Raport Projektu Chmury 1

Mateusz Staszków, #263949, IST PSI

1. Wprowadzenie

Celem projektu było stworzenie aplikacji webowej (np. prostego chatu grupowego) z podziałem na moduły backend oraz frontend, które są hostowane niezależnie. Realizacja projektu obejmowała:

- Implementację co najmniej 2 endpointów GET oraz 2 endpointów POST (w tym jeden do przesyłania plików multimedialnych).
- Dockerizację obu modułów oraz konfigurację Dockera.
- Synchronizację aplikacji z usługami AWS, w tym Elastic Beanstalk, RDS, S3, CloudWatch, Cognito oraz Lambda.
- Automatyzację wdrożenia infrastruktury przy użyciu Terraform.

2. Opis aplikacji

Aplikacja została pierwotnie przetestowana lokalnie (na localhost) z wykorzystaniem kontenerów Docker, a następnie zsynchronizowana z AWS. Główne funkcjonalności aplikacji to:

- Rejestracja i uwierzytelnianie użytkowników z wykorzystaniem AWS Cognito (wraz z automatycznym potwierdzaniem kont przy użyciu funkcji Lambda).
- Możliwość wysyłania wiadomości między użytkownikami oraz przesyłania plików multimedialnych.
- Prosty interfejs umożliwiający pobieranie przesłanych plików oraz podgląd wiadomości zarówno wysłanych, jak i odebranych

Kod całego projektu znajduje się pod linkiem: https://github.com/stvshy/chat-app-aws/tree/using-terraform

Repozytorium zostało podzielone na dwa branche: master (konfiguracja manualna) oraz using-terraform (konfiguracja z terraformem)

3. Implementacja i konfiguracja

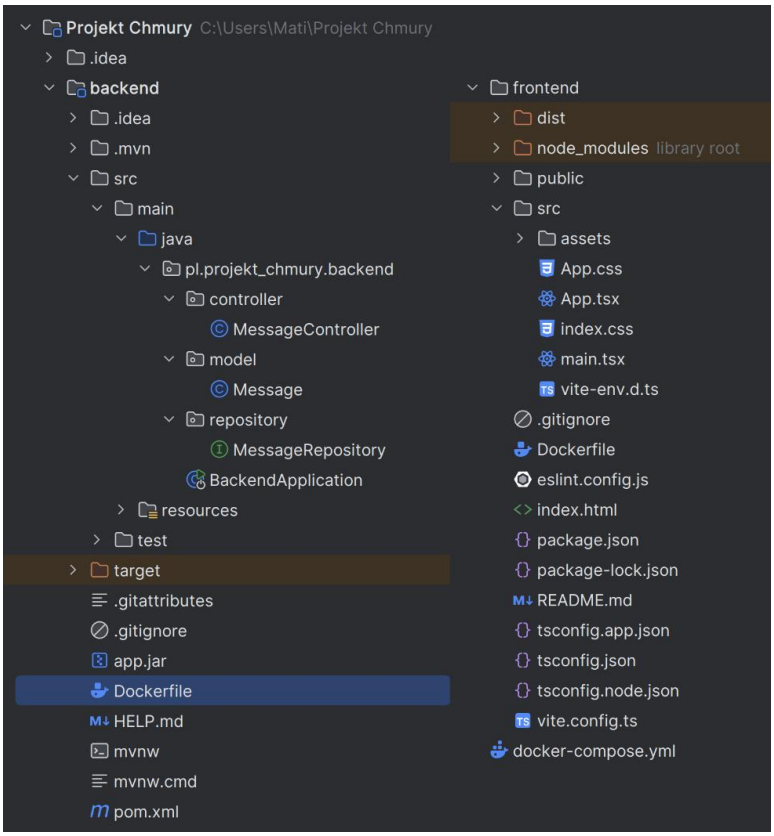
3.1 Dockerizacja

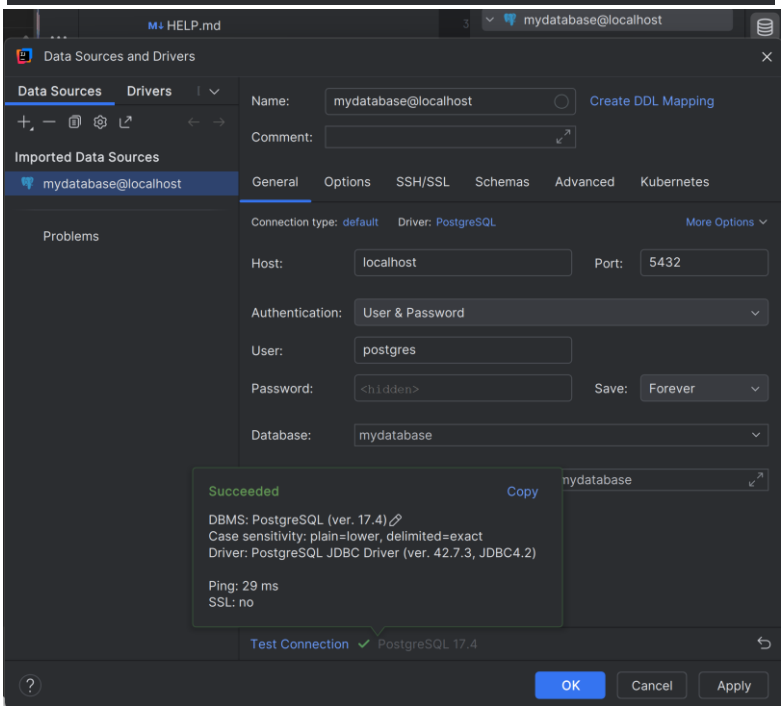
- Wszystkie moduły (frontend i backend) zostały zdockerowane, co umożliwiło uruchomienie aplikacji lokalnie przed wdrożeniem na AWS.

```
PS C:\Users\Mati\Projekt Chmury> docker-compose up -d
time="2025-03-18T11:56:38+01:00" level=warning msg="C:\\Users\\Mati\\Projekt Chmury
\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, plea
se remove it to avoid potential confusion"
[+] Running 3/3
 ✓ Container postgres_container  Started                                    1.5s
 ✓ Container backend-container   Started                                    1.8s
 ✓ Container frontend-container  Started                                    0.8s
```

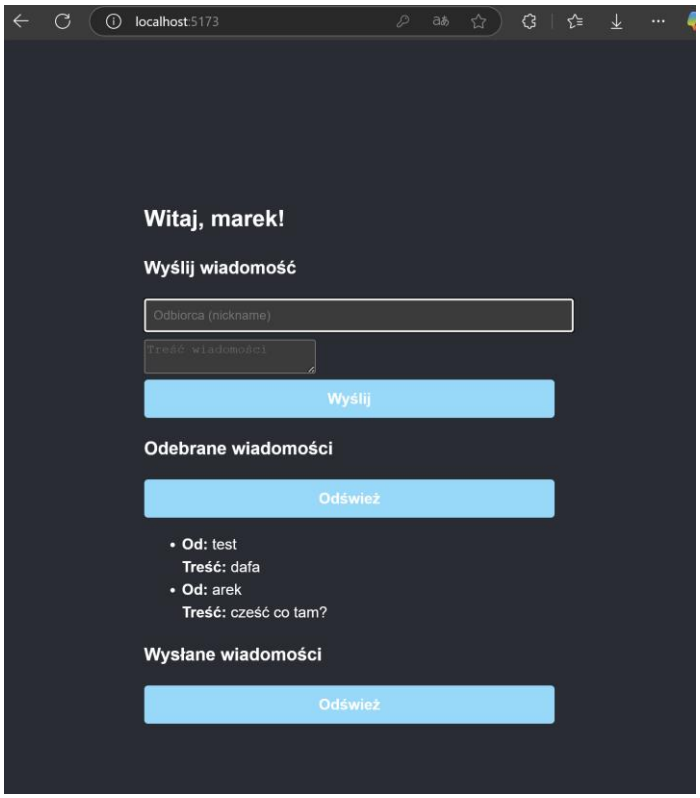
```
C:\Users\Mati>docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS
 NAMES
6ad59f74eb92   projektchmury-backend  "java -jar app.jar"      25 seconds ago   Up 24 seconds   0.0.0.0:8081->8080/tcp
 backend-container
4e05b1c359cb   postgres:latest        "docker-entrypoint.s…"   25 seconds ago   Up 25 seconds   0.0.0.0:5432->5432/tcp
 postgres_container
4fde22e07aa0   projektchmury-frontend "docker-entrypoint.s…"   58 seconds ago   Up 25 seconds   0.0.0.0:5173->5173/tcp
 frontend-container
```

- Po potwierdzeniu poprawności działania na localhost, przystąpiono do synchronizacji z chmurą. Tak na tamten moment wyglądała struktura projektu:

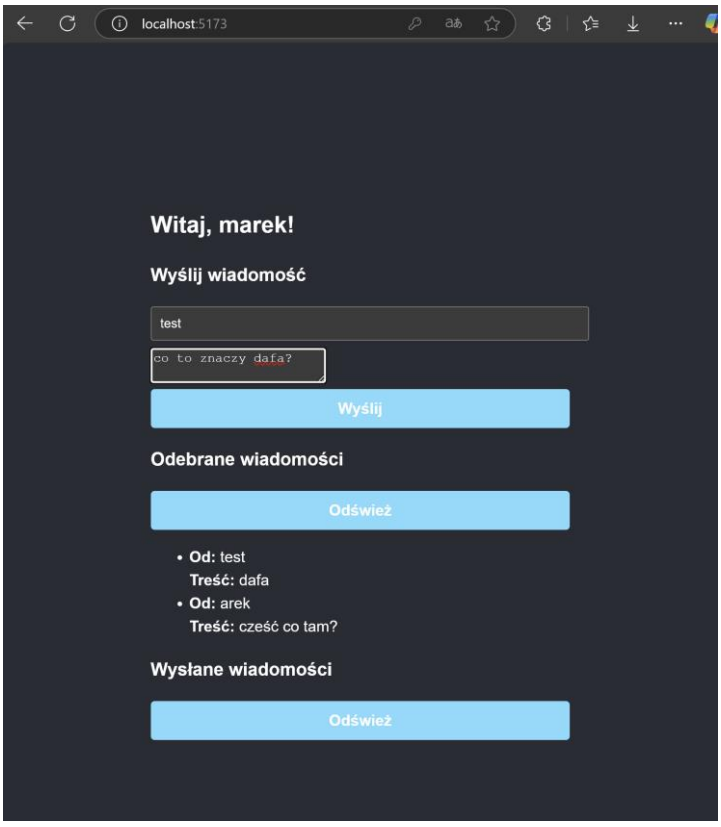


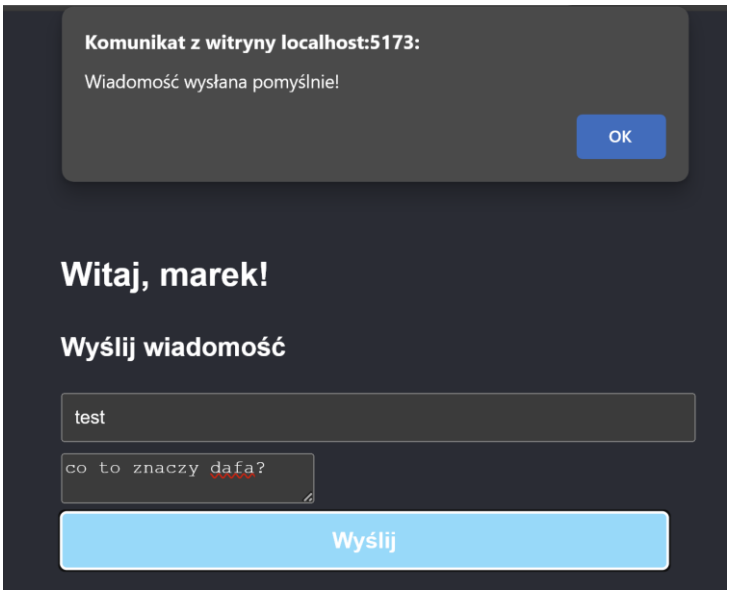


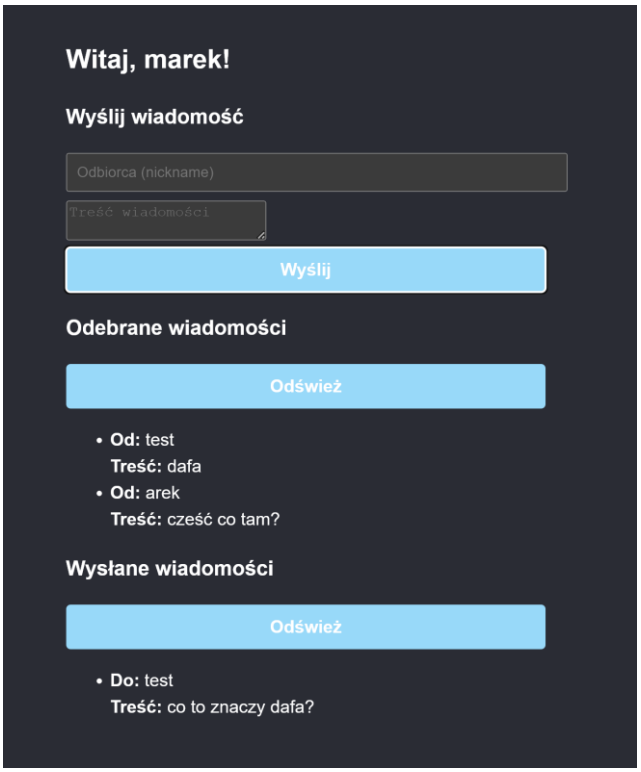
- Tak natomiast wyglądało na tamten moment UI:

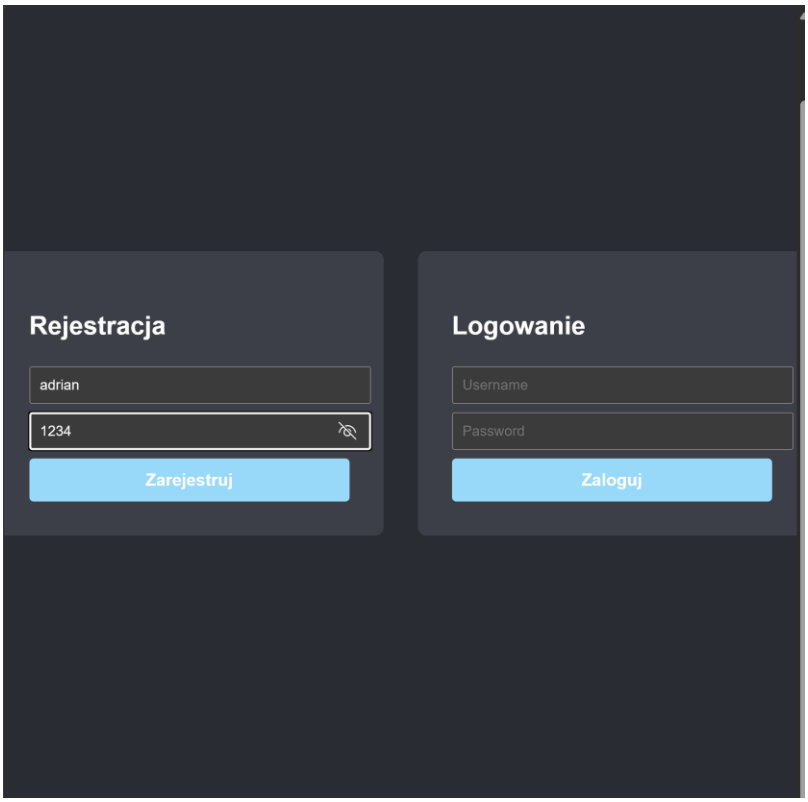


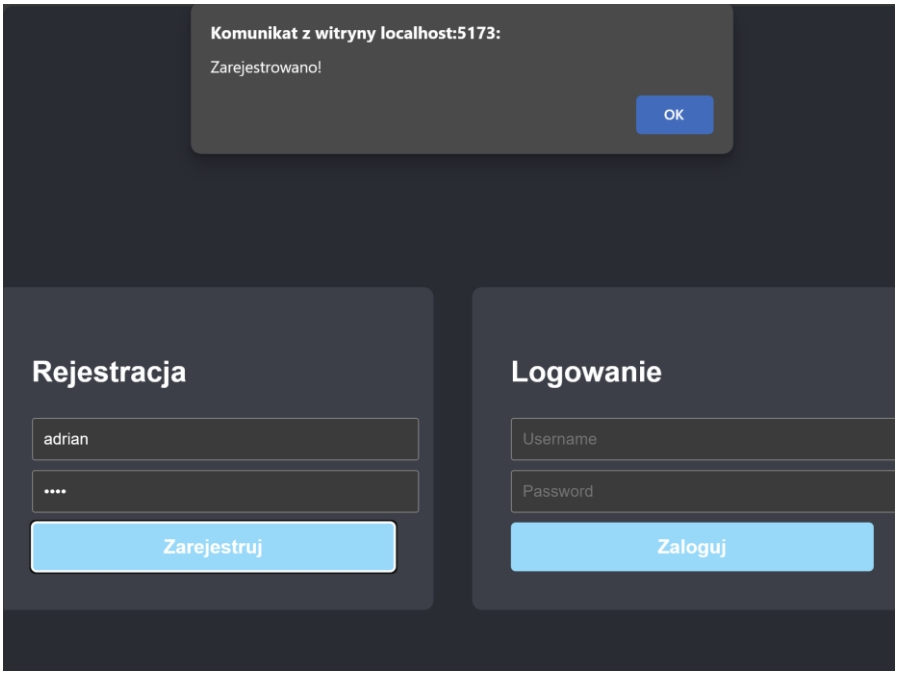
- Poprawne działanie na localhoście:




Komunikat z witryny localhost:5173:
Wiadomość wysłana pomyślnie!
OK
Witaj, marek!
Wyślij wiadomość
test
co to znaczy dafa?
Wyślij


Witaj, marek!
Wyślij wiadomość
Odbiorca (nickname)
Treść wiadomości
Wyślij
Odebrane wiadomości
Odśwież
Od: test
Treść: dafa
Od: arek
Treść: cześć co tam?
Wysłane wiadomości
Odśwież
Do: test
Treść: co to znaczy dafa?


Rejestracja
adrian
1234
Zarejestruj
Logowanie
Username
Password
Zaloguj


Komunikat z witryny localhost:5173:
Zarejestrowano!
OK
Rejestracja
adrian
••••
Zarejestruj
Logowanie
Username
Password
Zaloguj

users | application.properties | Ba

Tx: Auto | DDL

WHERE | ORDER BY

	id	password	username
1	1	$2a$10$FhLljduSao…	marek
2	2	$2a$10$ILDG2epHAp…	test
3	52	$2a$10$1My.L1DCYG…	arek
4	53	$2a$10$agoIRgVO0F…	adrian

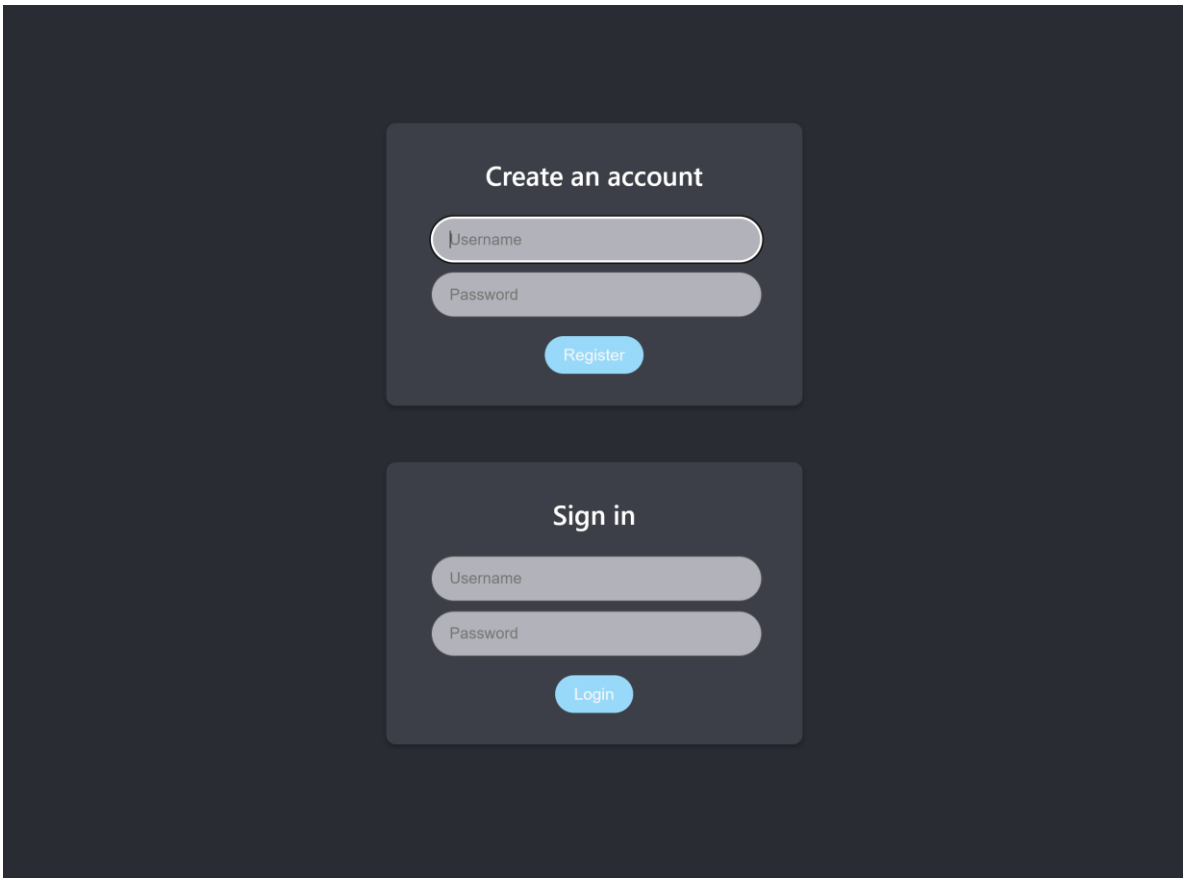
message | package.json | .dockerignore | pom.xml (backend)

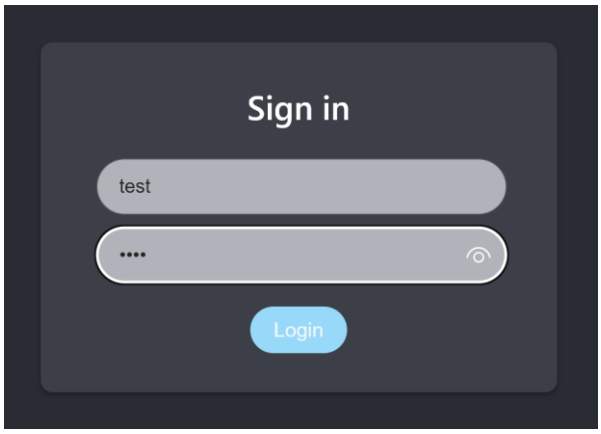
Tx: Auto | DDL

WHERE

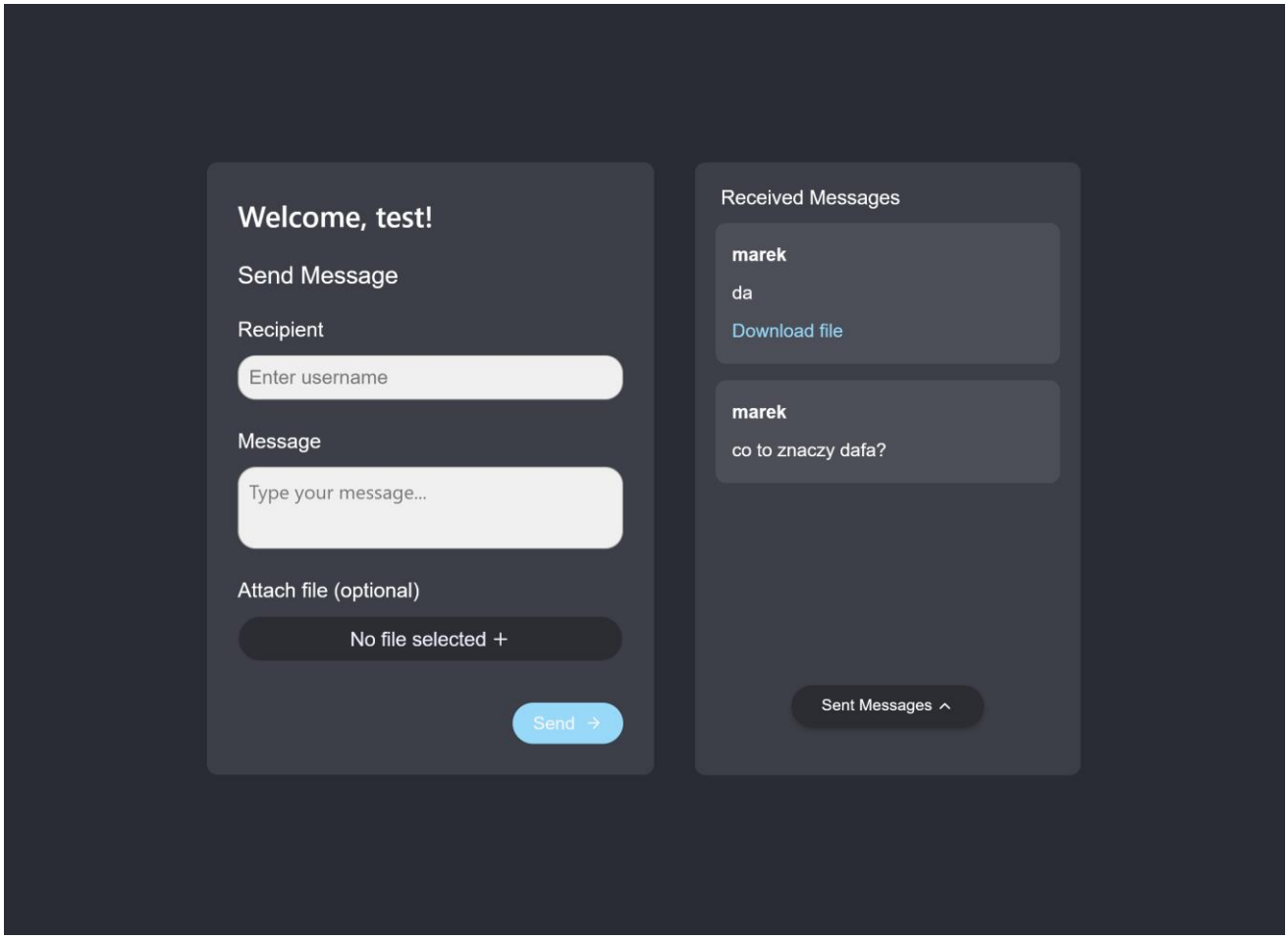
	id	content	author_id	recipient_id
1	1	dafa	2	1
2	2	cześć co tam?	52	1
3	3	co to znaczy dafa?	1	2

