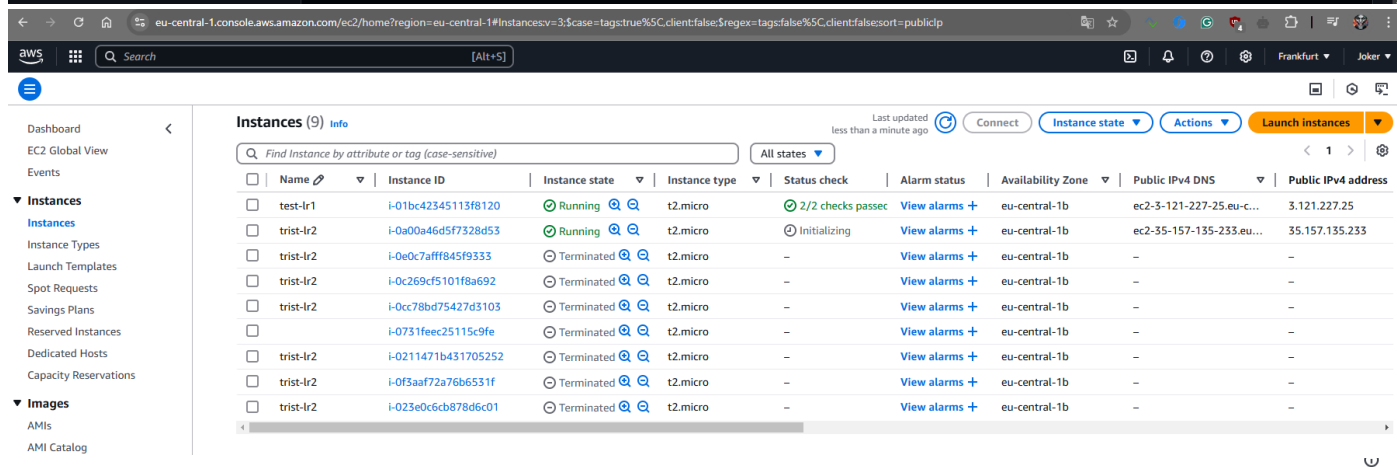
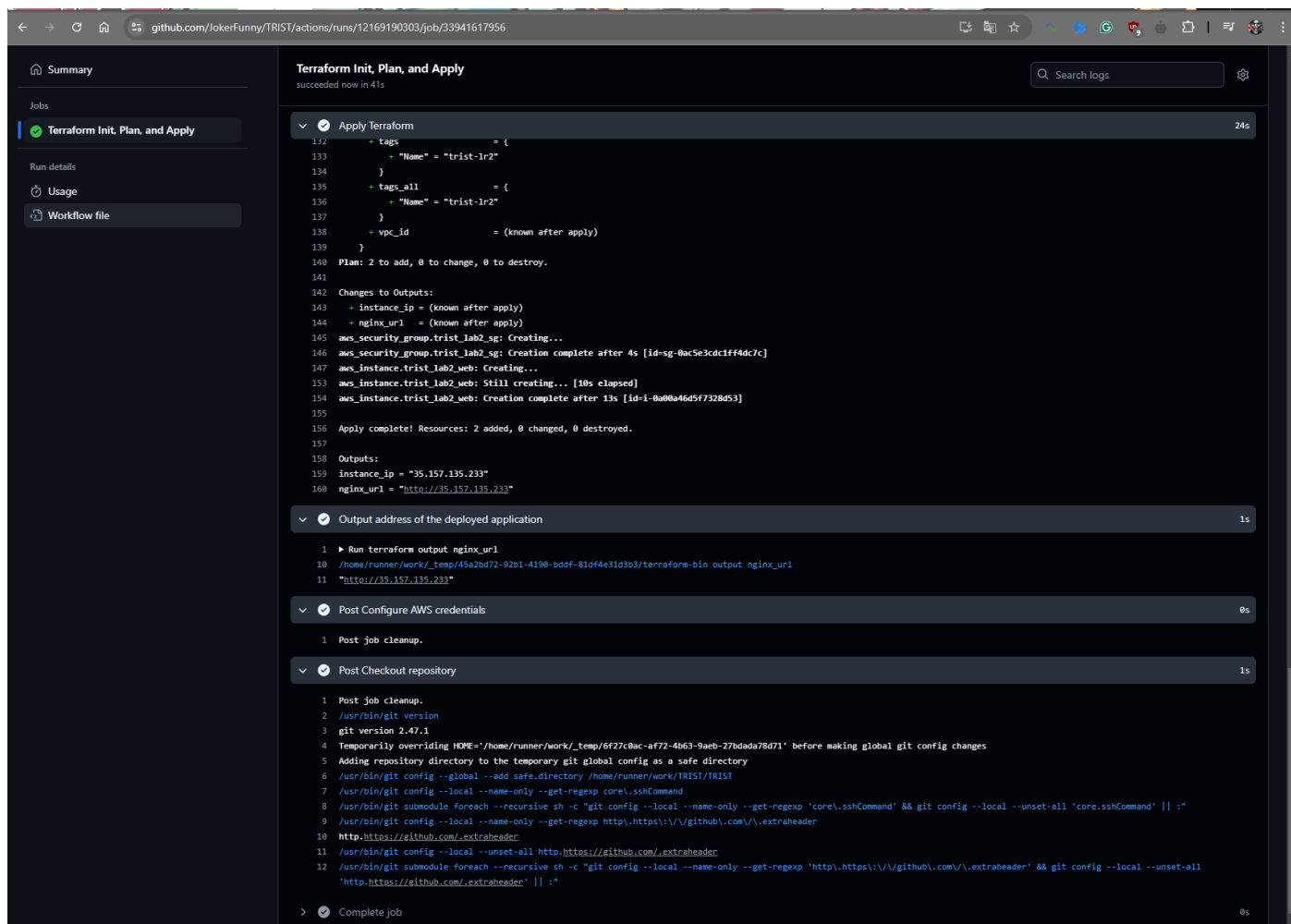
      # Step 3: Generates an execution plan for Terraform
      - name: Plan Terraform
        working-directory: LR2
        run: terraform plan