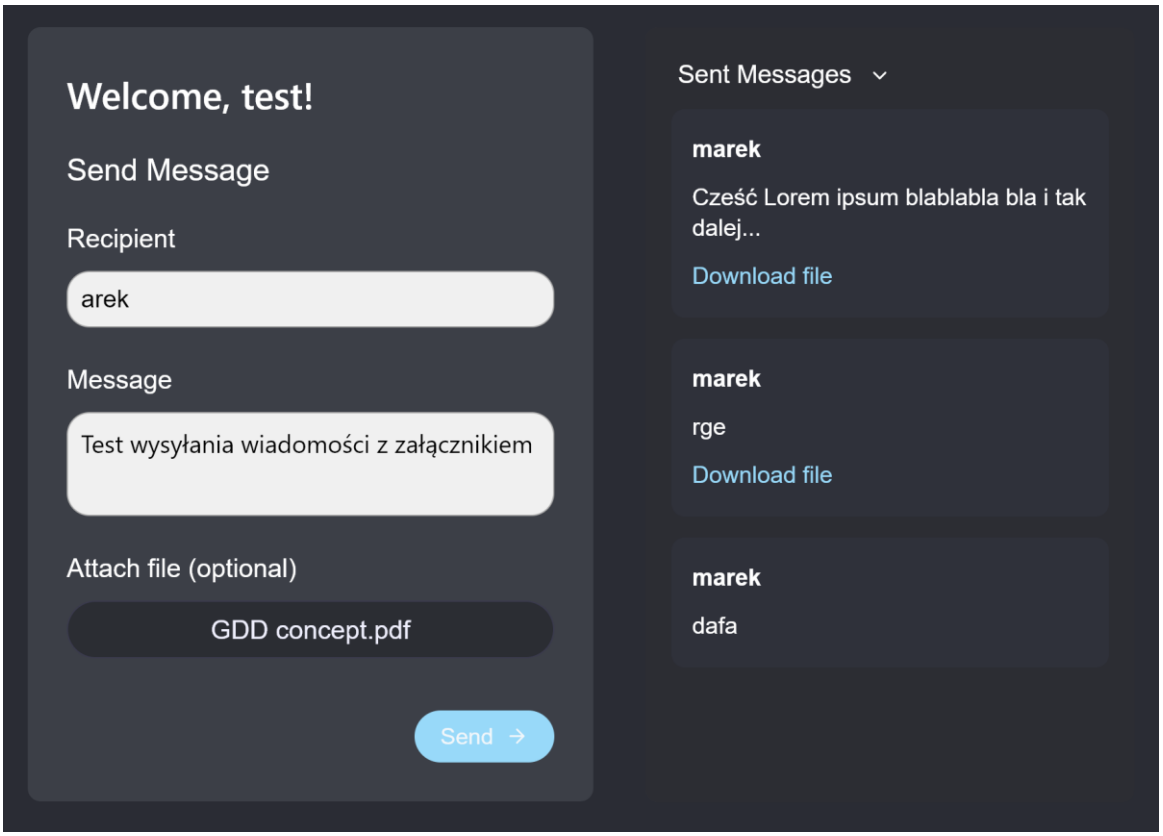
- Nowe UI logowania:

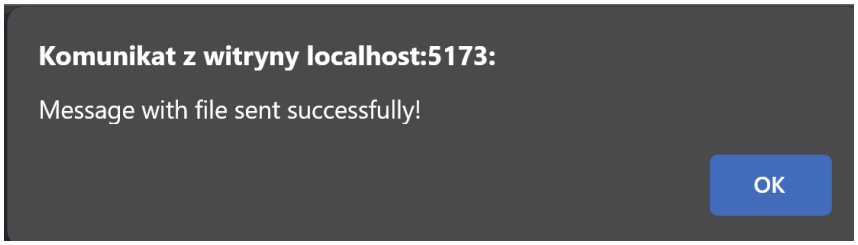


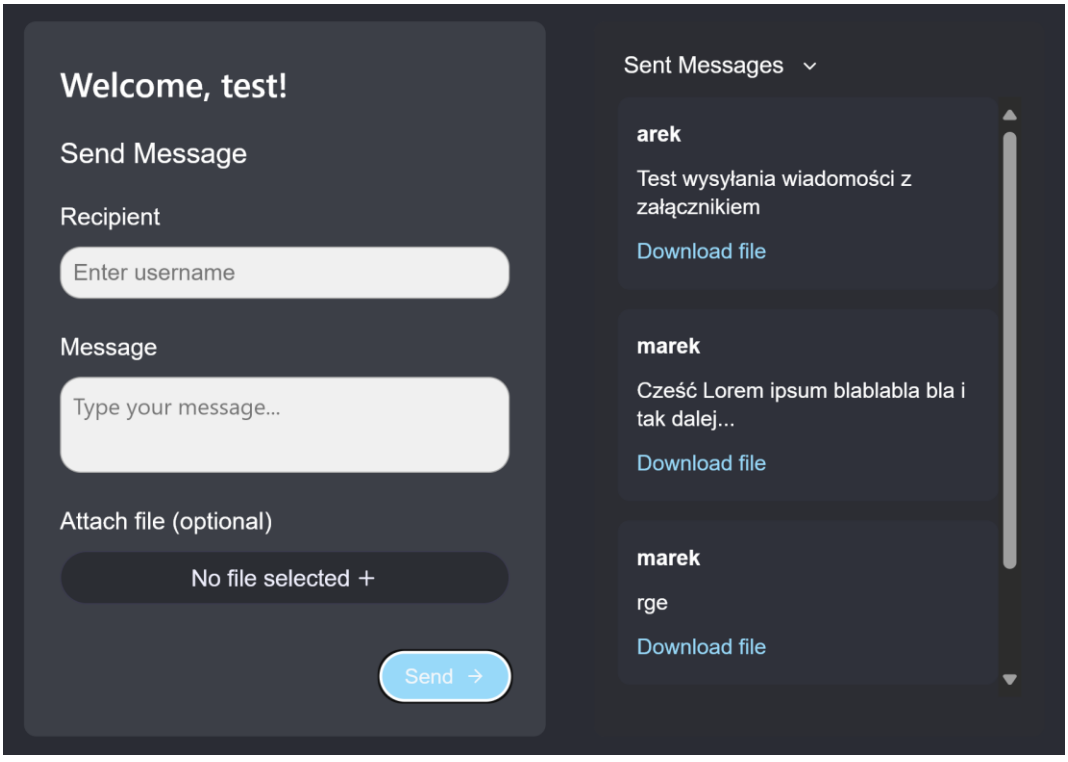


- Nowy widok czatu:

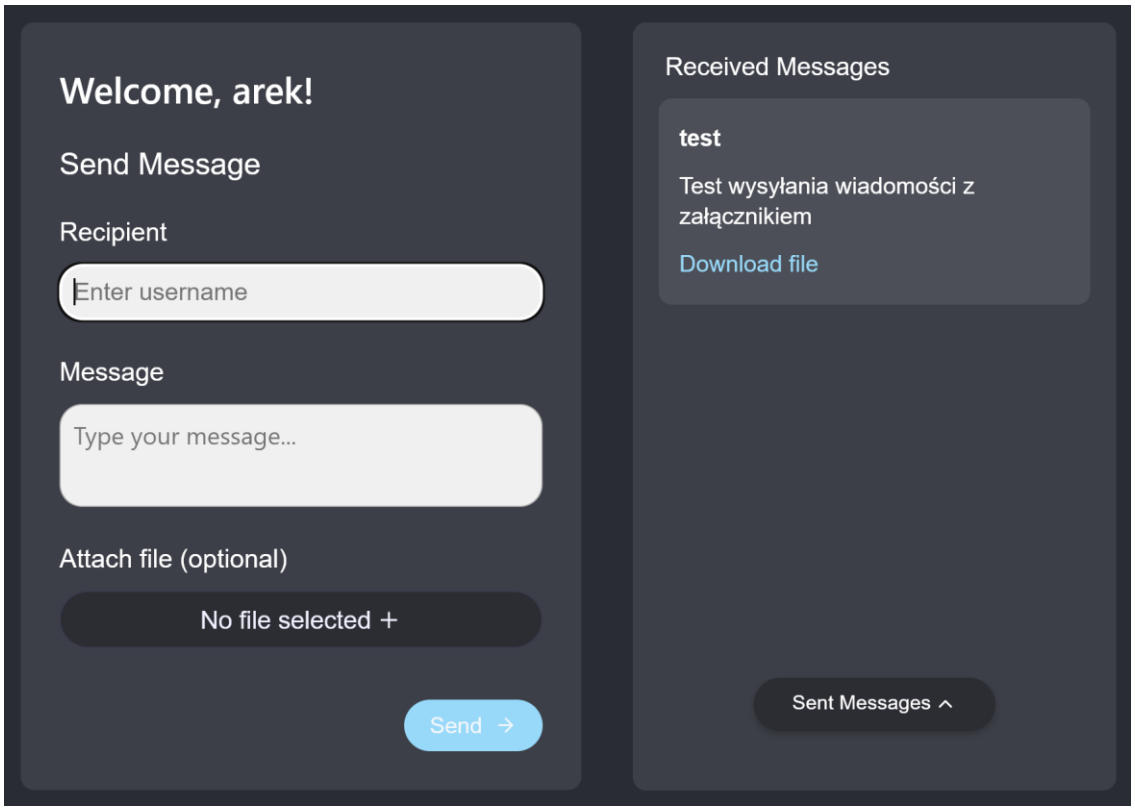



Welcome, test!
Send Message
Recipient
arek
Message
Test wysyłania wiadomości z załącznikiem
Attach file (optional)
GDD concept.pdf
Send
Sent Messages
marek
Cześć Lorem ipsum blablabla bla i tak dalej...
Download file
marek
rge
Download file
marek
dafa

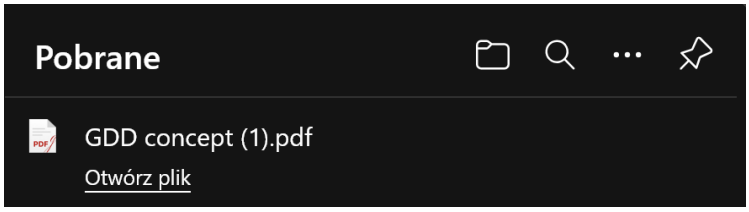

Komunikat z witryny localhost:5173:
Message with file sent successfully!
OK


Welcome, test!
Send Message
Recipient
Enter username
Message
Type your message...
Attach file (optional)
No file selected +
Send
Sent Messages
arek
Test wysyłania wiadomości z załącznikiem
Download file
marek
Cześć Lorem ipsum blablabla bla i tak dalej...
Download file
marek
rge
Download file

Teraz sprawdzamy u arka



Po kliknięciu na przycisk download file:

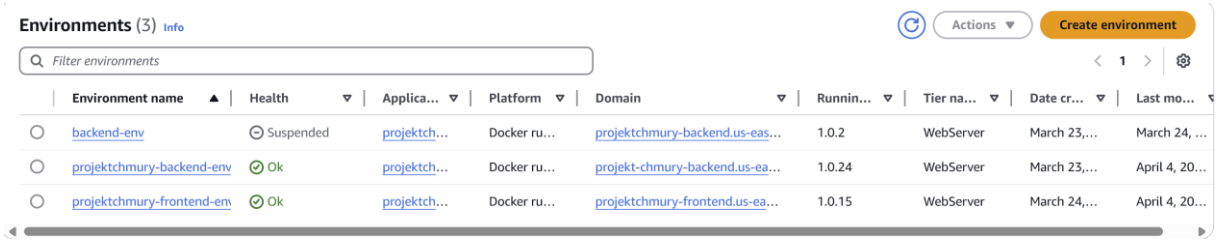


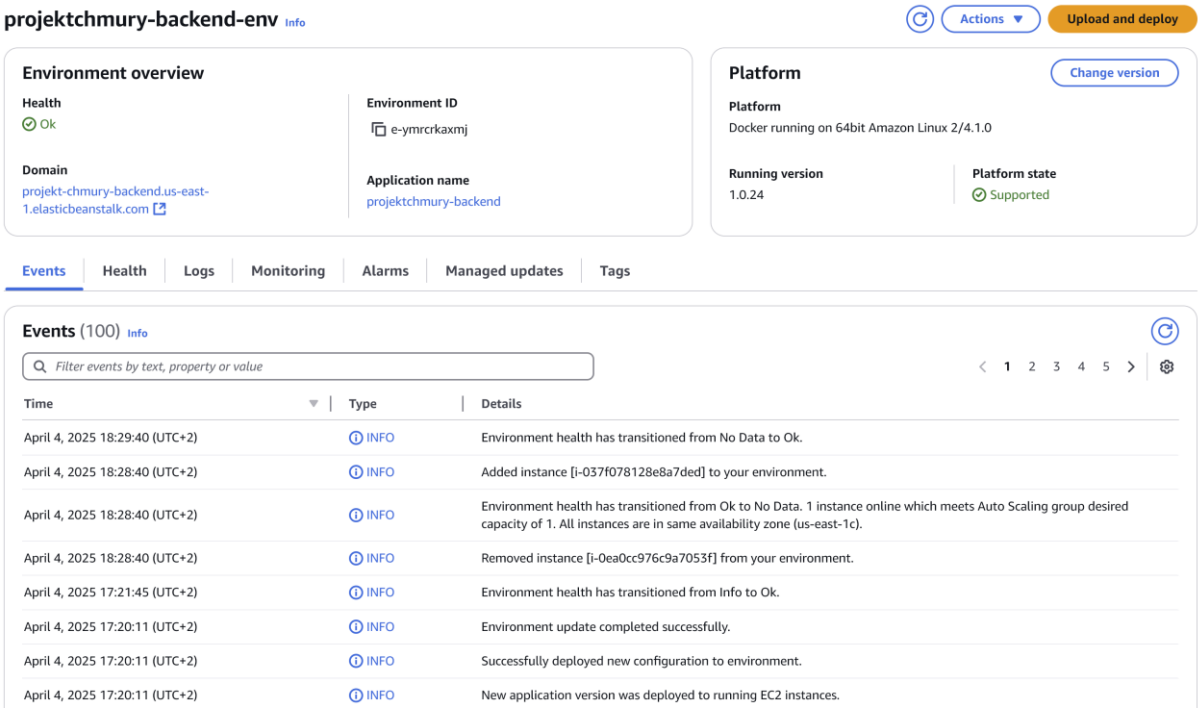
3.2 Konfiguracja AWS przez interfejs webowy

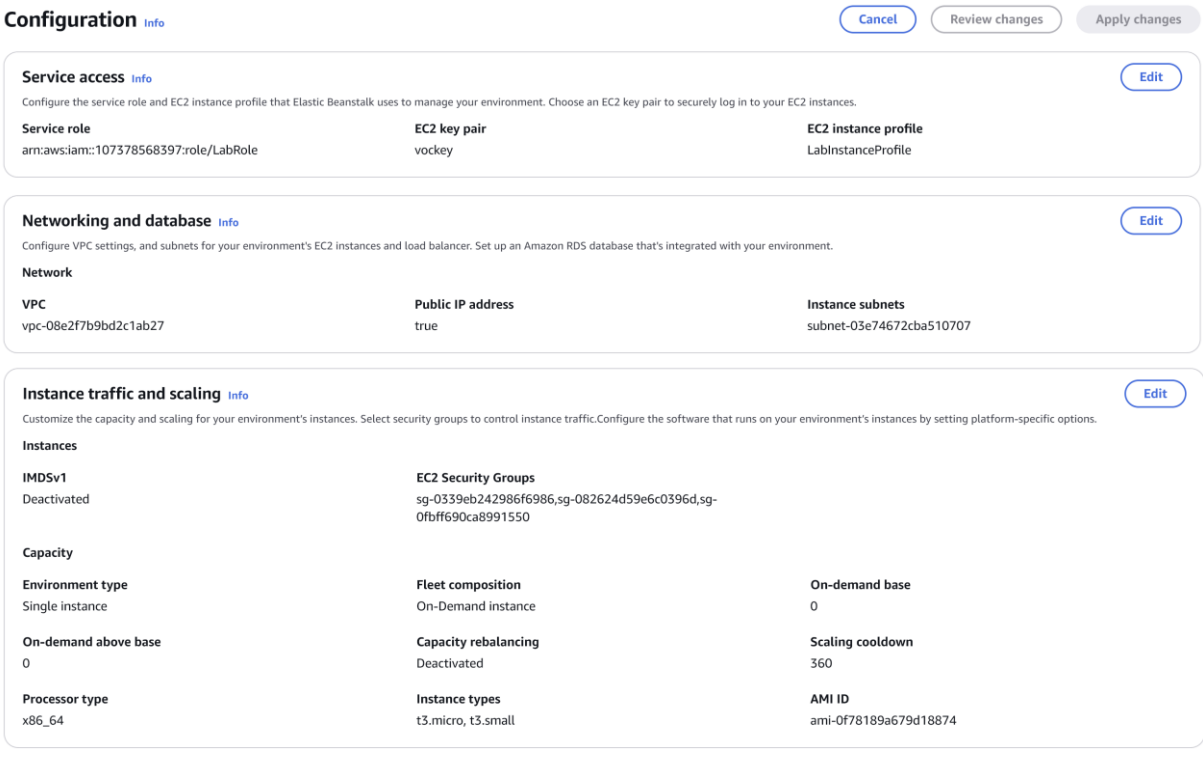
Najpierw wykonana została przeze mnie manualna konfiguracja poprzez konsolę AWS. W raporcie znajdują się zrzuty ekranu przedstawiające między innymi:

- Konfigurację backendu z CloudWatch.
- Integrację frontendu z CloudWatch.
- Konfiguracje wszystkich wymaganych serwisów AWS
- Testowe logowanie oraz przesyłanie wiadomości z załącznikami.
- Weryfikację poprawności działania aplikacji (np. pobieranie plików, wyświetlanie wiadomości w różnych kontach użytkowników).

- Backend w Elastic Beanstalk:







Updates, monitoring, and logging Info

Edit

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

Monitoring

Log group	System	Cloudwatch custom metrics - instance
—	Activated	7

[Alt+S] United States (N. Virginia) voclabs/user3928724=Mateusz_Staszk__w @ 1073-7856-8391

projektchmury-backend-env > Configuration

Lifecycle
false

Updates

Managed updates	Deployment batch size	Deployment batch size type
Activated	100	Percentage
Command timeout	**Deployment policy**	**Health threshold**
600	AllAtOnce	Ok
Ignore health check	**Instance replacement**	
false	false	

Platform software

Lifecycle	Log groups	Log streaming
false	/aws/elasticbeanstalk/projektchmury-backend-env	Activated
Proxy server	**Logs retention**	**Rotate logs**
none	7	Deactivated
Update level	**X-Ray enabled**	
minor	Deactivated	

Environment properties

Source	Key	Value
Plain text	SPRING_DATASOURCE_PASSWORD	admin1234
Plain text	SPRING_DATASOURCE_URL	jdbc:postgresql://baza-chmury.cdle9v0rnyed.us-east-1.rds.a…
Plain text	SPRING_DATASOURCE_USERNAME	postgres

Instance metadata service (IMDS)

Your environment's platform supports both IMDSv1 and IMDSv2. To enforce IMDSv2, deactivate IMDSv1. Learn more

IMDSv1
With the current setting, the environment enables only IMDSv2.
☑ Deactivated

EC2 security groups

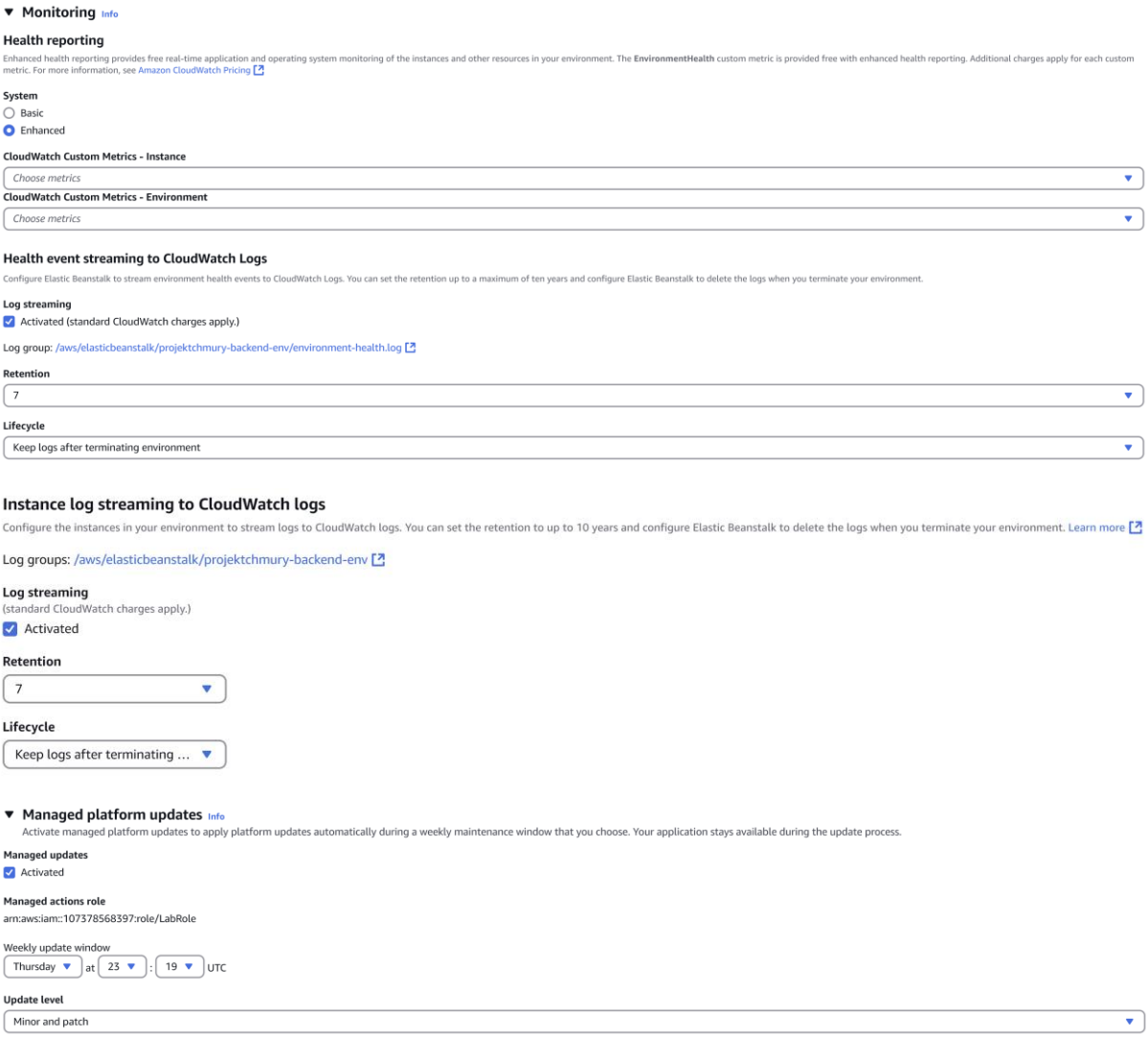
Select security groups to control traffic.