      # Step 4: Apply Terraform
      - name: Apply Terraform
        working-directory: LR2
        run: terraform apply -auto-approve

      # Step 5: Output Instance IP
      - name: Output address of the deployed application
        working-directory: LR2
        run: terraform output nginx_url

```

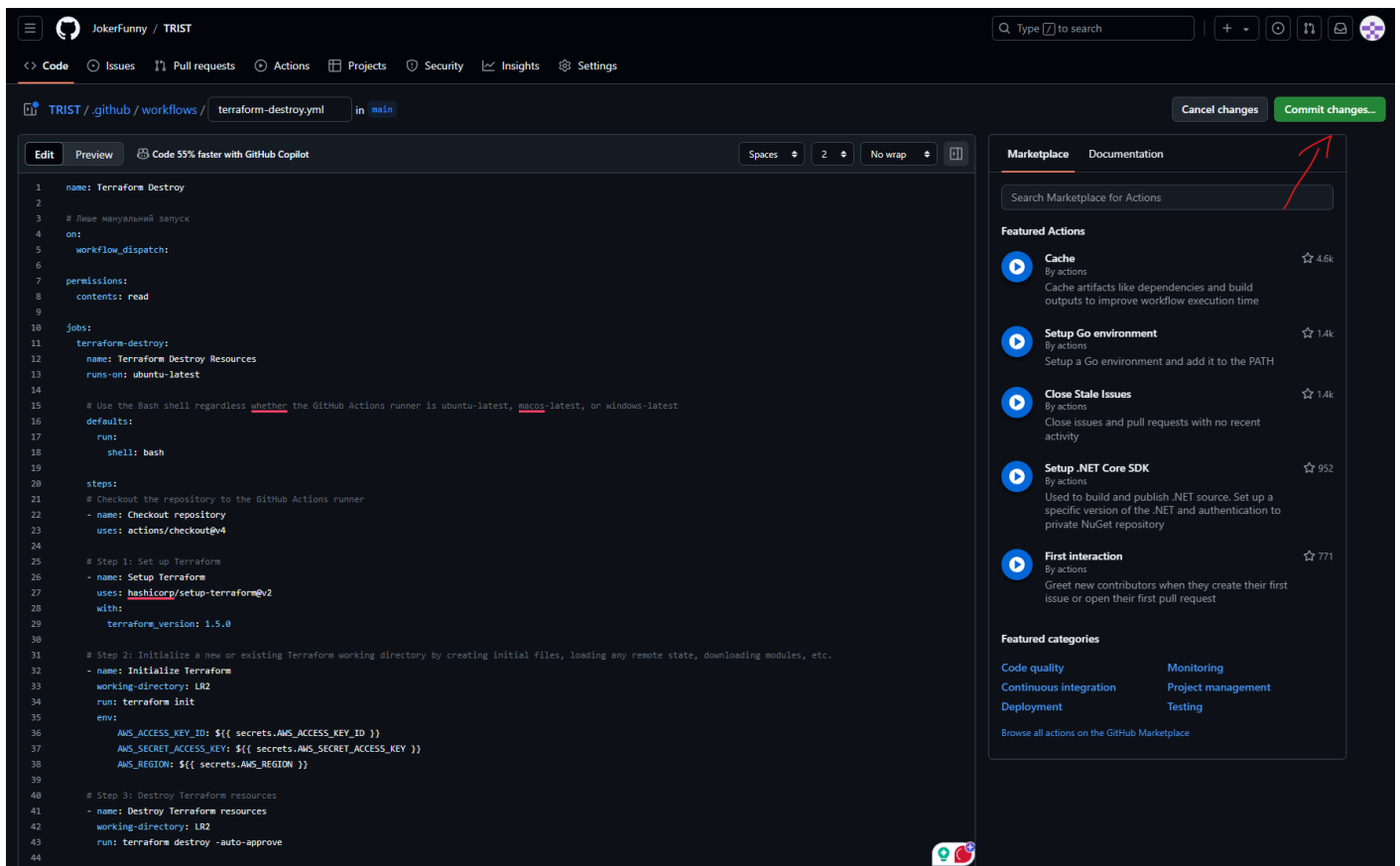
в) Перевіримо виконання пайплайну, переконавшись, що ресурси створені успішно:



г) Створимо окремий GitHub Actions workflow для автоматизації процесу знищення інфраструктури за допомогою Terraform. Workflow

налаштовується на manual trigger (ручний запуск). Має включати наступні етапи:

- i. Ініціалізацію Terraform (terraform init);
- ii. Знищення інфраструктури (terraform destroy).



terraform-destroy.yml:

```
name: Terraform Destroy

# Лише мануальний запуск
on:
  workflow_dispatch:

permissions:
  contents: read

jobs:
  terraform-destroy:
    name: Terraform Destroy Resources
    runs-on: ubuntu-latest

    # Use the Bash shell regardless whether the GitHub Actions runner is ubuntu-latest, macos-latest, or windows-latest
    defaults:
      run:
        shell: bash

    steps:
      # Checkout the repository to the GitHub Actions runner
      - name: Checkout repository
        uses: actions/checkout@v4
```

Step 1: Set up Terraform

```
- name: Setup Terraform
  uses: hashicorp/setup-terraform@v2
  with:
    terraform_version: 1.5.0
```

Step 2: Initialize a new or existing Terraform working directory by creating initial files, loading any remote state, downloading modules, etc.

```
- name: Initialize Terraform
  working-directory: LR2
  run: terraform init
  env:
    AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
    AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
    AWS_REGION: ${ secrets.AWS_REGION }
```

Step 3: Destroy Terraform resources

```
- name: Destroy Terraform resources
  working-directory: LR2
  run: terraform destroy -auto-approve
```

д) Перевіримо виконання пайплайну знищення інфраструктури:

The screenshot shows the GitHub Actions interface for a workflow named 'Terraform Destroy Resources'. The left sidebar contains navigation links: Summary, Jobs, Terraform Destroy Resources (selected), Run details, Usage, and Workflow file. The main panel displays the job logs. The job 'Terraform Destroy Resources' is marked as 'succeeded now in 13s'. The logs show the following steps:

- Set up job (1s)
- Checkout repository (2s)
- Setup Terraform (0s)
- Initialize Terraform (6s)
- Destroy Terraform resources (3s)
- Post Checkout repository (0s)
- Complete job (0s)

The 'Initialize Terraform' step shows the execution of 'terraform init' with the following output:

```
1 Run terraform init
2 terraform init
3 shell: /usr/bin/bash --noprofile --norc -e -o pipefail (0)
4 env:
5   TERRAFORM_CLI_PATH: /home/runner/work/_temp/f58f8ee4-a959-4d92-bf5a-66f312d0d605
6   AWS_ACCESS_KEY_ID: ***
7   AWS_SECRET_ACCESS_KEY: ***
8   AWS_REGION: ***
9
10 /home/runner/work/_temp/f58f8ee4-a959-4d92-bf5a-66f312d0d605/terraform-bin init
11 Initializing the backend...
12 Initializing provider plugins...
13 - Finding latest version of hashicorp/aws...
14 - Installing hashicorp/aws v5.80.0...
15 - Installed hashicorp/aws v5.80.0 (signed by HashiCorp)
16 Terraform has created a lock file .terraform.lock.hcl to record the provider
17 selections it made above. Include this file in your version control repository
18 so that Terraform can guarantee to make the same selections by default when
19 you run "terraform init" in the future.
20 Terraform has been successfully initialized!
21
22 You may now begin working with Terraform. Try running "terraform plan" to see
23 any changes that are required for your infrastructure. All Terraform commands
24 should now work.
25 If you ever set or change modules or backend configuration for Terraform,
26 rerun this command to reinitialize your working directory. If you forget, other
27 commands will detect it and remind you to do so if necessary.
```

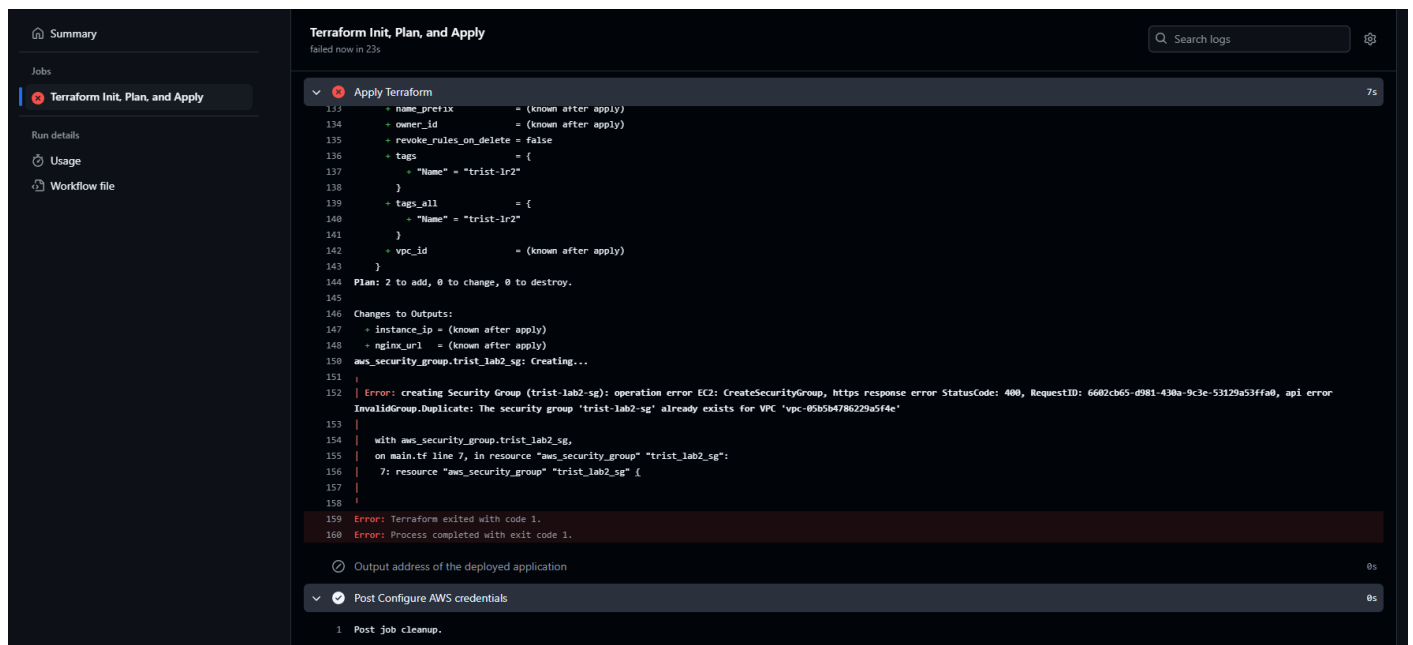
The 'Destroy Terraform resources' step shows the execution of 'terraform destroy -auto-approve' with the following output:

```
1 Run terraform destroy -auto-approve
2 /home/runner/work/_temp/f58f8ee4-a959-4d92-bf5a-66f312d0d605/terraform-bin destroy -auto-approve
3 No changes. No objects need to be destroyed.
4 Either you have not created any objects yet or the existing objects were
5 already deleted outside of Terraform.
6
7 Destroy complete! Resources: 0 destroyed.
8 ::debug::Terraform exited with code 0.
```

The message 'No changes. No objects need to be destroyed.' is highlighted with a red circle in the original image.

Нажаль, при виконанні Workflow для знищення інфраструктури (наприклад, terraform-destroy.yml), ми отримаємо повідомлення про те, що немає інфраструктури для знищення. Це відбувається через те, що Workflow не бачить інфраструктури, яка була створена раніше в іншому Workflow (наприклад, terraform.yml).

Так само, повторне виконання Workflow terraform.yml для створення інфраструктури призведе до помилки – Terraform спробує повторно створити вже існуючі ресурси:



```
Terraform Init, Plan, and Apply
failed now in 23s

Apply Terraform
132 + name_prefix = (known after apply)
133 + owner_id    = (known after apply)
134 + revoke_rules_on_delete = false
135 + tags        = {
136   + "Name" = "trist-lr2"
137 }
138 + tags_all    = {
139   + "Name" = "trist-lr2"
140 }
141 + vpc_id       = (known after apply)
142 }
143
144 Plan: 2 to add, 0 to change, 0 to destroy.
145
146 Changes to Outputs:
147 + instance_ip = (known after apply)
148 + nginx_url   = (known after apply)
149
150 aws_security_group.trist_lab2_sg: Creating...
151
152 Error: creating Security Group (trist-lab2-sg): operation error EC2: CreateSecurityGroup, https response error StatusCode: 400, RequestID: 6602cb65-d981-430a-9c3e-53129a53ffa0, api error InvalidGroup.Duplicate: The security group 'trist-lab2-sg' already exists for VPC 'vpc-05b5b4786229a5f4e'
153
154 with aws_security_group.trist_lab2_sg,
155 on main.tf line 7, in resource "aws_security_group" "trist_lab2_sg":
156   7: resource "aws_security_group" "trist_lab2_sg" {
157
158   }
159
160 Error: Terraform exited with code 1.
161 Error: Process completed with exit code 1.

Output address of the deployed application
0s

Post Configure AWS credentials
0s

1 Post job cleanup.
```

Це відбувається через те, що Terraform state (файл terraform.tfstate) був створений під час виконання першого пайплайну, але не збережений після його завершення. Це унеможливорює коректну взаємодію Terraform із вже створеними ресурсами (Terraform залежить від state-файлу для відстеження існуючої інфраструктури). Без цього файлу Terraform вважає, що інфраструктури немає, і не може виконати terraform destroy/повторно застосувати terraform apply (який без наявних змін в існуючій інфраструктурі має нічого не робити).

Для коректного виконання всіх workflow, які взаємодіють із інфраструктурою, необхідно зберігати та синхронізувати файл terraform.tfstate. Це може бути виконано багатьма варіантами, наприклад використовуючи інфраструктуру AWS:

- Використовувати S3 bucket для зберігання Terraform state;
- Використовувати DynamoDB для блокування стану, щоб уникнути конфліктів при одночасному доступі;
- Додати налаштування backend у Terraform;
- Включити синхронізацію state-файлу у всіх Terraform workflow.

Висновки: в результаті виконання цієї лабораторної роботи було ознайомлено з базовими концепціями автоматизації розгортання інфраструктури за допомогою Terraform і GitHub Actions.

На основі отриманих знань було реалізовано практичну частину, яка полягала у підготовці інфраструктури з минулої лабораторної роботи засобами Terraform та створенні пайплайнів для автоматизації процесів створення та знищення інфраструктури в AWS EC2.

Було розглянуто важливість використання віддаленого Terraform state для синхронізації між Workflow та забезпечення узгодженості інфраструктури.

Вихідний код та конфігурації можна знайти за наступним посиланням на GitHub:

- [main.tf](#);
- [Github actions](#).