EC2 security groups (6)

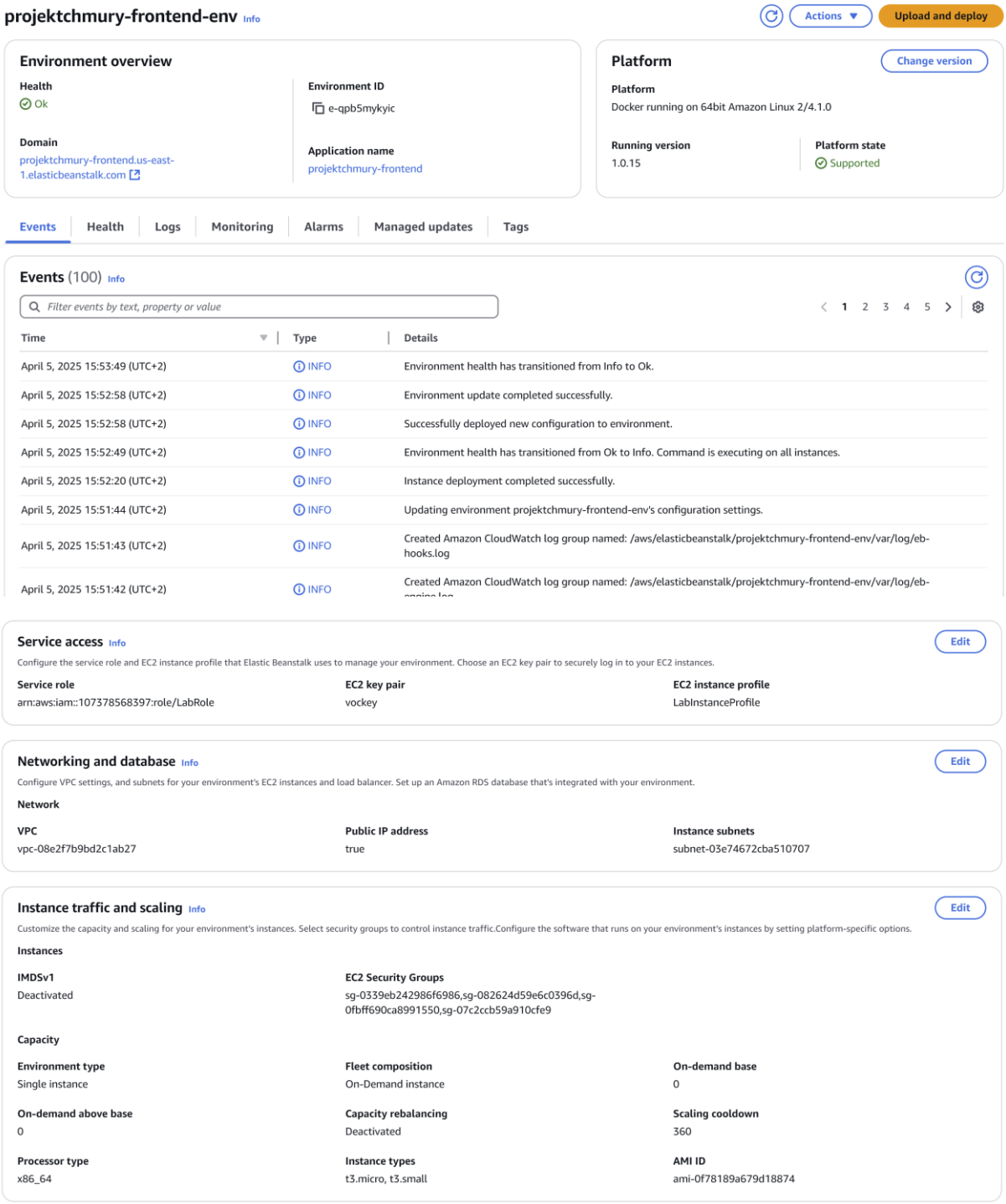
Filter security groups

Group name	Group ID	Name
☑ awseb-e-muct4ccfvm-stack-AWSEBSecurityGroup-iUjAJGVd7mNK	sg-082624d59e6c0396d	backend-env
☐ awseb-e-qpb5mykyic-stack-AWSEBSecurityGroup-HN1oB1NMSpld	sg-07c2ccb59a910cfe9	projektchmury-frontend-env
☑ awseb-e-ymrcrkaxmj-stack-AWSEBSecurityGroup-Q5UoebqzA2Ml	sg-0fbff690ca8991550	projektchmury-backend-env
☑ default	sg-0339eb242986f6986	
☐ ec2-rds-1	sg-082f9d6e474396f0c	
☐ rds-ec2-1	sg-01897eb9d86089958	

Konfiguracja backendu z CloudWatch:



Frontend w Elastic Beanstalk:



projektchmury-frontend-env Info

Actions | Upload and deploy

Environment overview

Health: Ok

Environment ID: e-qpb5mykyic

Domain: projektchmury-frontend.us-east-1.elasticbeanstalk.com

Application name: projektchmury-frontend

Platform — Change version

Platform: Docker running on 64bit Amazon Linux 2/4.1.0

Running version: 1.0.15

Platform state: Supported

Events | Health | Logs | Monitoring | Alarms | Managed updates | Tags

Events (100) Info

Filter events by text, property or value

1 2 3 4 5

Time	Type	Details
April 5, 2025 15:53:49 (UTC+2)	INFO	Environment health has transitioned from Info to Ok.
April 5, 2025 15:52:58 (UTC+2)	INFO	Environment update completed successfully.
April 5, 2025 15:52:58 (UTC+2)	INFO	Successfully deployed new configuration to environment.
April 5, 2025 15:52:49 (UTC+2)	INFO	Environment health has transitioned from Ok to Info. Command is executing on all instances.
April 5, 2025 15:52:20 (UTC+2)	INFO	Instance deployment completed successfully.
April 5, 2025 15:51:44 (UTC+2)	INFO	Updating environment projektchmury-frontend-env's configuration settings.
April 5, 2025 15:51:43 (UTC+2)	INFO	Created Amazon CloudWatch log group named: /aws/elasticbeanstalk/projektchmury-frontend-env/var/log/eb-hooks.log
April 5, 2025 15:51:42 (UTC+2)	INFO	Created Amazon CloudWatch log group named: /aws/elasticbeanstalk/projektchmury-frontend-env/var/log/eb-

Service access Info — Edit

Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Service role: arn:aws:iam::107378568397:role/LabRole

EC2 key pair: vockey

EC2 instance profile: LabInstanceProfile

Networking and database Info — Edit

Configure VPC settings, and subnets for your environment's EC2 instances and load balancer. Set up an Amazon RDS database that's integrated with your environment.

Network

VPC: vpc-08e2f7b9bd2c1ab27

Public IP address: true

Instance subnets: subnet-03e74672cba510707

Instance traffic and scaling Info — Edit

Customize the capacity and scaling for your environment's instances. Select security groups to control instance traffic.Configure the software that runs on your environment's instances by setting platform-specific options.

Instances

IMDSv1: Deactivated

EC2 Security Groups: sg-0339eb242986f6986,sg-082624d59e6c0396d,sg-0fbff690ca8991550,sg-07c2ccb59a910cfe9

Capacity

Environment type: Single instance

Fleet composition: On-Demand instance

On-demand base: 0

On-demand above base: 0

Capacity rebalancing: Deactivated

Scaling cooldown: 360

Processor type: x86_64

Instance types: t3.micro, t3.small

AMI ID: ami-0f78189a679d18874

Updates, monitoring, and logging Info

Edit

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

Monitoring

Log group
/aws/elasticbeanstalk/projektchmury-frontend-env/environment-health.log

System
enhanced

Cloudwatch custom metrics - instance
—

Cloudwatch custom metrics - environment
—

Log streaming
Activated

Retention
7

Lifecycle
false

Updates

Managed updates
Activated

Deployment batch size
100

Deployment batch size type
Percentage

Command timeout
600

Deployment policy
AllAtOnce

Health threshold
Ok

Ignore health check
false

Instance replacement
false

Platform software

Lifecycle
false

Log groups
/aws/elasticbeanstalk/projektchmury-frontend-env

Log streaming
Activated

none

/

Deactivated

Update level
minor

X-Ray enabled
Deactivated

Environment properties

Source	Key	Value
Plain text	VITE_API_URL	projekt-chmury-backend.us-east-1.elasticbeanstalk.com

Instance metadata service (IMDS)

Your environment's platform supports both IMDSv1 and IMDSv2. To enforce IMDSv2, deactivate IMDSv1. Learn more

IMDSv1
With the current setting, the environment enables only IMDSv2.

☑ Deactivated

EC2 security groups

Select security groups to control traffic.

EC2 security groups (6)

Filter security groups

Group name	Group ID	Name
☑ awseb-e-muct4ccfvm-stack-AWSEBSecurityGroup-iUjAJGVd7mNK	sg-082624d59e6c0396d	backend-env
☑ awseb-e-qpb5mykyic-stack-AWSEBSecurityGroup-HN1oB1NMSpld	sg-07c2ccb59a910cfe9	projektchmury-frontend-env
☑ awseb-e-ymrcrkaxmj-stack-AWSEBSecurityGroup-QSUoebqzA2Ml	sg-0fbff690ca8991550	projektchmury-backend-env
☑ default	sg-0339eb242986f6986	
☐ ec2-rds-1	sg-082f9d6e474396f0c	
☐ rds-ec2-1	sg-01897eb9d86089958	

Instance types

Add instance types for your environment with your preferred launch order. The order preference only applies to On-Demand Ins

1. t3.micro
2. t3.small Remove

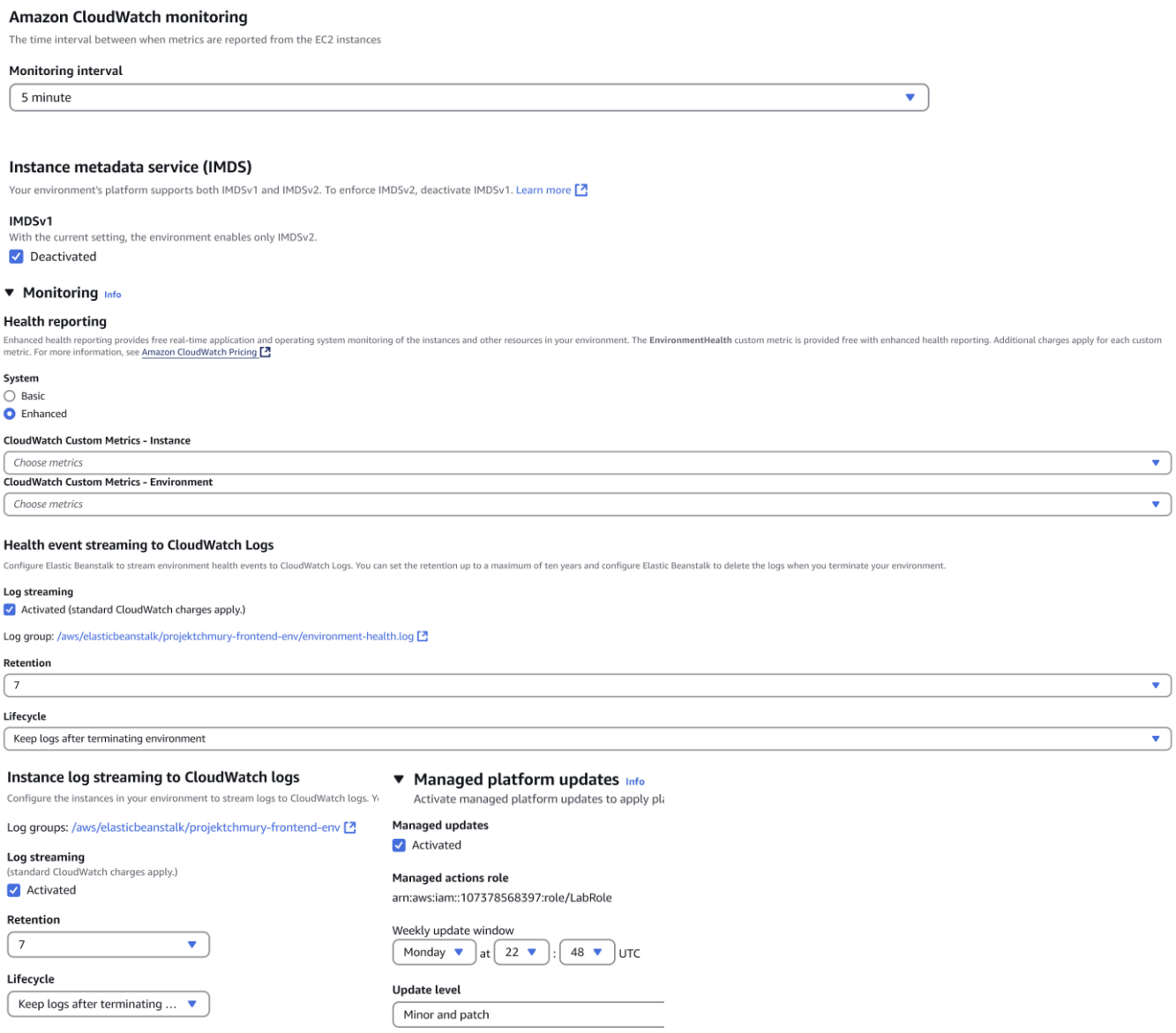
Add instance type

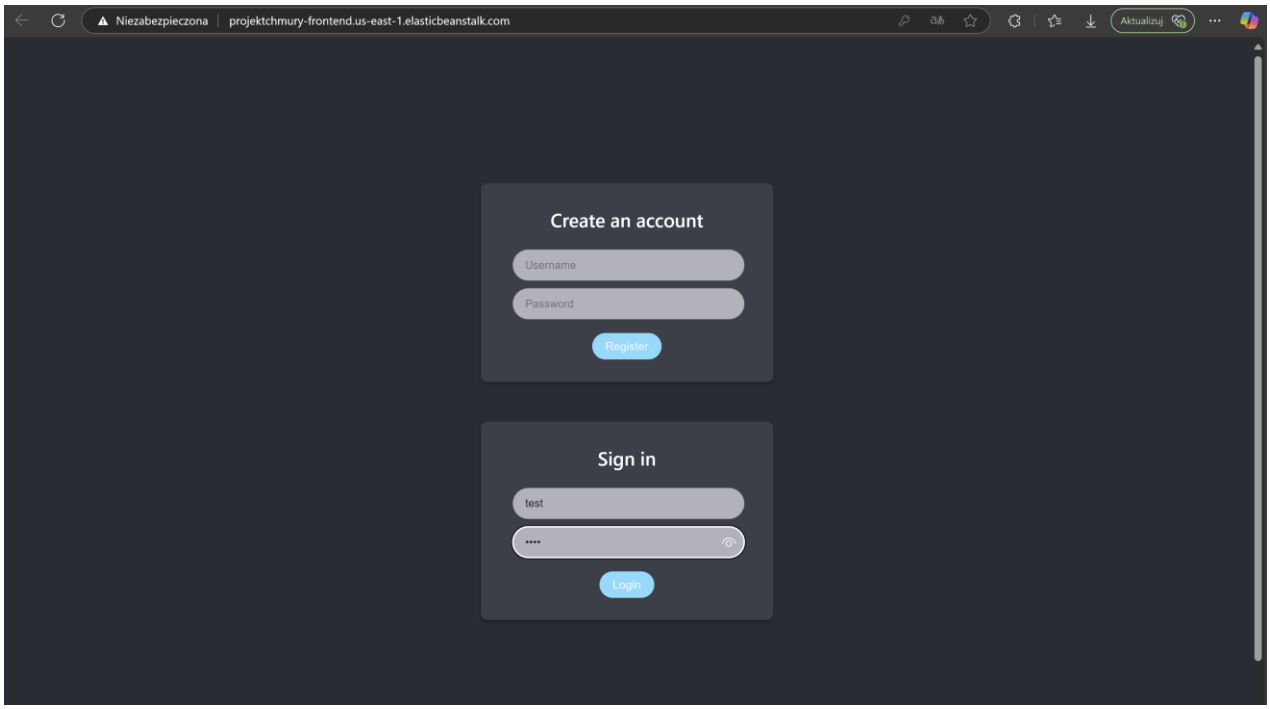
AMI ID

Elastic Beanstalk selects a default Amazon Machine Image (AMI) for your environment based on the Region, platform version, a

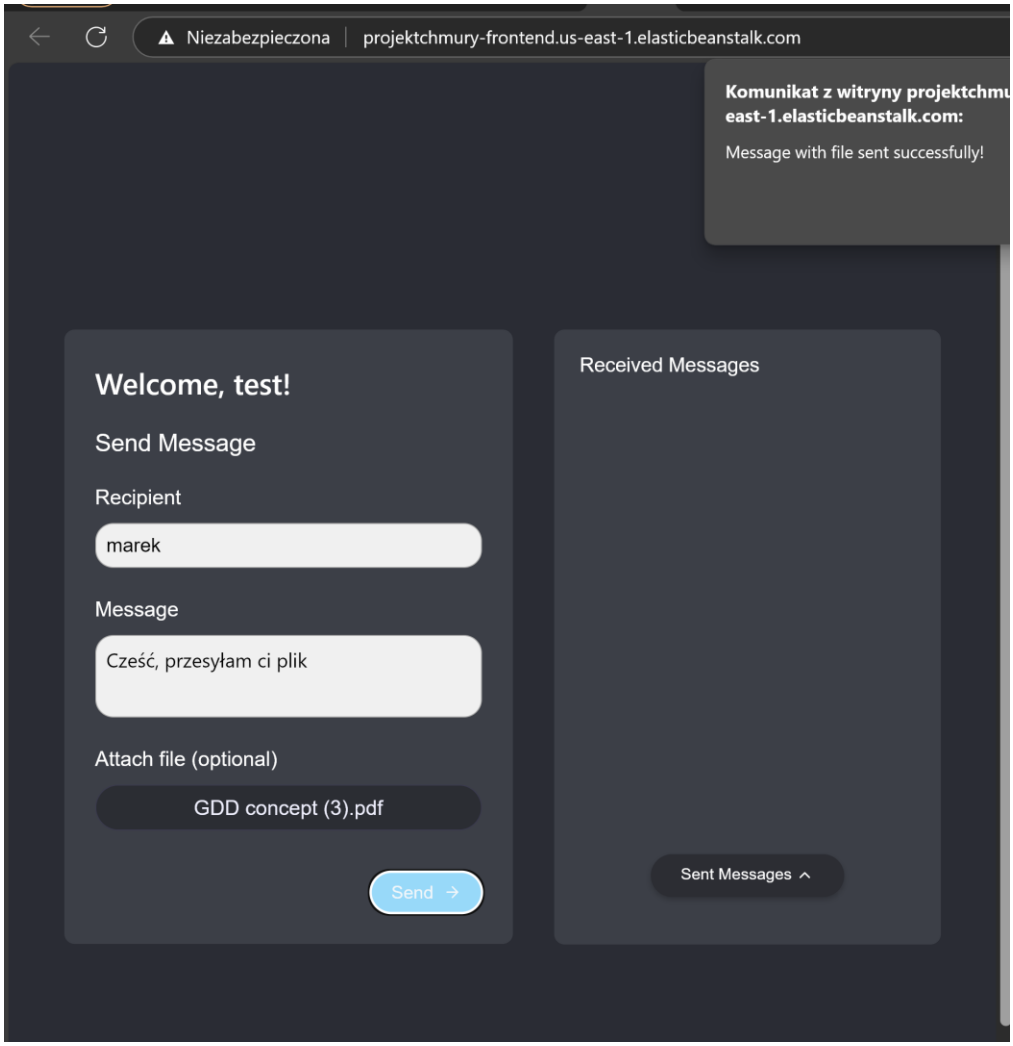
ami-0f78189a679d18874

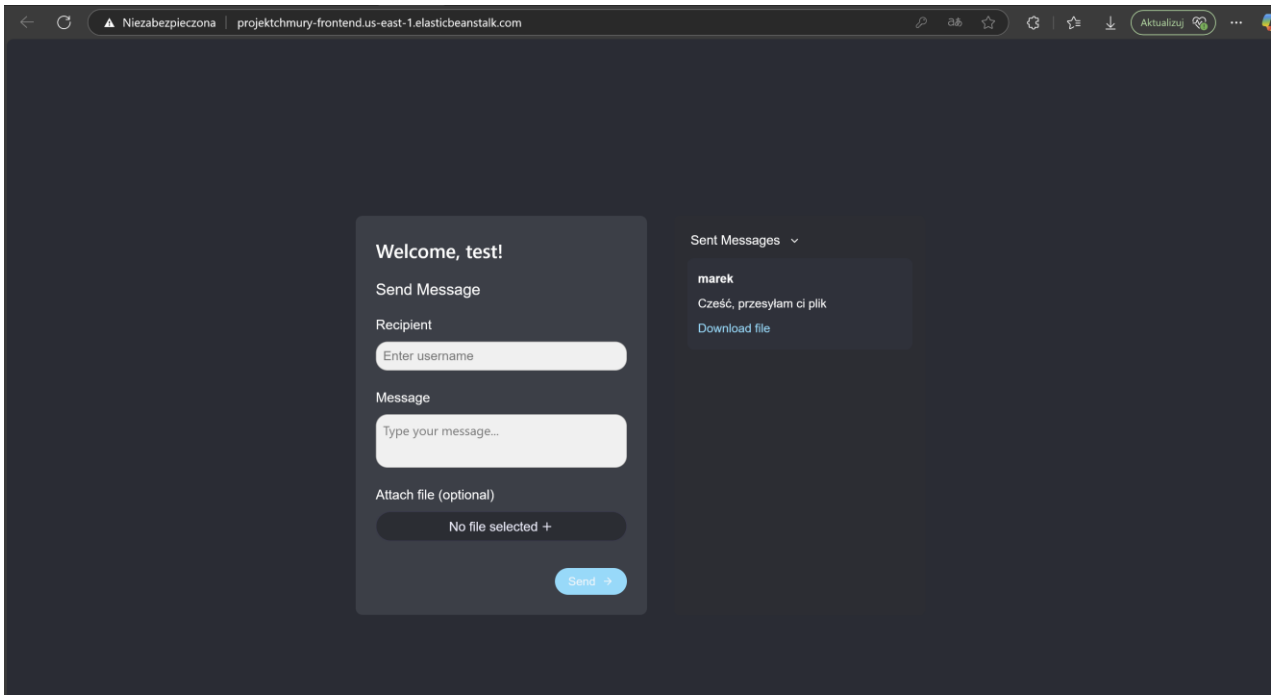
Integracja frontendu z Cloudwatch:



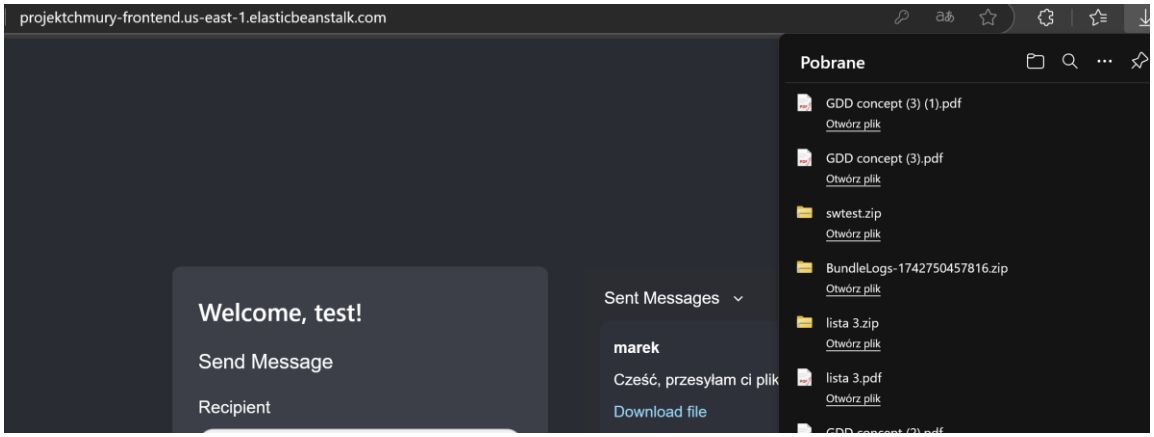


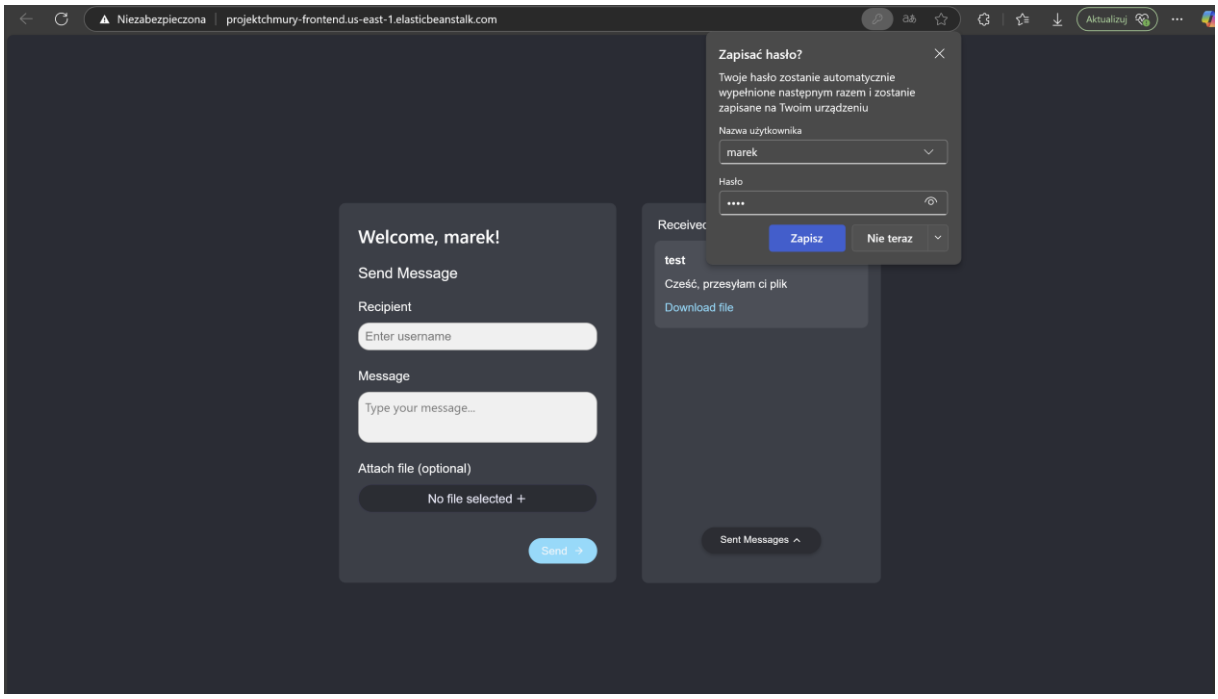
Działający frontend i backend na AWS wchodząc a link dla frontendu, po zalogowaniu wysyłamy wiadomość do innego użytkownika (**test** do **marek**).



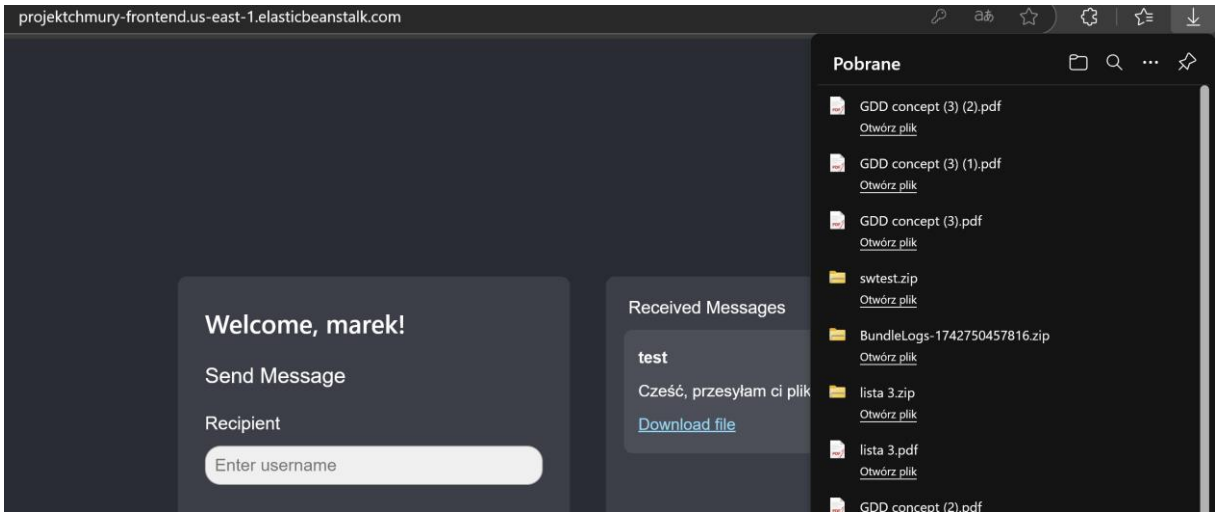


Po kliknięciu Download file plik się pobiera



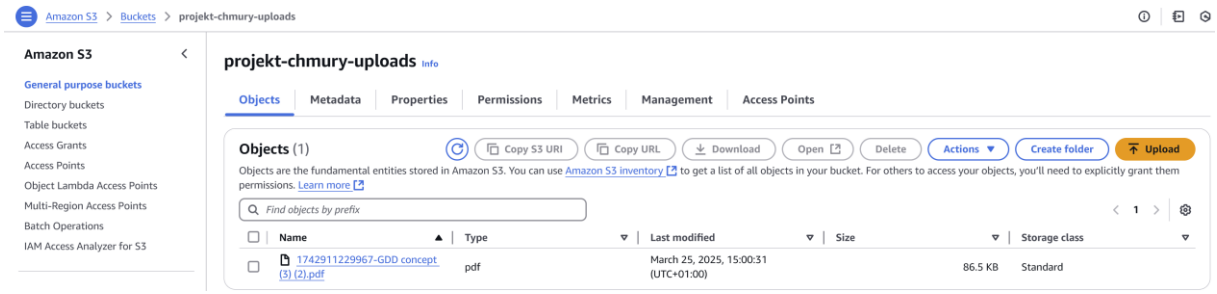


Po zalogowaniu na drugie konto mamy wiadomość od test w otrzymanych wiadomościach



I również plik pobiera się bez problemu

- Przesyłane pliki przechowują się w S3:



Bucket overview

AWS Region
US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)
arn:aws:s3:::projekt-chmury-uploads

Creation date
March 18, 2025, 19:41:54 (UTC+01:00)

Bucket Versioning

Edit

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Disabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more
Disabled

Bucket overview

AWS Region
US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)
arn:aws:s3:::projekt-chmury-uploads

Creation date
March 18, 2025, 19:41:54 (UTC+01:00)

Block public access (bucket settings)

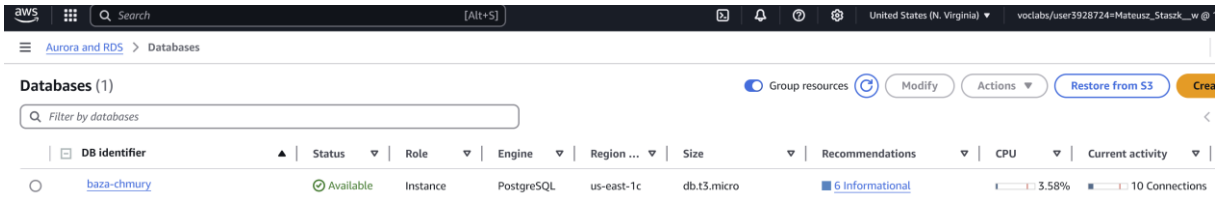
Edit

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

Block *all* public access
Off

▶ **Individual Block Public Access settings for this bucket**

- RDS:



Connectivity & security

Endpoint & port

Endpoint
baza-chmury.cdle9v0rnyed.us-east-1.rds.amazonaws.com

Port
5432

Networking

Availability Zone
us-east-1c

VPC
vpc-08e2f7b9bd2c1ab27

Subnet group
default

Subnets
subnet-0c6e24690d329a7e7
subnet-054fc5687b1b5f979
subnet-03e74672cba510707
subnet-067e210886b5cdd71
subnet-0dd7000c3bab5adbd
subnet-0fee47b94f4f1394f

Network type
IPv4

Security

VPC security groups
awseb-e-qpb5mykyic-stack-AWSEBSecurityGroup-HN1oB1NMSpld (sg-07c2ccb59a910cfe9)
Active
rds-ec2-1 (sg-01897eb9d86089958)
Active
default (sg-0339eb242986f6986)
Active
awseb-e-ymrcrkaxmj-stack-AWSEBSecurityGroup-QSUoebqzA2Ml (sg-0fbff690ca8991550)
Active
awseb-e-muct4ccfvm-stack-AWSEBSecurityGroup-iUjAJGVd7mNK (sg-082624d59e6c0396d)
Active

Publicly accessible
Yes

Certificate authority Info
rds-ca-rsa2048-g1

Certificate authority date
May 26, 2061, 01:34 (UTC+02:00)

DB instance certificate expiration date
March 24, 2026, 00:17 (UTC+01:00)

Security group rules (7)

Filter by Security group rules

1

Security group	Type	Rule
awseb-e-muct4ccfvm-stack-AWSEBSecurityGroup-iUjAJGVd7mNK (sg-082624d59e6c0396d)	CIDR/IP - Outbound	0.0.0.0/0
awseb-e-qpb5mykyic-stack-AWSEBSecurityGroup-HN1oB1NMSpld (sg-07c2ccb59a910cfe9)	CIDR/IP - Outbound	0.0.0.0/0
awseb-e-ymrcrkaxmj-stack-AWSEBSecurityGroup-QSUoebqzA2Ml (sg-0fbff690ca8991550)	CIDR/IP - Outbound	0.0.0.0/0
default (sg-0339eb242986f6986)	EC2 Security Group - Inbound	sg-0339eb242986f6986
default (sg-0339eb242986f6986)	CIDR/IP - Outbound	0.0.0.0/0
rds-ec2-1 (sg-01897eb9d86089958)	CIDR/IP - Inbound	109.243.146.84/32
rds-ec2-1 (sg-01897eb9d86089958)	EC2 Security Group - Inbound	sg-082f9d6e474396f0c

Instance

Configuration

DB instance ID
baza-chmury

Engine version
14.15

RDS Extended Support
Enabled

DB name
mydatabase

License model
Postgresql License

Option groups
default:postgres-14 In sync

Amazon Resource Name (ARN)
arn:aws:rds:us-east-1:107378568397:db:baza-chmury

Resource ID
db-CWQQ3SNTB7BQ7HZFXY7VBDRG3M

Created time
March 18, 2025, 18:52 (UTC+01:00)

DB instance parameter group
default.postgres14 In sync

Deletion protection
Disabled

Architecture settings
Non-multitenant architecture

Instance class

Instance class
db.t3.micro

vCPU
2

RAM
1 GB

Availability

Master username
postgres

Master password
****

IAM DB authentication
Not enabled

Multi-AZ
No

Secondary Zone
-

Storage

Encryption
Not enabled

Storage type
General Purpose SSD (gp2)

Storage
20 GiB

Provisioned IOPS
-

Storage throughput
-

Storage autoscaling
Disabled

Storage file system configuration
Current

Monitoring

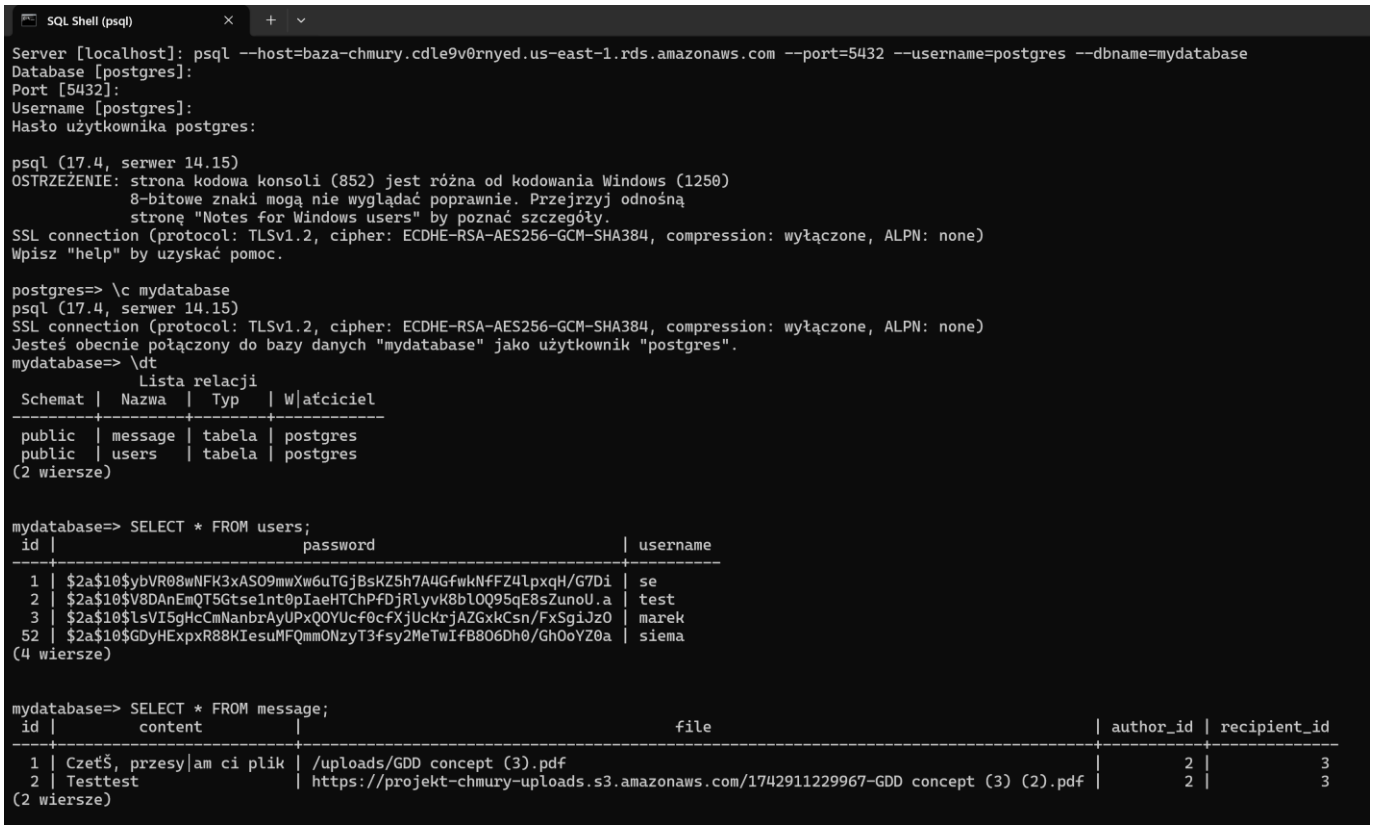
Monitoring type
Database Insights - Standard

Performance Insights
Disabled

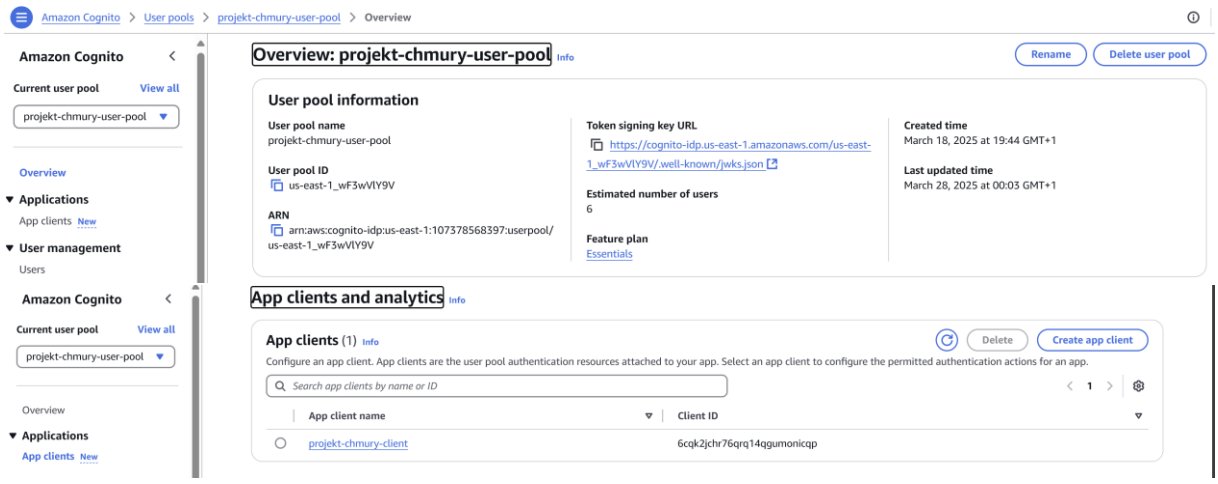
Enhanced Monitoring
Disabled

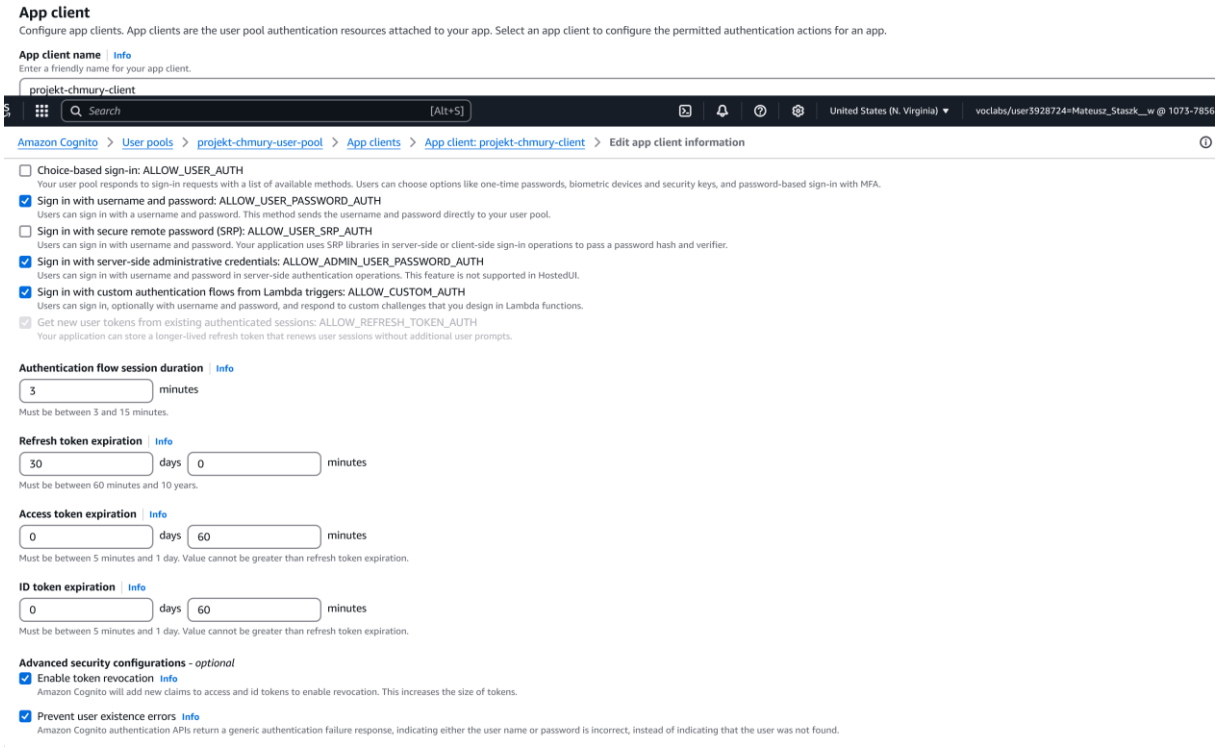
DevOps Guru
Disabled

Potwierdzenie poprawnej konfiguracji z RDS i prawidłowego przechowywania danych łącząc się z RDS poprzez psql:

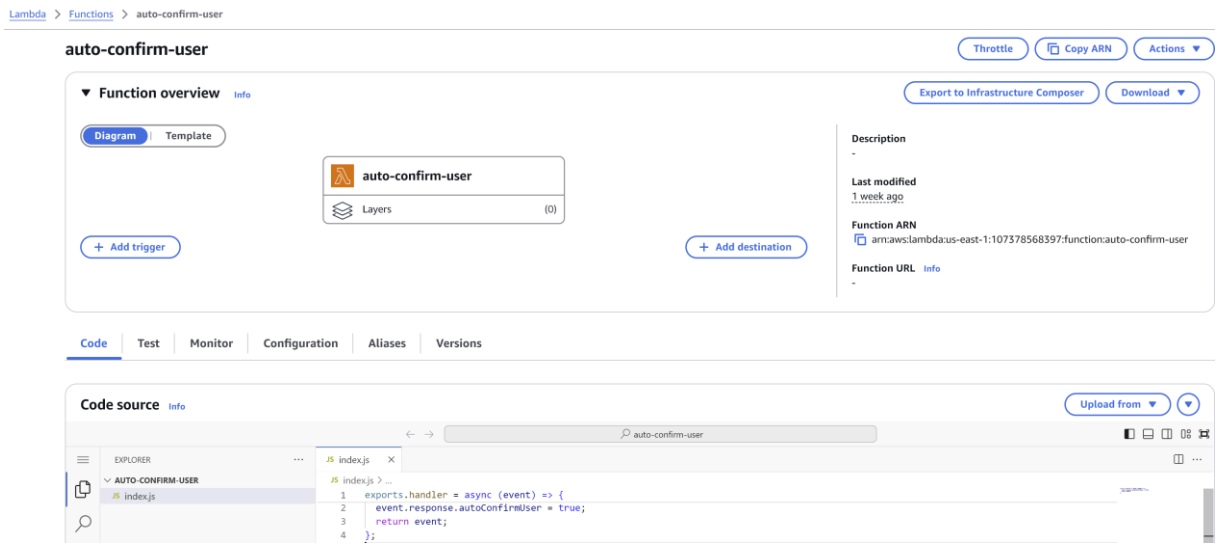


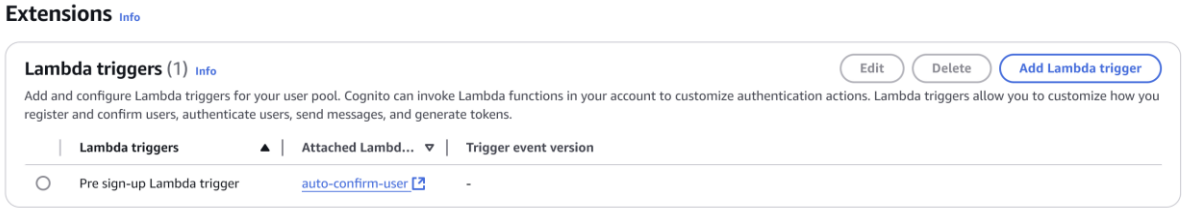
- Cognito:



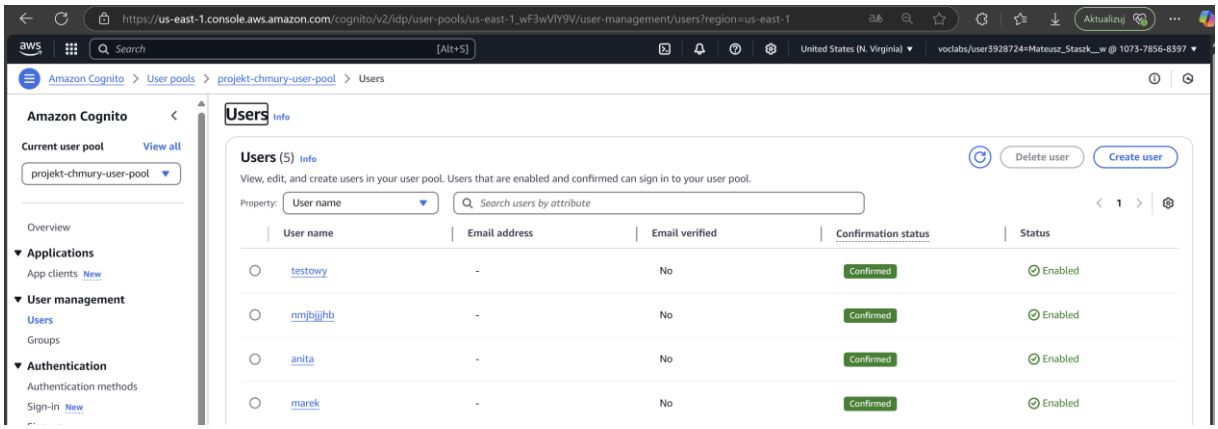


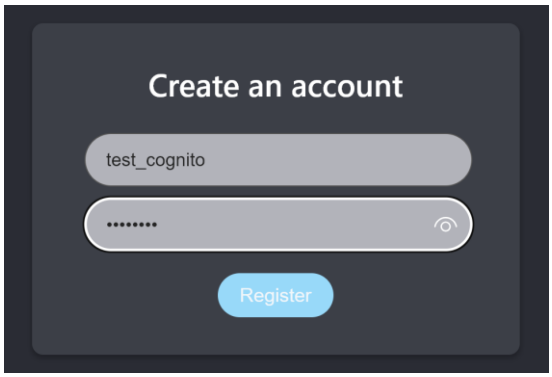
Warto zaznaczyć, że żeby automatycznie potwierdzać nowo założone konta (bez potwierdzenia konta nie będzię się dało na nie zalogować) musimy napisać własną funkcję Lambda, którą dodamy jako Extention do Cognito User Pool:

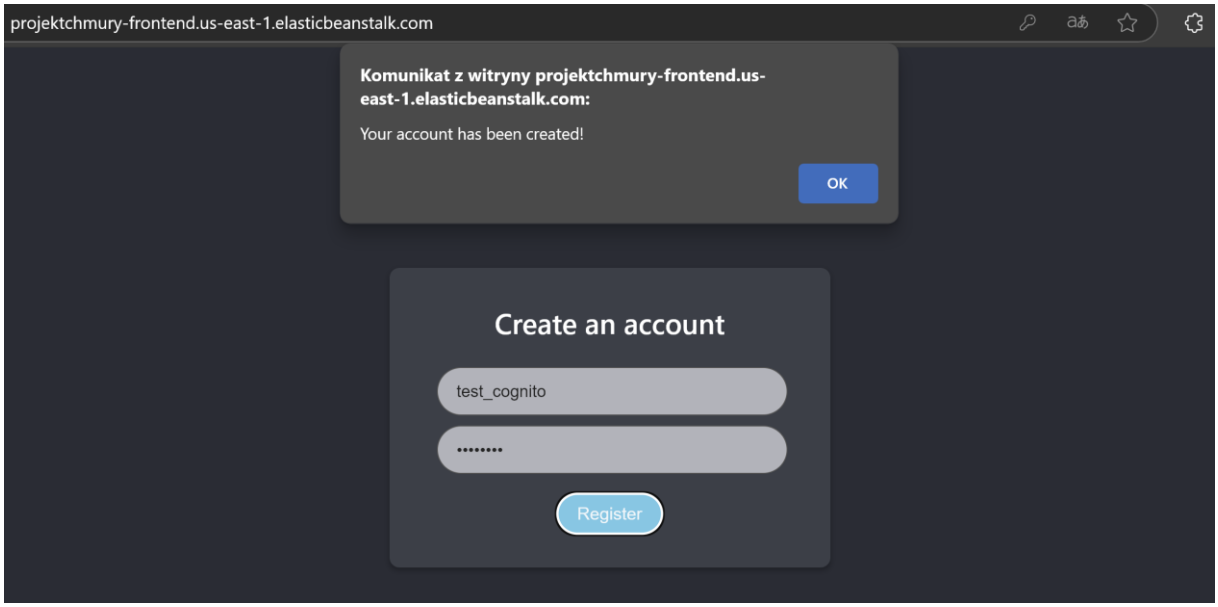


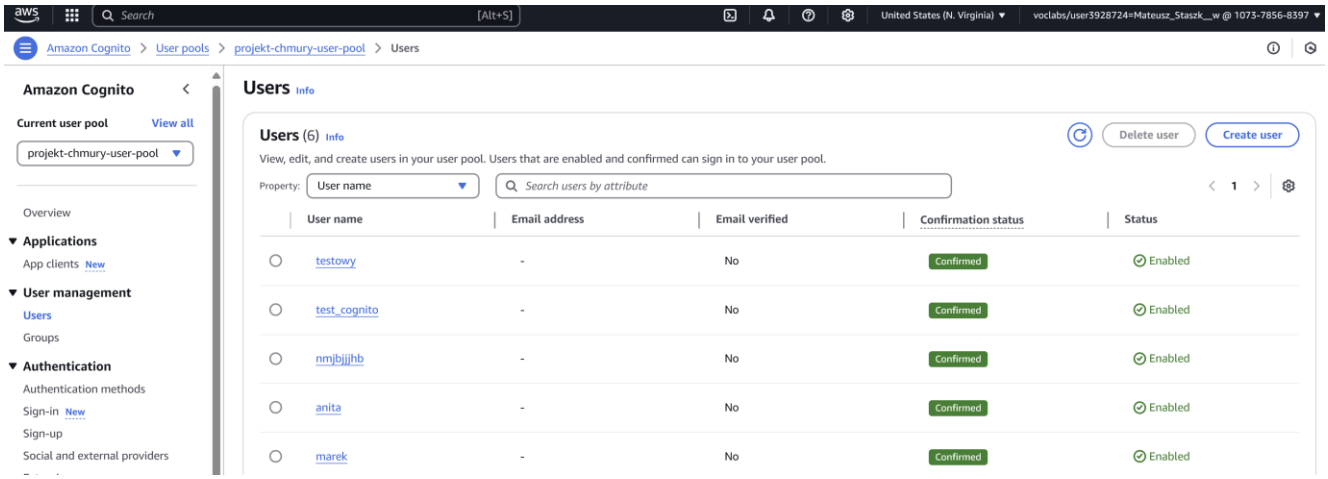


Spróbujmy założyć nowego użytkownika i zobaczymy czy pojawi się jego instancja na stronie AWS Cognito

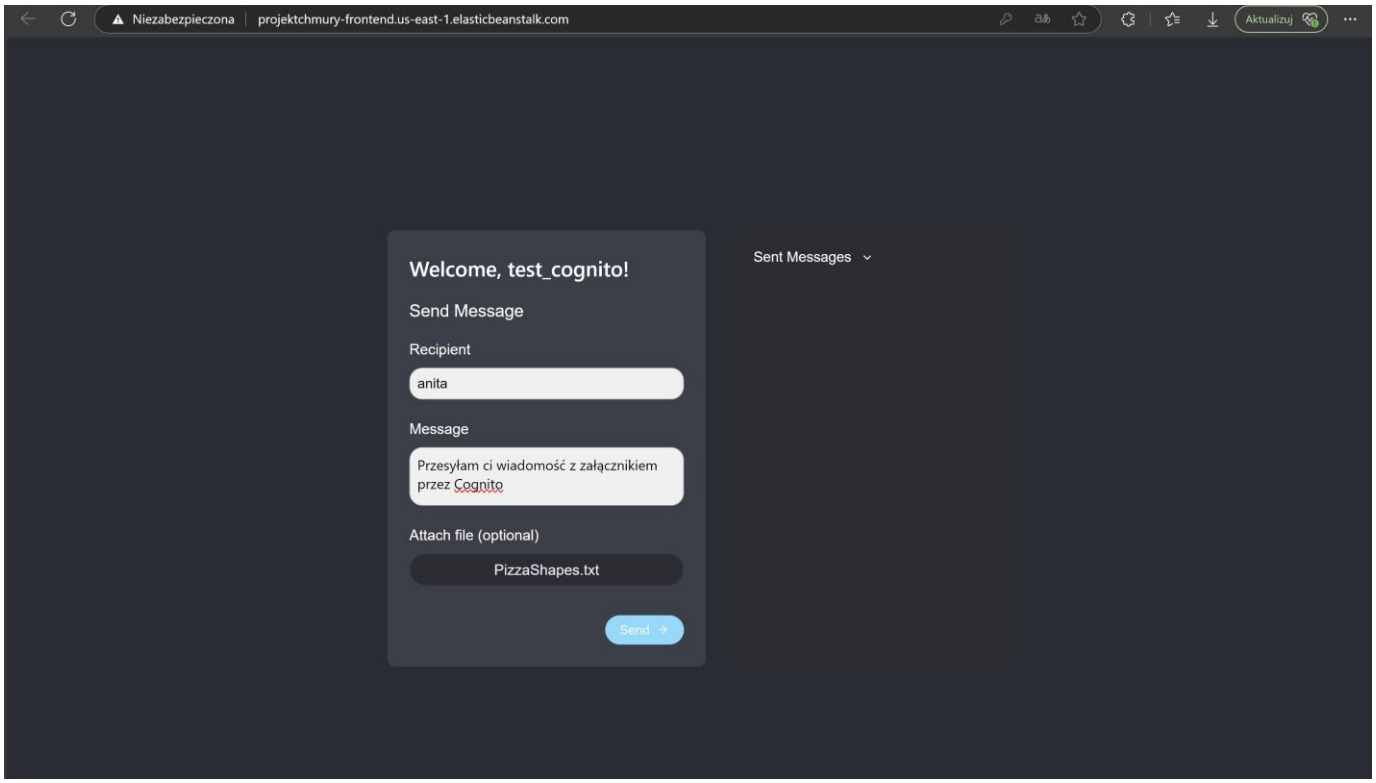

https://us-east-1.console.aws.amazon.com/cognito/v2/idp/user-pools/us-east-1_wF3wVIY9V/user-management/users?region=us-east-1
Amazon Cognito
User pools
projekt-chmury-user-pool
Users
Amazon Cognito
Current user pool
View all
projekt-chmury-user-pool
Overview
Applications
App clients New
User management
Users
Groups
Authentication
Authentication methods
Sign-in New
Users Info
Users (5) Info
View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.
Delete user
Create user
Property: User name
Search users by attribute
User name
Email address
Email verified
Confirmation status
Status
testowy - No Confirmed Enabled
nmjbjjjhb - No Confirmed Enabled
anita - No Confirmed Enabled
marek - No Confirmed Enabled

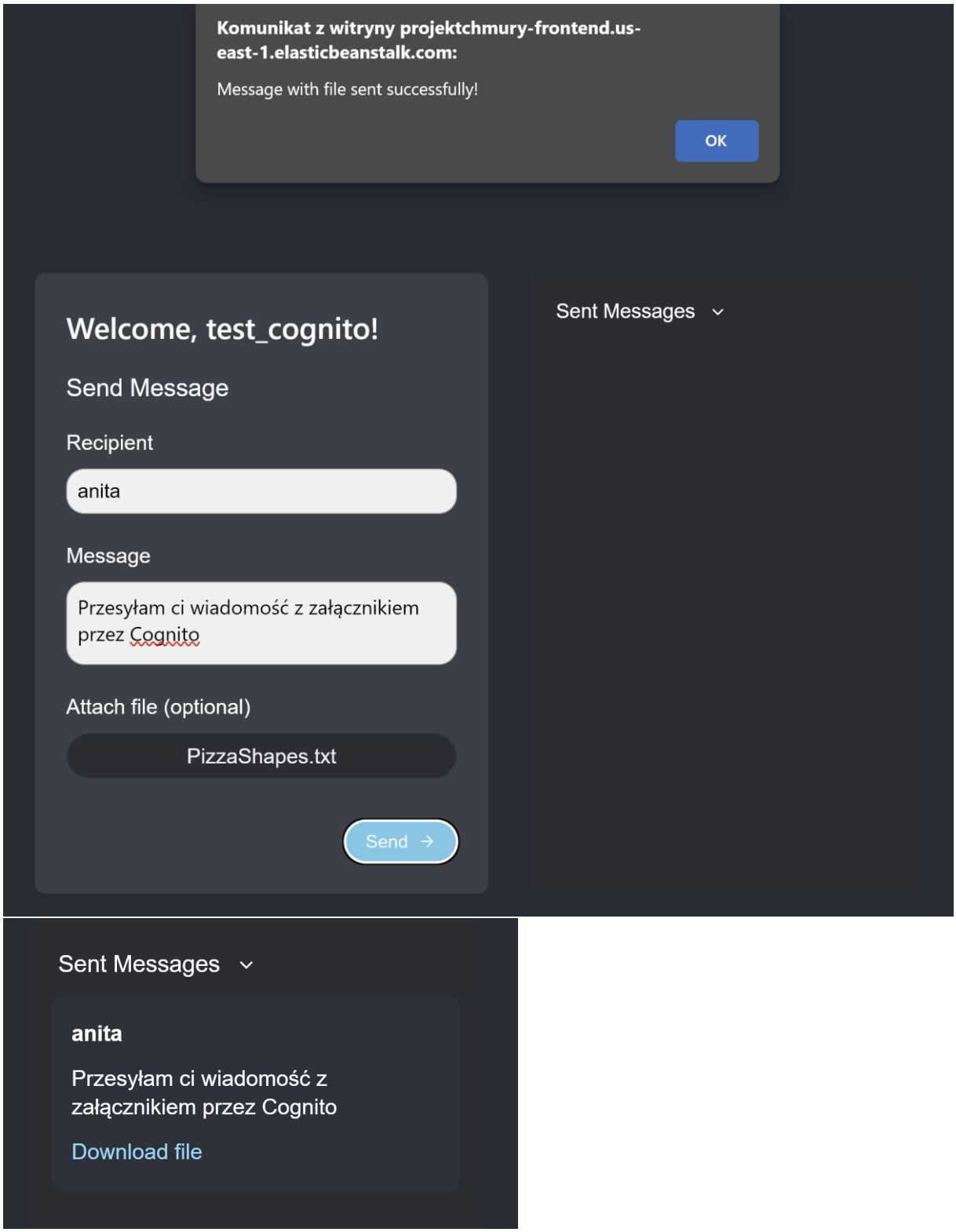

Create an account
test_cognito
Register


projektchmury-frontend.us-east-1.elasticbeanstalk.com
Komunikat z witryny projektchmury-frontend.us-east-1.elasticbeanstalk.com:
Your account has been created!
OK
Create an account
test_cognito
Register



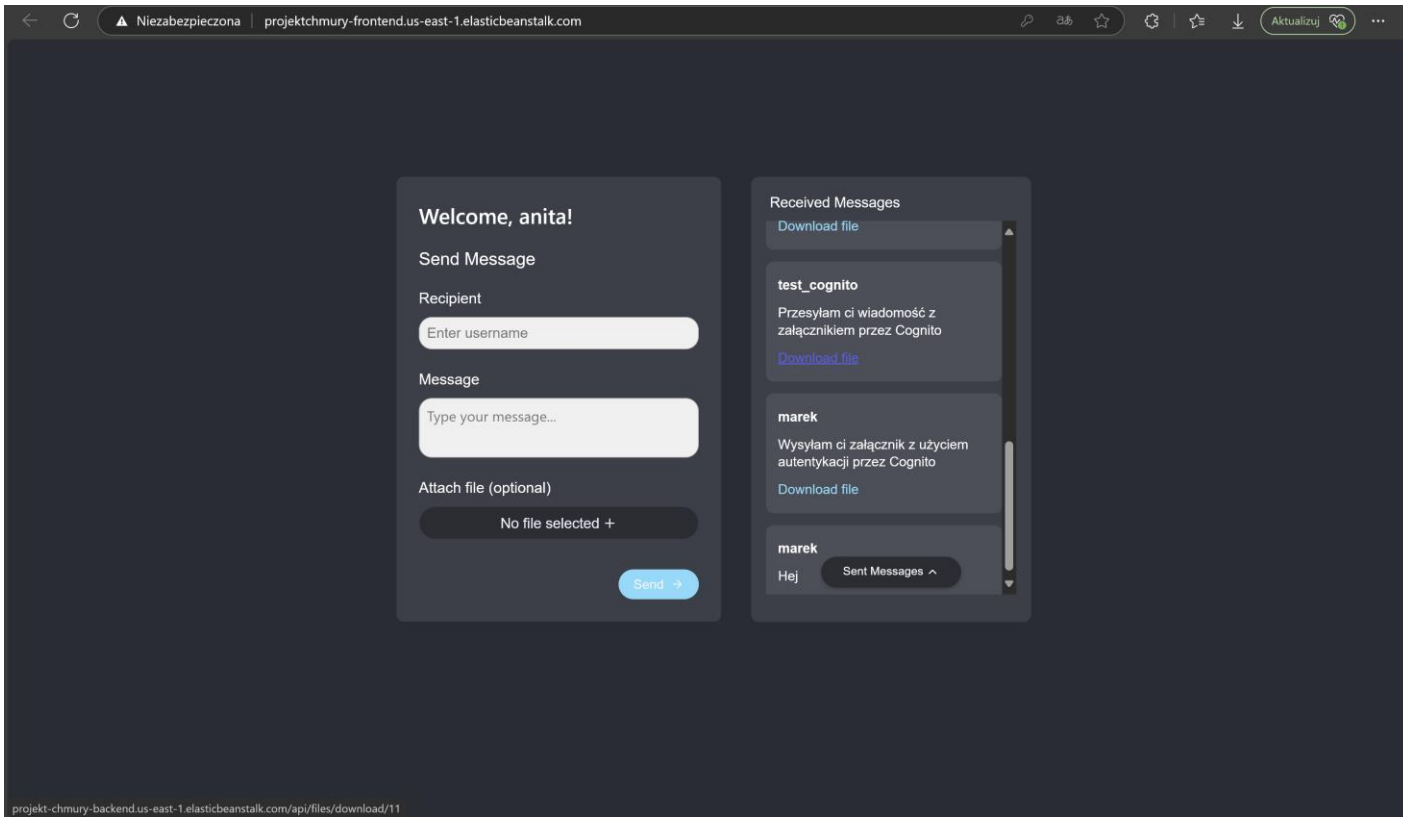
Konto pojawia się w Cognito User Pool i automatycznie jest potwierdzone



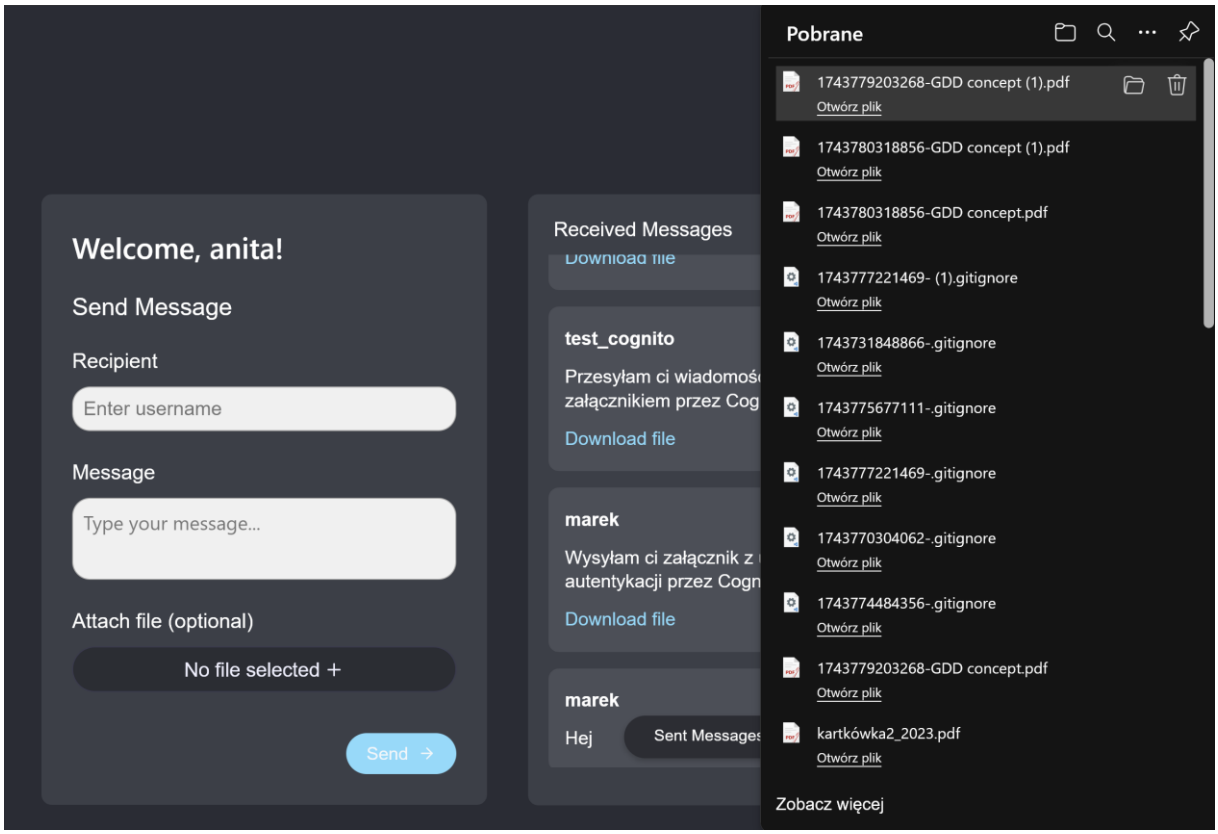

Komunikat z witryny projektchmury-frontend.us-east-1.elasticbeanstalk.com:
Message with file sent successfully!
OK
Welcome, test_cognito!
Send Message
Recipient
anita
Message
Przesyłam ci wiadomość z załącznikiem przez Cognito
Attach file (optional)
PizzaShapes.txt
Send
Sent Messages

Sent Messages
anita
Przesyłam ci wiadomość z załącznikiem przez Cognito
Download file

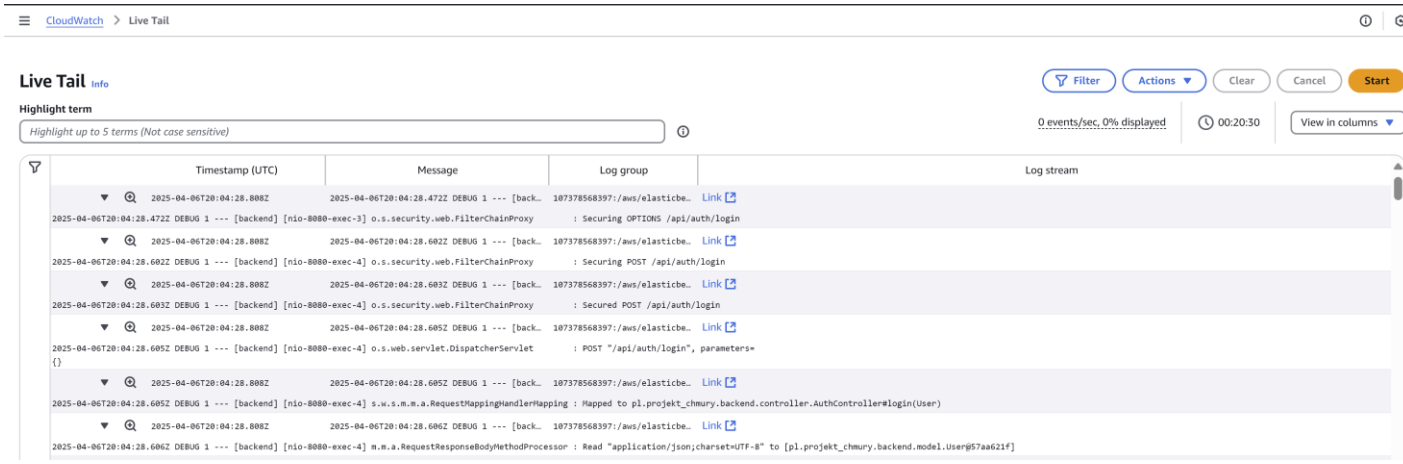
I jak zalogujemy się na konto anita to również mamy wiadomość od test_cognito



I rzeczywiście plik się pobiera, a więc wszystko działa tak jak należy



- Cloudwatch i pojawiające się logi potwierdzające poprawne podłączenie:



3.3 Infrastruktura za pomocą Terraforma

- Za pomocą takiego main.tf:

```
provider "aws" {
  region = "us-east-1"
}

############################
# Losowy sufiks do nazw #
############################
resource "random_string" "suffix" {
  length  = 4
  special = false
  upper   = false
}

##############################
# RDS — PostgreSQL instance #
##############################
resource "aws_db_instance" "mydb" {
  identifier        = "terraform-mydb-${random_string.suffix.result}"
  allocated_storage    = 20
  engine            = "postgres"
  engine_version       = "14.15"
  instance_class       = "db.t3.micro"
  db_name           = "mydatabase"
```

```
  username            = "postgres"
  password            = "admin1234"
  parameter_group_name = "default.postgres14"
  skip_final_snapshot  = true
  publicly_accessible = true

  # Włącz eksport logów do CloudWatch
  enabled_cloudwatch_logs_exports = ["postgresql", "upgrade"]
}

output "db_endpoint" {
  value = aws_db_instance.mydb.address
}

######################
# S3 — File uploads & App Versions
######################
resource "aws_s3_bucket" "upload_bucket" {
  bucket = "terraform-projekt-chmury-uploads-${random_string.suffix.result}"
}

# Przesyłanie plików do S3
resource "aws_s3_object" "backend_app_zip" {
  bucket = aws_s3_bucket.upload_bucket.bucket
  key    = "backend-app.zip"
  source = "../backend/backend-app.zip"
  etag   = filemd5("../backend/backend-app.zip")
}

resource "aws_s3_object" "frontend_app_zip" {
  bucket = aws_s3_bucket.upload_bucket.bucket
  key    = "frontend-app.zip"
  source = "../frontend/frontend-app.zip"
  etag   = filemd5("../frontend/frontend-app.zip")
}

############################
# CloudWatch — Log Group  #
############################
resource "aws_cloudwatch_log_group" "app_logs" {
  name              = "/terraform/projekt-chmury/app-logs-${random_string.suffix.result}"
  retention_in_days = 30
}

############################
# Cognito — User Pool and Client
############################

resource "aws_cognito_user_pool_client" "chat_pool_client" {
  name         = "terraform-projekt-chmury-client-${random_string.suffix.result}"
  user_pool_id = aws_cognito_user_pool.chat_pool.id
  explicit_auth_flows = ["ALLOW_USER_PASSWORD_AUTH", "ALLOW_REFRESH_TOKEN_AUTH"]
}

#######################
# ECR Repositories
#######################
```

```
resource "aws_ecr_repository" "backend_repo" {
  name = "terraform-projekt-chmury-backend-${random_string.suffix.result}"
}

resource "aws_ecr_repository" "frontend_repo" {
  name = "terraform-projekt-chmury-frontend-${random_string.suffix.result}"
}

########################################
# Elastic Beanstalk — Backend Application
########################################
resource "aws_elastic_beanstalk_application" "backend_app" {
  name        = "terraform-backend-app-${random_string.suffix.result}"
  description = "Backend for Projekt Chmury"
}

resource "aws_elastic_beanstalk_application_version" "backend_app_version" {
  name        = "terraform-backend-app-v1-${filemd5("../backend/backend-app.zip")}"
  application = aws_elastic_beanstalk_application.backend_app.name
  bucket      = aws_s3_bucket.upload_bucket.bucket
  key         = aws_s3_object.backend_app_zip.key
  description = "Backend application version 1"
}

resource "aws_elastic_beanstalk_environment" "backend_env" {
  name                = "terraform-backend-env-${random_string.suffix.result}"
  application         = aws_elastic_beanstalk_application.backend_app.name
  solution_stack_name = "64bit Amazon Linux 2 v4.0.8 running Docker"
  version_label       = aws_elastic_beanstalk_application_version.backend_app_version.name

  setting {
    namespace = "aws:elasticbeanstalk:environment"
    name      = "ServiceRole"
    value     = "arn:aws:iam::107378568397:role/LabRole"
  }

  setting {
    namespace = "aws:autoscaling:launchconfiguration"
    name      = "IamInstanceProfile"
    value     = "LabInstanceProfile"
  }

  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "SPRING_DATASOURCE_URL"
    value     = "jdbc:postgresql://${aws_db_instance.mydb.address}:5432/mydatabase"
  }

  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "SPRING_DATASOURCE_USERNAME"
    value     = aws_db_instance.mydb.username
  }

  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "SPRING_DATASOURCE_PASSWORD"
```

```
    value     = aws_db_instance.mydb.password
  }

  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "S3_BUCKET_NAME"
    value     = aws_s3_bucket.upload_bucket.bucket
  }

  # Przekazanie konfiguracji Cognito do backendu
  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "aws.cognito.userPoolId"
    value     = aws_cognito_user_pool.chat_pool.id
  }
  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "aws.cognito.clientId"
    value     = aws_cognito_user_pool_client.chat_pool_client.id
  }
  # Ustawienia log streamingu do CloudWatch
  setting {
    namespace = "aws:elasticbeanstalk:cloudwatch:logs"
    name      = "StreamLogs"
    value     = "true"
  }
  setting {
    namespace = "aws:elasticbeanstalk:cloudwatch:logs"
    name      = "DeleteOnTerminate"
    value     = "true"
  }
  setting {
    namespace = "aws:elasticbeanstalk:cloudwatch:logs"
    name      = "RetentionInDays"
    value     = "7"
  }
  wait_for_ready_timeout = "30m"
}

##########################################
# Elastic Beanstalk — Frontend Application
##########################################
resource "aws_elastic_beanstalk_application" "frontend_app" {
  name        = "terraform-frontend-app-${random_string.suffix.result}"
  description = "Frontend for Projekt Chmury"
}

resource "aws_elastic_beanstalk_application_version" "frontend_app_version" {
  name        = "terraform-frontend-app-v1-${filemd5("../frontend/frontend-app.zip")}"
  application = aws_elastic_beanstalk_application.frontend_app.name
  bucket      = aws_s3_bucket.upload_bucket.bucket
  key         = aws_s3_object.frontend_app_zip.key
  description = "Frontend application version 1"
}


resource "aws_elastic_beanstalk_environment" "frontend_env" {
```

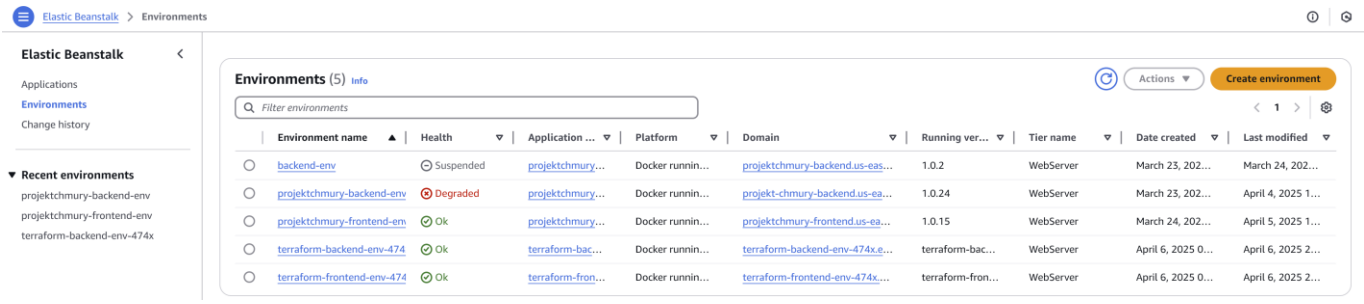
```
  name                = "terraform-frontend-env-${random_string.suffix.result}"
  application         = aws_elastic_beanstalk_application.frontend_app.name
  solution_stack_name = "64bit Amazon Linux 2 v4.0.8 running Docker"
  version_label       = aws_elastic_beanstalk_application_version.frontend_app_version.name

  setting {
    namespace = "aws:elasticbeanstalk:application:environment"
    name      = "VITE_API_URL"
    value     = "http://${aws_elastic_beanstalk_environment.backend_env.cname}/api"
  }

  # Usuwamy blok "Image" – wersja aplikacji (zip) określa już obraz
  setting {
    namespace = "aws:autoscaling:launchconfiguration"
    name      = "IamInstanceProfile"
    value     = "LabInstanceProfile"
  }
  # Ustawienia log streamingu do CloudWatch
  setting {
    namespace = "aws:elasticbeanstalk:cloudwatch:logs"
    name      = "StreamLogs"
    value     = "true"
  }
  setting {
    namespace = "aws:elasticbeanstalk:cloudwatch:logs"
    name      = "DeleteOnTerminate"
    value     = "true"
  }
  setting {
    namespace = "aws:elasticbeanstalk:cloudwatch:logs"
    name      = "RetentionInDays"
    value     = "7"
  }
  wait_for_ready_timeout = "30m"
}


##########################################
#  Lambda function for Cognito triggers
##########################################

resource "aws_lambda_function" "auto_confirm_user" {
  function_name = "auto-confirm-user"
  runtime       = "python3.9"
  handler       = "lambda_function.lambda_handler"
  role          = aws_iam_role.lambda_cognito_triggers.arn

  # Plik zip z kodem Lambdy (np. w folderze lambda/)
  filename         = "${path.module}/lambda/auto_confirm_user.zip"
  source_code_hash = filebase64sha256("${path.module}/lambda/auto_confirm_user.zip")
}

resource "aws_iam_role" "lambda_cognito_triggers" {
  name               = "lambda_cognito_triggers"
  assume_role_policy = data.aws_iam_policy_document.lambda_assume_role_policy.json
}
```

```
data "aws_iam_policy_document" "lambda_assume_role_policy" {
  statement {
    actions = ["sts:AssumeRole"]
    principals {
      type        = "Service"
      identifiers = ["lambda.amazonaws.com"]
    }
  }
}

resource "aws_iam_role_policy_attachment" "lambda_basic_execution" {
  role       = aws_iam_role.lambda_cognito_triggers.name
  policy_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
}

# Sam user pool:
resource "aws_cognito_user_pool" "chat_pool" {
  name = "terraform-projekt-chmury-user-pool-${random_string.suffix.result}"

  # Podłączamy naszą Lambdę w lambda_config
  lambda_config {
    pre_sign_up = aws_lambda_function.auto_confirm_user.arn
  }

  # Polityka haseł
  password_policy {
    minimum_length    = 6
    require_lowercase = true
    require_numbers   = true
    require_symbols   = false
    require_uppercase = true
    temporary_password_validity_days = 7
  }

  auto_verified_attributes = ["email"]
}


############################
# Outputs
############################
output "backend_url" {
  value = "http://${aws_elastic_beanstalk_environment.backend_env.cname}"
}

output "frontend_url" {
  value = "http://${aws_elastic_beanstalk_environment.frontend_env.cname}"
}

output "database_endpoint" {
  value = aws_db_instance.mydb.address
}

output "s3_bucket" {
  value = aws_s3_bucket.upload_bucket.bucket
}
```

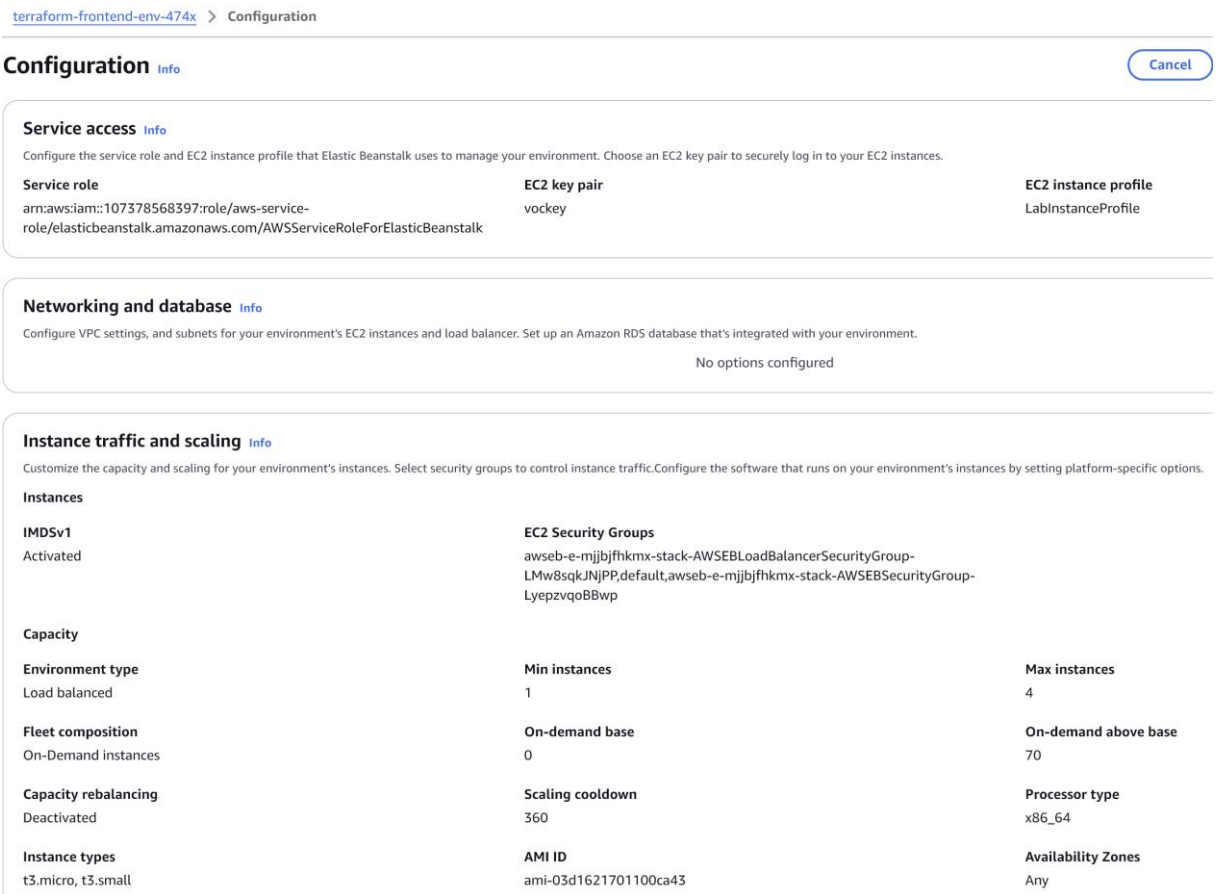
```
output "cognito_user_pool_id" {
  value = aws_cognito_user_pool.chat_pool.id
}

output "cognito_client_id" {
  value = aws_cognito_user_pool_client.chat_pool_client.id
}
```

Udało się pomyślnie postawić całą działającą infrastrukturę korzystając z Terraforma (poprzez terraform init, terraform plan, a następnie terraform apply). Oczywiście oprócz tego musieliśmy wykonać parę zmian w kodzie, w plikach takich jak .env, vite.config.js dla frontendu, czy application.properties oraz SecurityConfig dla backendu. Tak prezentuje się działająca infrastruktura na stronie AWS:

- Elastic Beanstalk (frontend, backend):



Środowisko frontendu:



Metric	Statistic	Unit
NetworkOut	Average	Bytes
Period	**Breach duration**	**Upper threshold**
5	5	6000000
Scale up increment	**Lower threshold**	**Scale down increment**
1	2000000	-1

Load balancer

Load balancer visibility	Load balancer type	Cross-zone load balancing
public	classic	Deactivated
Connection draining	**Connection draining timeout**	**Health check timeout**
Deactivated	20	5
Health check interval	**Health check unhealthy threshold**	**Health check healthy threshold**
10	5	3

Updates, monitoring, and logging Info

Edit

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

Monitoring

Log group	System	Cloudwatch custom metrics - instance
/aws/elasticbeanstalk/terraform-frontend-env-474x/environment-health.log	enhanced	—
Cloudwatch custom metrics - environment	**Log streaming**	**Retention**
—	Activated	7

Lifecycle
false

Updates

Managed updates	Deployment batch size	Deployment batch size type
Deactivated	100	Percentage
Command timeout	**Deployment policy**	**Health threshold**
600	AllAtOnce	Ok
Ignore health check	**Instance replacement**	
false	false	

Platform software

Lifecycle	Log groups	Log streaming
false	/aws/elasticbeanstalk/terraform-frontend-env-474x	Activated
Proxy server	**Logs retention**	**Rotate logs**
none	7	Deactivated

X-Ray enabled
Deactivated

Environment properties

Source	Key	Value
Plain text	VITE_API_URL	http://terraform-backend-env-474x.eba-hppnpfaf.us-east-1.elasticb…

Backend:

Configuration Info

Cancel

Service access Info

Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Service role	EC2 key pair	EC2 instance profile
arn:aws:iam::107378568397:role/LabRole	vockey	LabInstanceProfile

Networking and database Info

Configure VPC settings, and subnets for your environment's EC2 instances and load balancer. Set up an Amazon RDS database that's integrated with your environment.

No options configured

Instance traffic and scaling Info

Customize the capacity and scaling for your environment's instances. Select security groups to control instance traffic.Configure the software that runs on your environment's instances by setting platform-specific options.

Instances

IMDSv1
Activated

EC2 Security Groups
awseb-e-pwmvkiqzti-stack-AWSEBLoadBalancerSecurityGroup-c9sssT7DHpHQ,default,awseb-e-pwmvkiqzti-stack-AWSEBSecurityGroup-LNmXThRUpafy

Capacity

Environment type
Load balanced

Min instances
1

Max instances
4

Fleet composition
On-Demand instances

On-demand base
0

On-demand above base
70

Capacity rebalancing
Deactivated

Scaling cooldown
360

Processor type
x86_64

Instance types
t3.micro, t3.small

AMI ID
ami-03d1621701100ca43

Availability Zones
Any

Metric
NetworkOut

Statistic
Average

Unit
Bytes

Period
5

Breach duration
5

Upper threshold
6000000

Scale up increment
1

Lower threshold
2000000

Scale down increment
-1

Load balancer

Load balancer visibility
public

Load balancer type
classic

Cross-zone load balancing
Deactivated

Connection draining
Deactivated

Connection draining timeout
20

Health check timeout
5

Health check interval
10

Health check unhealthy threshold
5

Health check healthy threshold
3

Updates, monitoring, and logging Info

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

Monitoring

Log group
/aws/elasticbeanstalk/terraform-backend-env-474x/environment-health.log

System
enhanced

Cloudwatch custom metrics - instance
—

Cloudwatch custom metrics - environment
—

Log streaming
Activated

Retention
7

Lifecycle
false

Updates

Managed updates
Deactivated

Deployment batch size
100

Deployment batch size type
Percentage

Command timeout
600

Deployment policy
AllAtOnce

Health threshold
Ok

Ignore health check
false

Instance replacement
false

Platform software

Lifecycle
false

Log groups
/aws/elasticbeanstalk/terraform-backend-env-474x

Log streaming
Activated

Proxy server
nginx

Logs retention
7

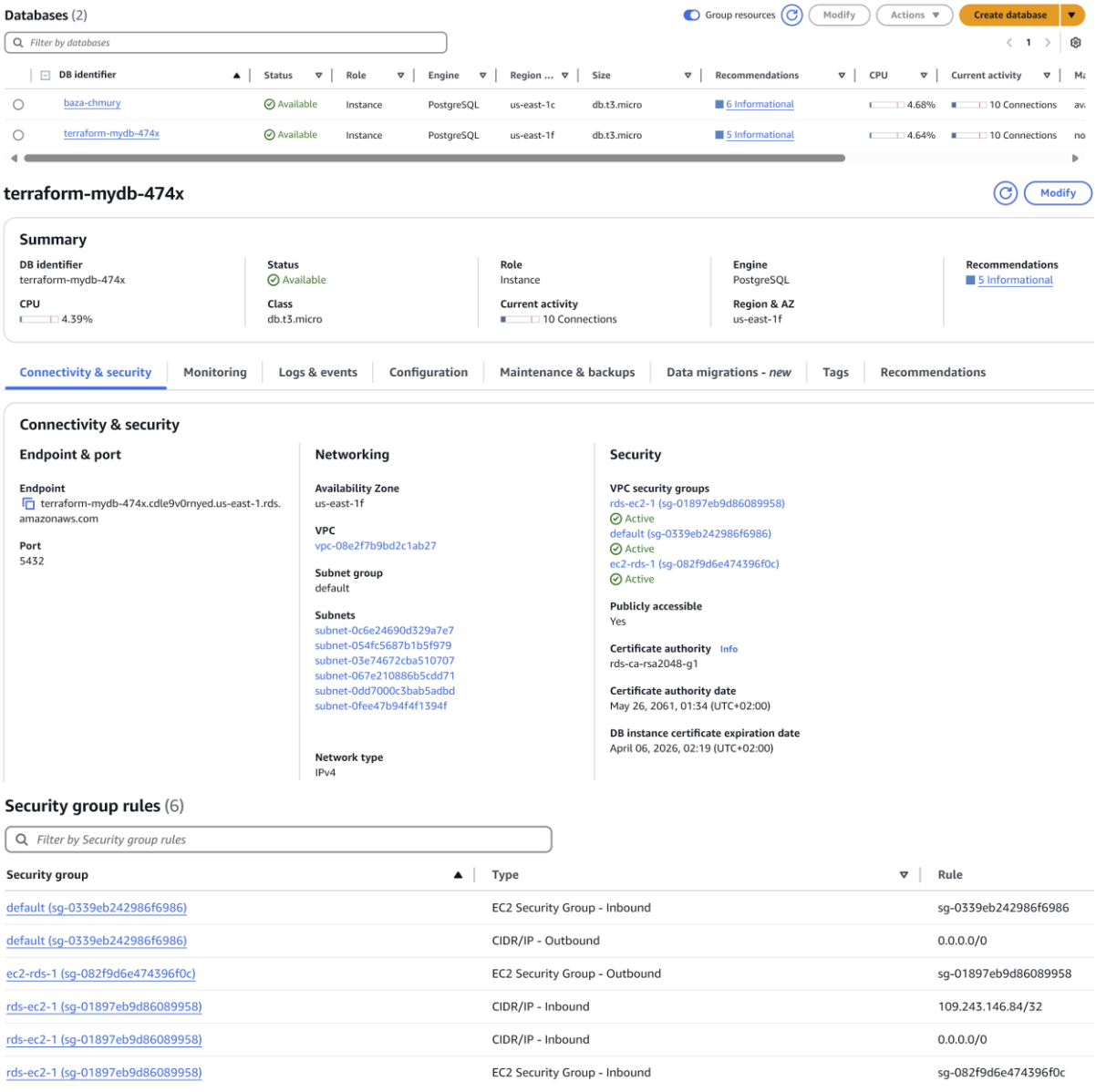
Rotate logs
Deactivated

X-Ray enabled
Deactivated

Environment properties

Source	Key	Value
Plain text	aws.cognito.clientId	3cu3kld5u5fql55630a89o94v
Plain text	aws.cognito.userPoolId	us-east-1_FkuffQRkS
Plain text	S3_BUCKET_NAME	terraform-projekt-chmury-uploads-474x
Plain text	SPRING_DATASOURCE_PASSWORD	admin1234
Plain text	SPRING_DATASOURCE_URL	jdbc:postgresql://terraform-mydb-474x.cdle9v0rnyed.us-east-1.rds.a…
Plain text	SPRING_DATASOURCE_USERNAME	postgres

- RDS:



Databases (2)

Filter by databases

Group resources | Modify | Actions | Create database

DB identifier	Status	Role	Engine	Region ...	Size	Recommendations	CPU	Current activity
baza-chmury	Available	Instance	PostgreSQL	us-east-1c	db.t3.micro	6 Informational	4.68%	10 Connections
terraform-mydb-474x	Available	Instance	PostgreSQL	us-east-1f	db.t3.micro	5 Informational	4.64%	10 Connections

terraform-mydb-474x — Modify

Summary

DB identifier: terraform-mydb-474x
Status: Available
Role: Instance
Engine: PostgreSQL
Recommendations: 5 Informational
CPU: 4.39%
Class: db.t3.micro
Current activity: 10 Connections
Region & AZ: us-east-1f

Connectivity & security | Monitoring | Logs & events | Configuration | Maintenance & backups | Data migrations - *new* | Tags | Recommendations

Connectivity & security

Endpoint & port

Endpoint: terraform-mydb-474x.cdle9v0rnyed.us-east-1.rds.amazonaws.com
Port: 5432

Networking

Availability Zone: us-east-1f
VPC: vpc-08e2f7b9bd2c1ab27
Subnet group: default
Subnets:
subnet-0c6e24690d329a7e7
subnet-054fc5687b1b5f979
subnet-03e74672cba510707
subnet-067e210886b5cdd71
subnet-0dd7000c3bab5adbd
subnet-0fee47b94f4f1394f
Network type: IPv4

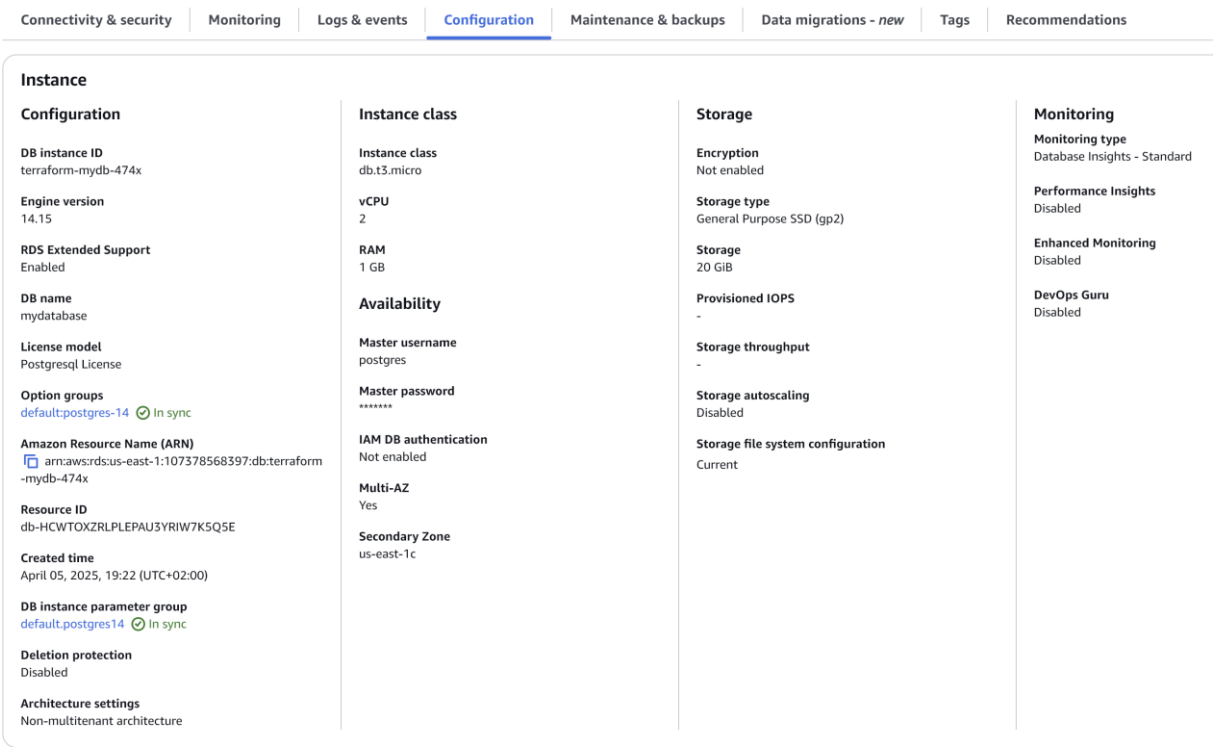
Security

VPC security groups:
rds-ec2-1 (sg-01897eb9d86089958) Active
default (sg-0339eb242986f6986) Active
ec2-rds-1 (sg-082f9d6e474396f0c) Active
Publicly accessible: Yes
Certificate authority Info: rds-ca-rsa2048-g1
Certificate authority date: May 26, 2061, 01:34 (UTC+02:00)
DB instance certificate expiration date: April 06, 2026, 02:19 (UTC+02:00)

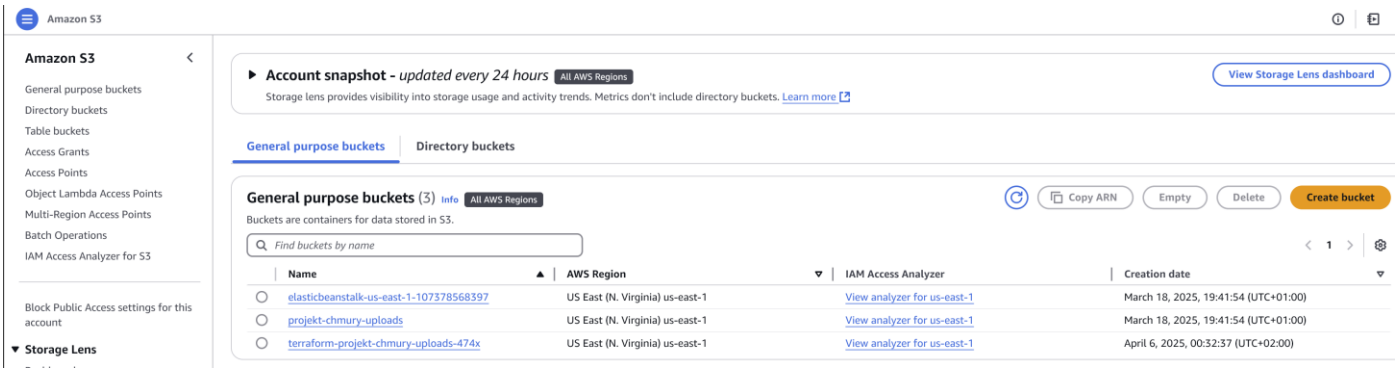
Security group rules (6)

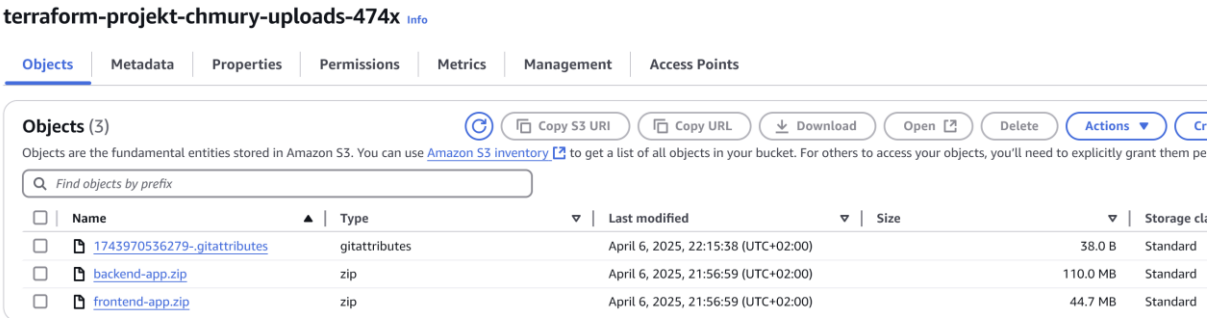
Filter by Security group rules

Security group	Type	Rule
default (sg-0339eb242986f6986)	EC2 Security Group - Inbound	sg-0339eb242986f6986
default (sg-0339eb242986f6986)	CIDR/IP - Outbound	0.0.0.0/0
ec2-rds-1 (sg-082f9d6e474396f0c)	EC2 Security Group - Outbound	sg-01897eb9d86089958
rds-ec2-1 (sg-01897eb9d86089958)	CIDR/IP - Inbound	109.243.146.84/32
rds-ec2-1 (sg-01897eb9d86089958)	CIDR/IP - Inbound	0.0.0.0/0
rds-ec2-1 (sg-01897eb9d86089958)	EC2 Security Group - Inbound	sg-082f9d6e474396f0c

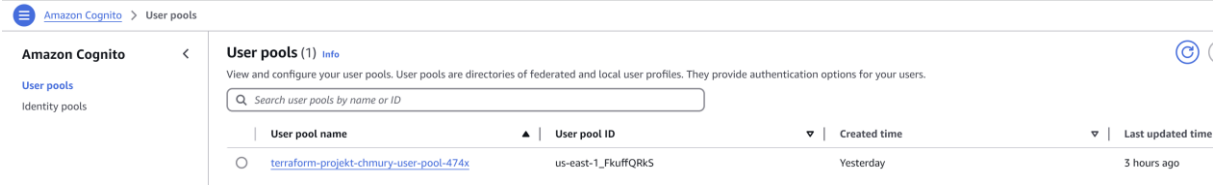


- S3:

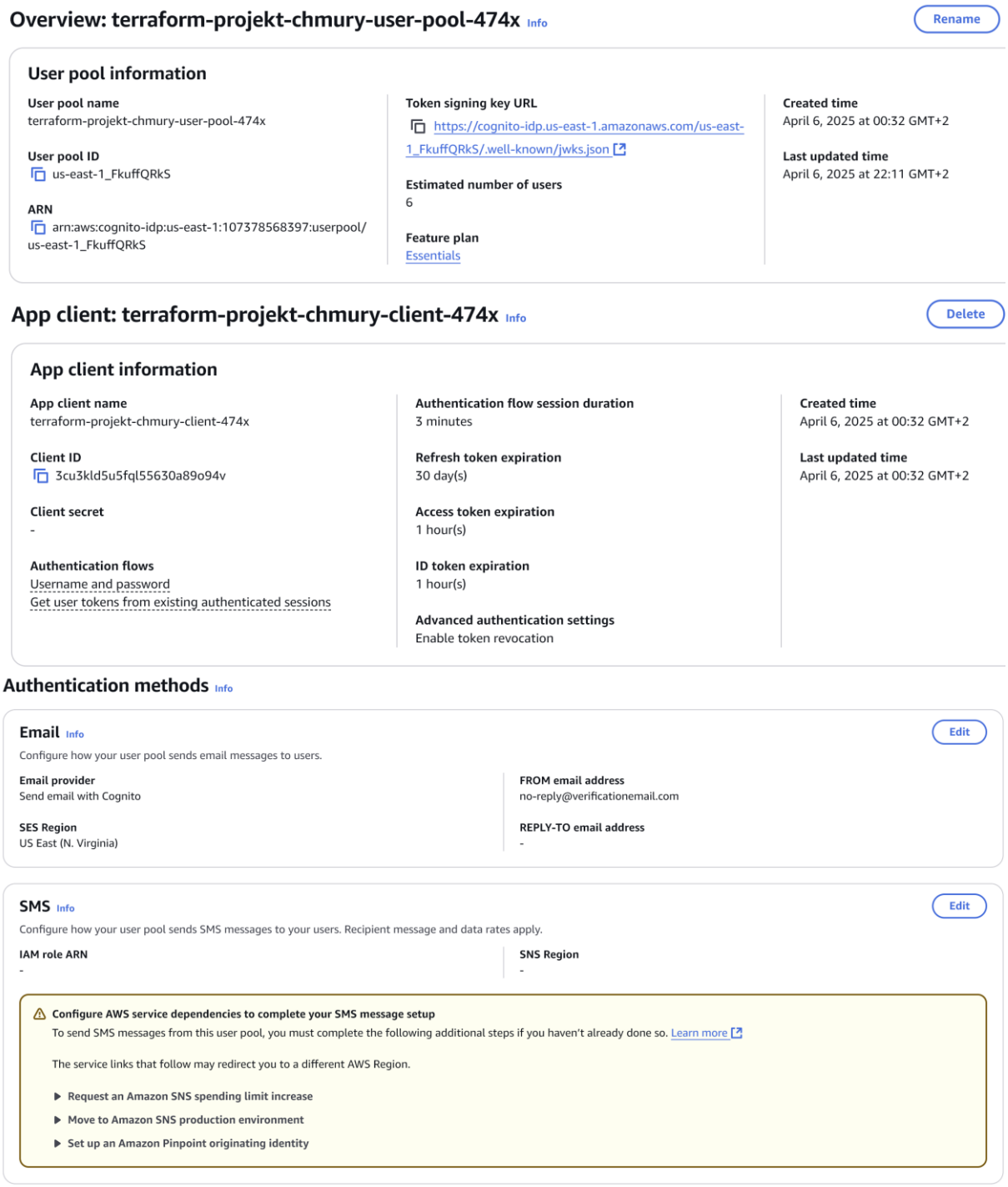




Jak widać obrazy aplikacji w postaci plików .zip prawidłowo trafiły do S3 za pomocą terraforma



- Cognito:


Overview: terraform-projekt-chmury-user-pool-474x Info
Rename
User pool information
User pool name
terraform-projekt-chmury-user-pool-474x
User pool ID
us-east-1_FkuffQRkS
ARN
arn:aws:cognito-idp:us-east-1:107378568397:userpool/us-east-1_FkuffQRkS
Token signing key URL
https://cognito-idp.us-east-1.amazonaws.com/us-east-1_FkuffQRkS/.well-known/jwks.json
Estimated number of users
6
Feature plan
Essentials
Created time
April 6, 2025 at 00:32 GMT+2
Last updated time
April 6, 2025 at 22:11 GMT+2
App client: terraform-projekt-chmury-client-474x Info
Delete
App client information
App client name
terraform-projekt-chmury-client-474x
Client ID
3cu3kld5u5fql55630a89o94v
Client secret
-
Authentication flows
Username and password
Get user tokens from existing authenticated sessions
Authentication flow session duration
3 minutes
Refresh token expiration
30 day(s)
Access token expiration
1 hour(s)
ID token expiration
1 hour(s)
Advanced authentication settings
Enable token revocation
Created time
April 6, 2025 at 00:32 GMT+2
Last updated time
April 6, 2025 at 00:32 GMT+2
Authentication methods Info
Email Info
Edit
Configure how your user pool sends email messages to users.
Email provider
Send email with Cognito
SES Region
US East (N. Virginia)
FROM email address
no-reply@verificationemail.com
REPLY-TO email address
-
SMS Info
Edit
Configure how your user pool sends SMS messages to your users. Recipient message and data rates apply.
IAM role ARN
-
SNS Region
-
Configure AWS service dependencies to complete your SMS message setup
To send SMS messages from this user pool, you must complete the following additional steps if you haven't already done so. Learn more
The service links that follow may redirect you to a different AWS Region.
Request an Amazon SNS spending limit increase
Move to Amazon SNS production environment
Set up an Amazon Pinpoint originating identity

Password policy Info

Edit

Create a password policy to define the length and complexity of the passwords your users can set.

Password minimum length
6 character(s)

Password requirements
Contains at least 1 lowercase letter

Temporary passwords set by administrators expire in
7 day(s)

Allow reuse of previous passwords

Passkey Info

Edit

Configure sign-in with biometrics, hardware keys, and authenticator apps.

User verification
Preferred

Domain for relying party ID
Third-party domain

Third-party domain
-

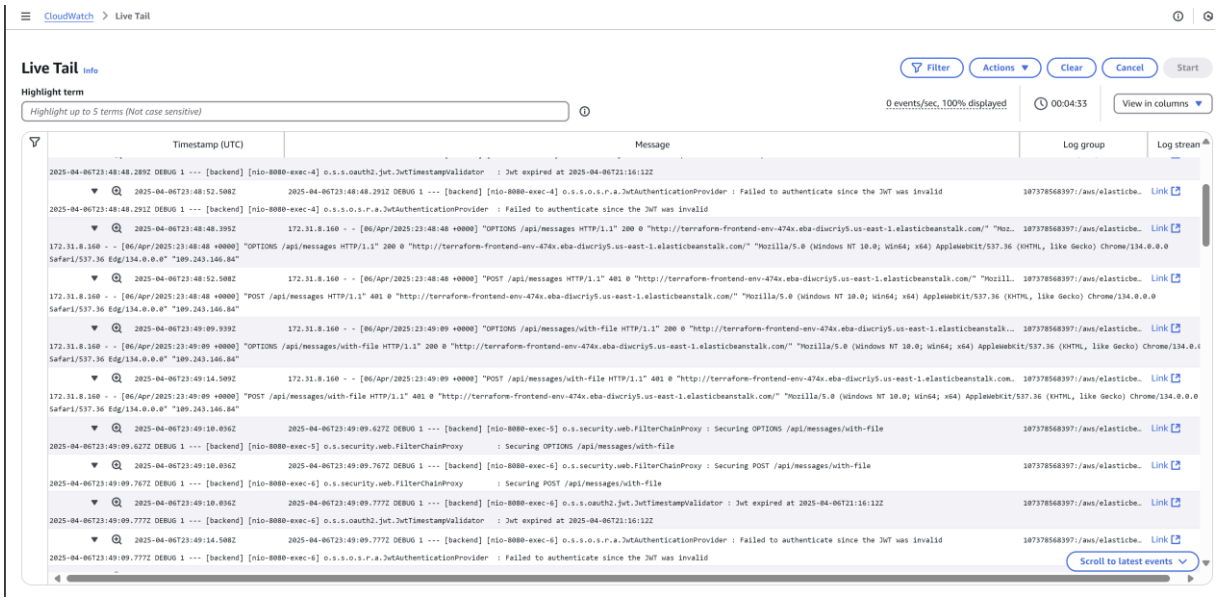
Extensions Info

Lambda triggers (1) Info

Add and configure Lambda triggers for your user pool. Cognito can invoke Lambda functions in your a register and confirm users, authenticate users, send messages, and generate tokens.

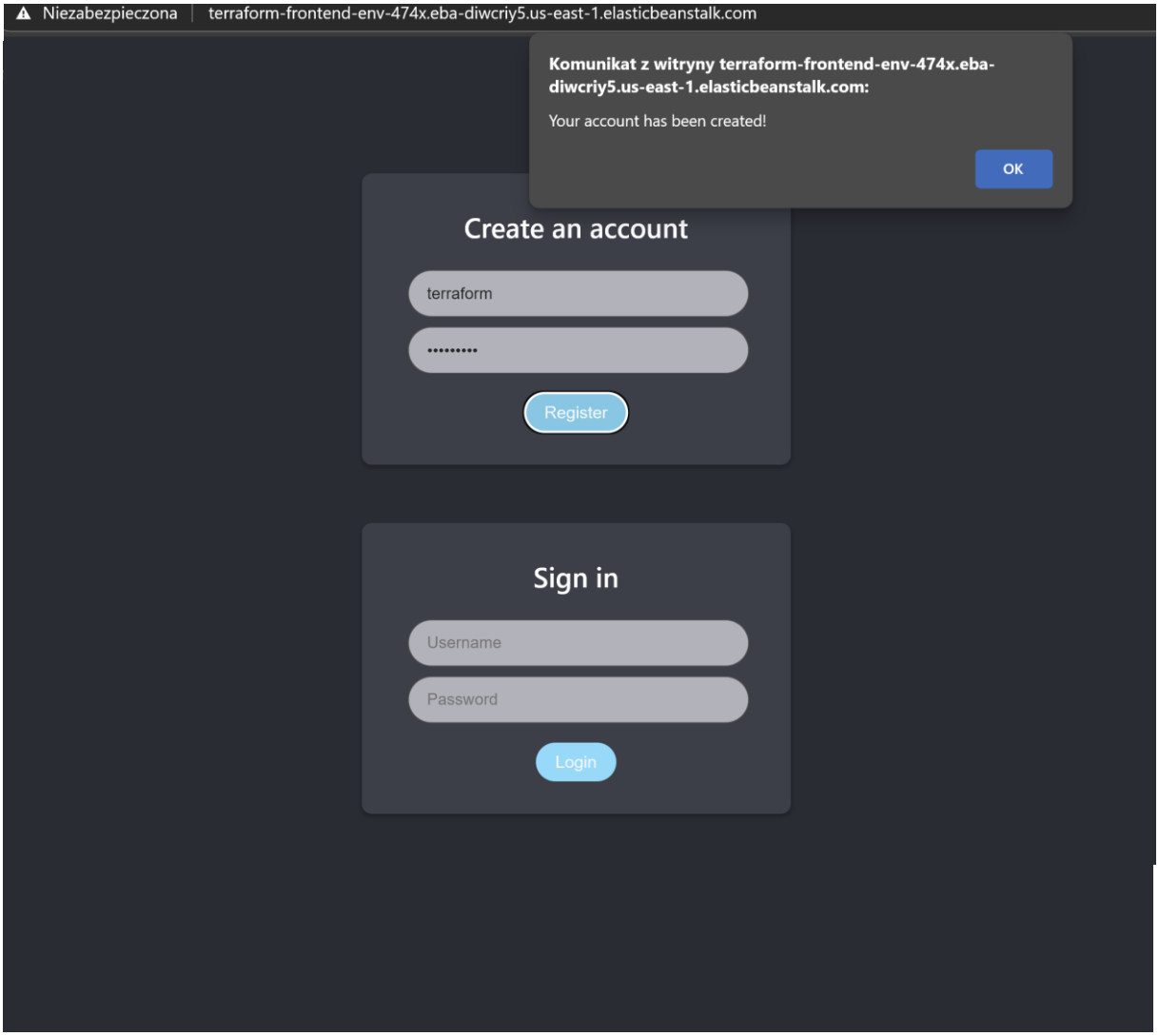
Lambda triggers	Attached Lambda function
Pre sign-up Lambda trigger	auto-confirm-user

- Cloudwatch:

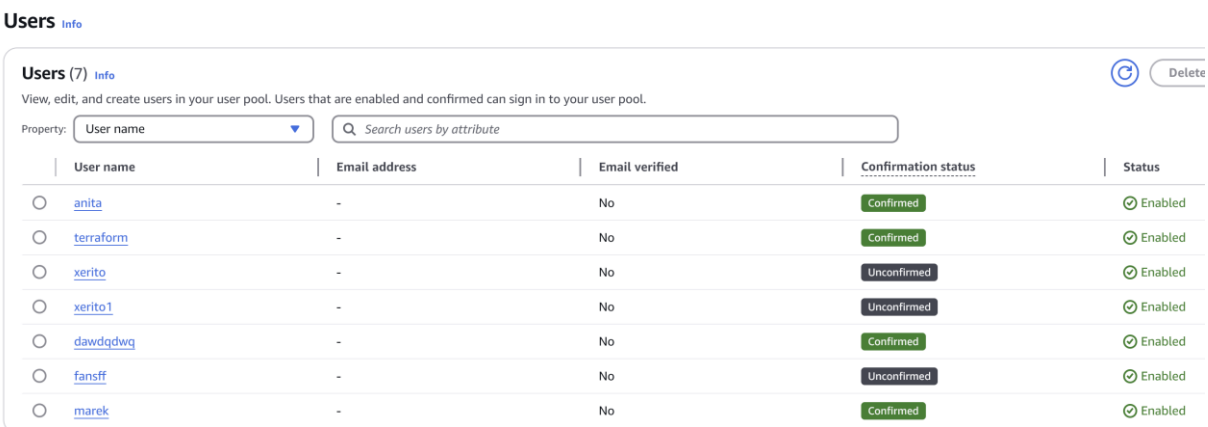


No to teraz sprawdźmy działanie

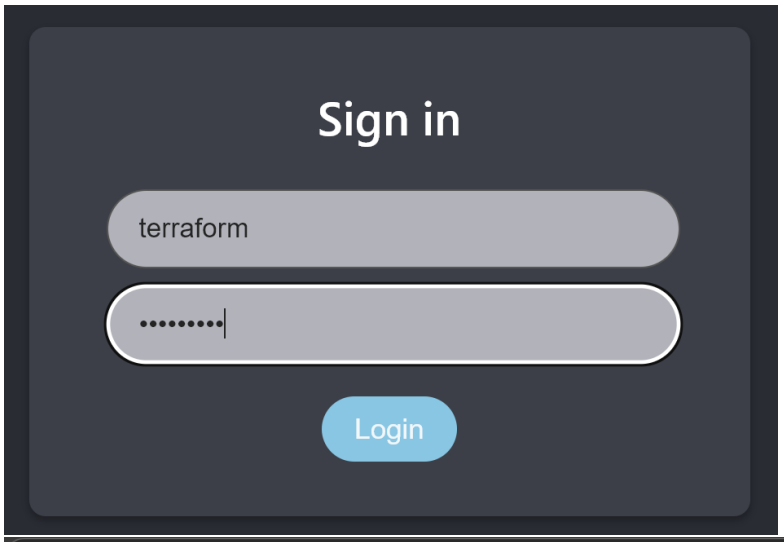
Spróbujmy stworzyć użytkownika terraform:

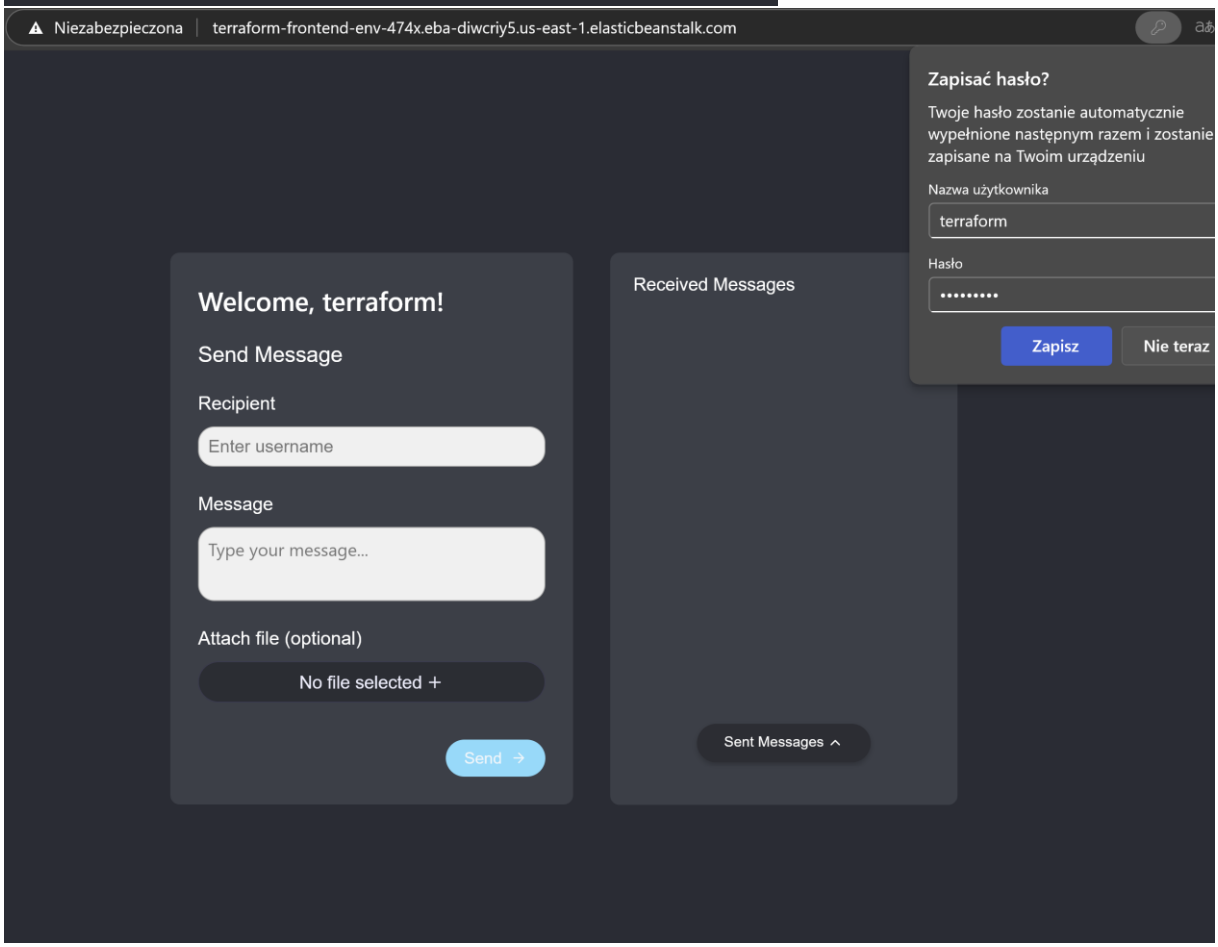


W Cognito:

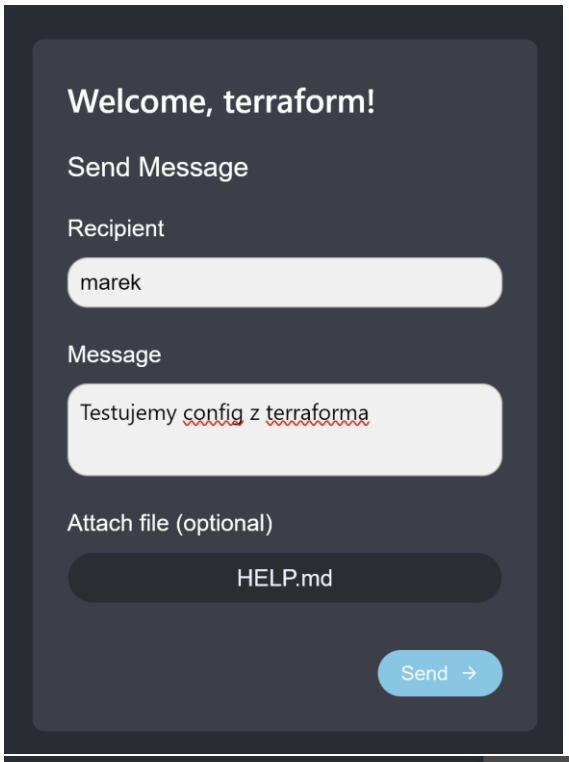


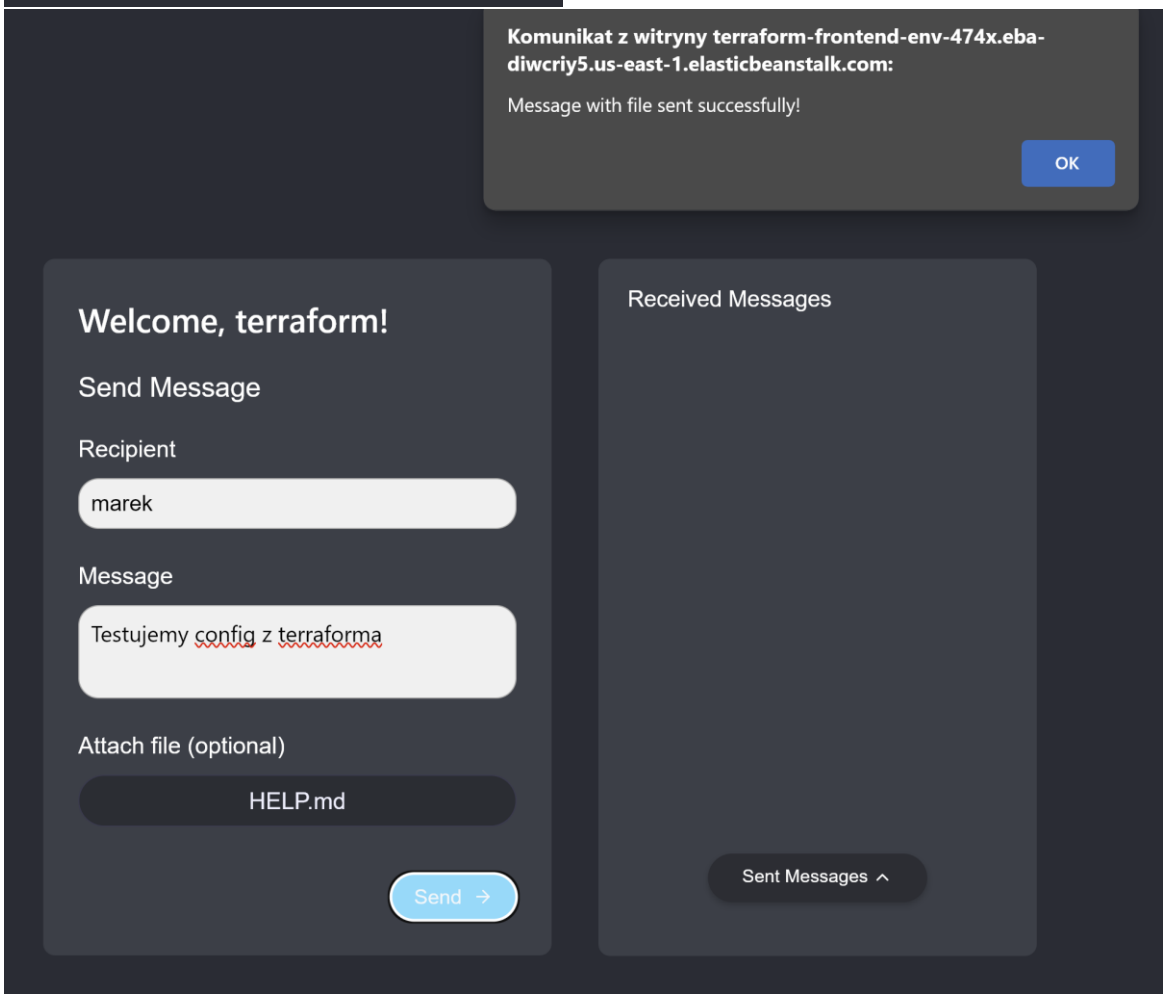
Spróbujmy się zalogować:

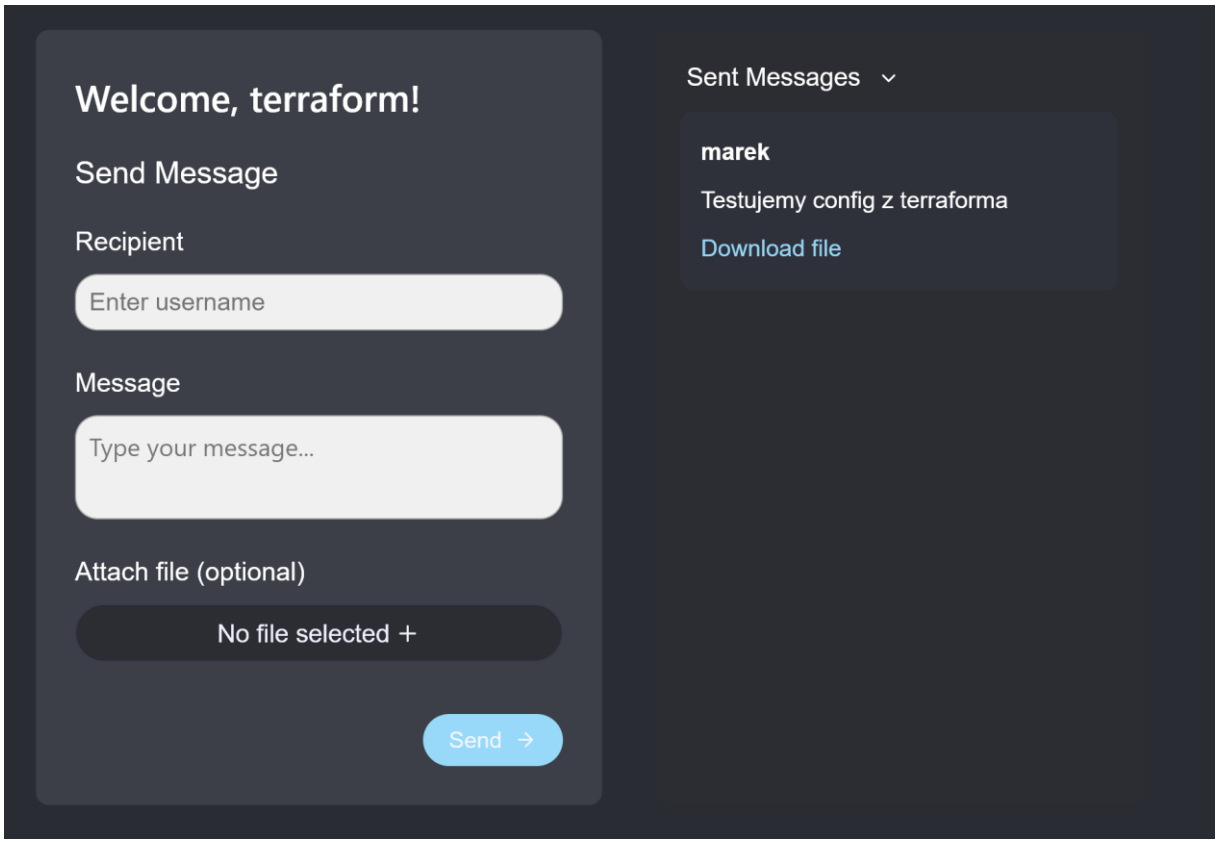




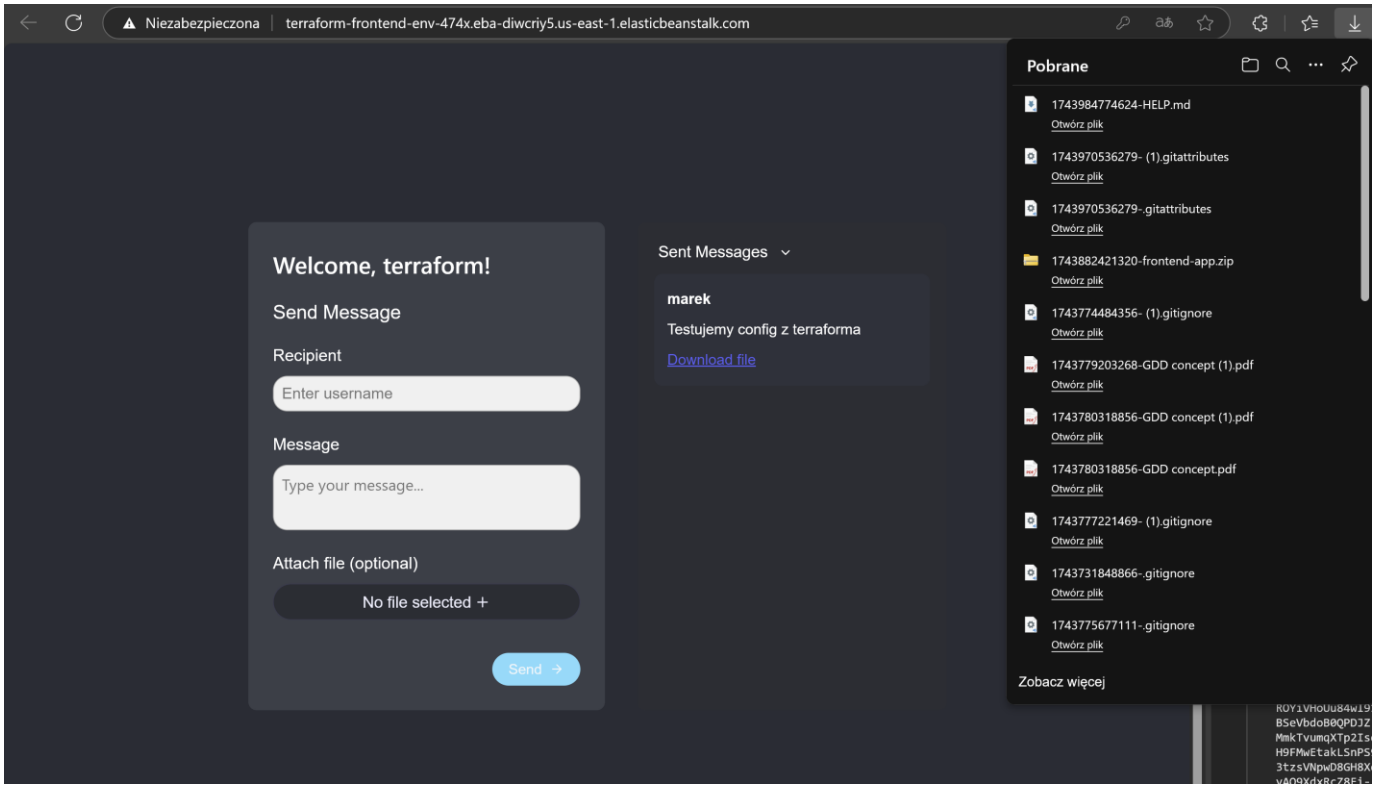
Zalogowaliśmy się z powodzeniem, to teraz wyślijmy jakąś wiadomość z załącznikiem do jakiegoś innego użytkownika:



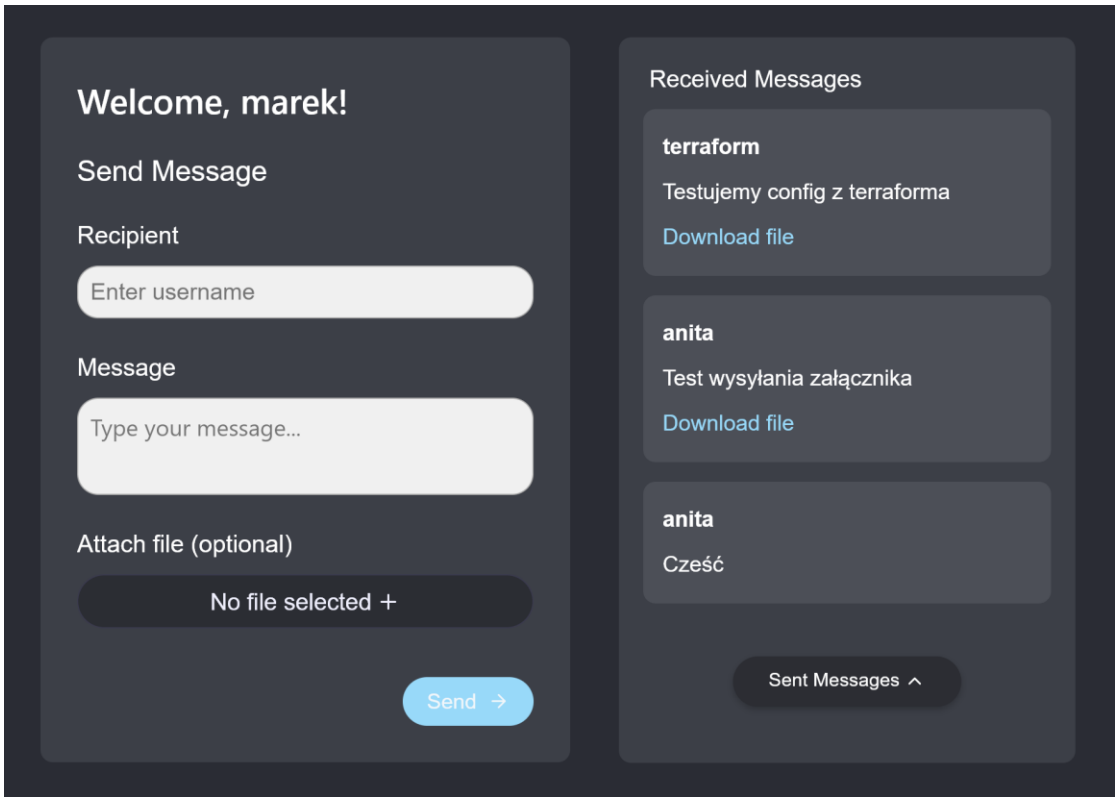




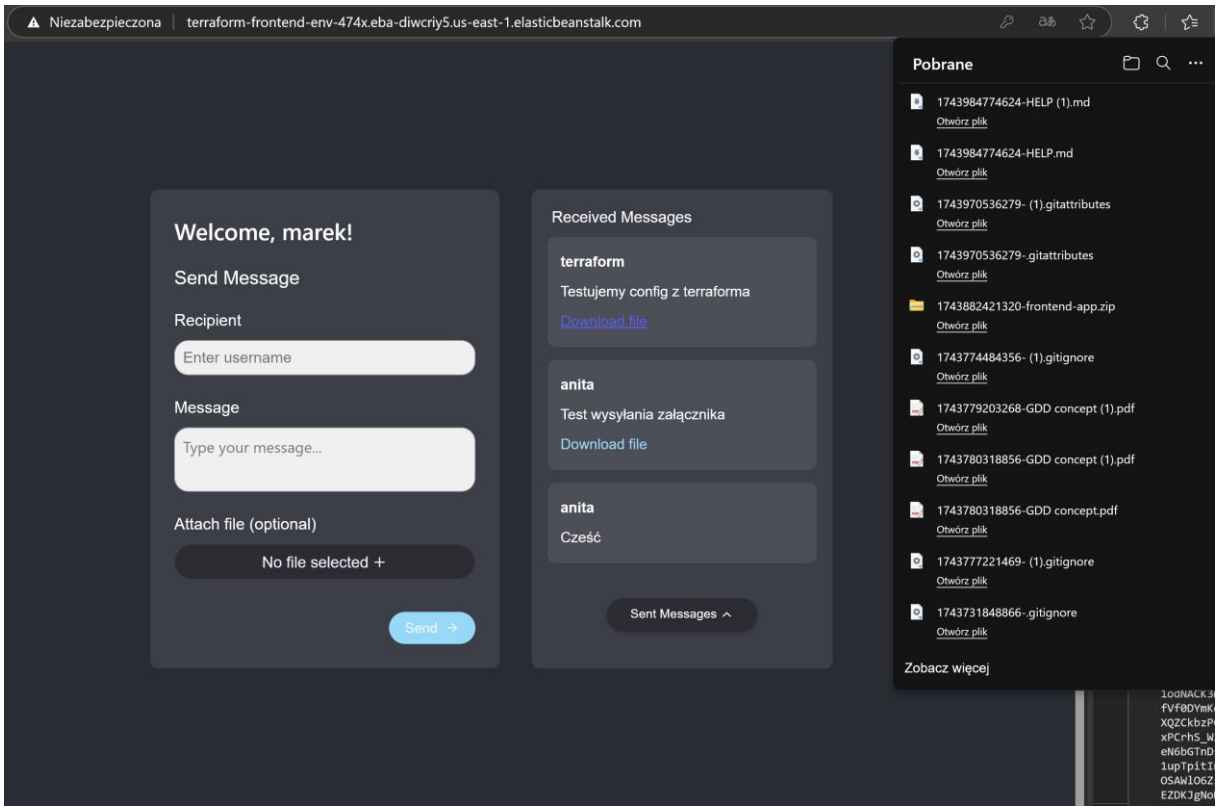
No i rzeczywiście w wysłanych mamy tą wiadomość, spróbujmy pobrać plik u siebie:



Plik się pobiera, wszystko działa. Sprawdźmy teraz jak to wygląda u marka:



Mamy wiadomość od terraforma, to sprawdźmy jeszcze pobieranie pliku



Wszystko działa, spełniłem wszystkie założenia projektowe

Jeśli natomiast chodzi o Endpointy to mamy ich 10: 5 POST i 5 GET

Endpoint POST 1:

```java
// Endpoint tworzenia wiadomości
@PostMapping  stvshy
public Message addMessage(@RequestBody Map<String, String> body) {
    String authorUsername = body.get("author");
    String content = body.get("content");
    String recipientUsername = body.get("recipient");

    logger.debug("Próba zapisu wiadomości. Nadawca: {}, treść: {}, odbiorca: {}",
            authorUsername, content, recipientUsername);

    Message msg = new Message(authorUsername, content);
    msg.setRecipientUsername(recipientUsername);
    return messageRepository.save(msg);
}
```

Endpoint GET 1:

```java
// Endpoint: wiadomości odebrane przez danego użytkownika
@GetMapping("/received")  stvshy
public List<Message> getReceivedMessages(@RequestParam String username) {
    logger.debug("getReceivedMessages called with username: {}", username);
    return messageRepository.findByRecipientUsername(username);
}
```

Endpoint GET 2:

```java
// Endpoint: wiadomości wysłane przez danego użytkownika
@GetMapping("/sent")  stvshy
public List<Message> getSentMessages(@RequestParam String username) {
    logger.debug("getSentMessages called with username: {}", username);
    return messageRepository.findByAuthorUsername(username);
}
```

Endpoint POST 2:

```java
// Endpoint do uploadu pliku pozostaje bez zmian
@PostMapping(value = "/upload", consumes = MediaType.MULTIPART_FORM_DATA_VALUE)  stv
public String uploadFile(@RequestParam("file") MultipartFile file) throws IOException {
    File uploadsDir = new File( pathname: "uploads");
    if (!uploadsDir.exists()) {
        uploadsDir.mkdir();
    }
    File destination = new File(uploadsDir, file.getOriginalFilename());
    file.transferTo(destination);
    return "Plik zapisany: " + destination.getAbsolutePath();
}
```

Endpoint POST 3:

```java
@PostMapping(value = "/with-file", consumes = MediaType.MULTIPART_FORM_DATA_VALUE)  stvshy
public Message addMessageWithFile(
        @RequestParam("author") String author,
        @RequestParam("content") String content,
        @RequestParam(value = "recipient", required = false) String recipient,
        @RequestParam("file") MultipartFile file
) throws IOException {
    // Wyślij plik do S3 i pobierz klucz (fileName)
    String s3Key = s3Service.uploadFile(file);
    Message msg = new Message(author, content);
    msg.setRecipientUsername(recipient);
    // Zapisz klucz pliku, nie pełny URL
    msg.setFile(s3Key);
    return messageRepository.save(msg);
}
```

Endpoint POST 4:

```java
@PostMapping("/login")  stvshy
public ResponseEntity<?> login(@RequestBody User loginData) {
    try {
        InitiateAuthResponse response = cognitoService.userLogin(loginData.getUsername(), loginData.getPassword());
        Map<String, String> tokens = new HashMap<>();
        tokens.put("idToken", response.authenticationResult().idToken());
        tokens.put("accessToken", response.authenticationResult().accessToken());
        tokens.put("refreshToken", response.authenticationResult().refreshToken());
        tokens.put("tokenType", response.authenticationResult().tokenType());
        return ResponseEntity.ok(tokens);
    } catch (Exception e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                .body("Błąd logowania: " + e.getMessage());
    }
}
```

Endpoint POST 5:

```java
@PostMapping("/register")  stvshy
public ResponseEntity<?> register(@RequestBody User user) {
    try {
        // Rejestrujemy użytkownika w Cognito
        SignUpResponse response = cognitoService.signUp(user.getUsername(), user.getPassword());
        return ResponseEntity.ok( body: "Rejestracja pomyślna. UserSub: " + response.userSub());
    } catch (Exception e) {
        // Wyświetlamy pełen stack trace w logach
        e.printStackTrace();

        return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body("Błąd rejestracji: " + e.getMessage() + "\nStackTrace: " + Arrays.toString(e.getStackTrace()));
    }
}
```

Endpoint GET 3:

```java
@GetMapping("/api/users")  stvshy
public List<String> getAllUsernames() {
    return userRepository.findAll().stream()  Stream<User>
            .map(User::getUsername)  Stream<String>
            .collect(Collectors.toList());
}
```

Endpoint GET 4:

```java
@GetMapping("/test-db")  stvshy
public String testDb() {
    try (Connection conn = DriverManager.getConnection(
            Objects.requireNonNull(environment.getProperty("spring.datasource.url")),
            environment.getProperty("spring.datasource.username"),
            environment.getProperty("spring.datasource.password")
    )) {
        return "Connected to DB: " + conn.getMetaData().getURL();
    } catch(Exception e) {
        return "Error: " + e.getMessage();
    }
}
```

Endpoint GET 5:

```java
// Endpoint do pobierania pliku na podstawie ID wiadomości
@GetMapping("/download/{id}")  stvshy
public ResponseEntity<?> downloadFile(@PathVariable Long id) {
    Message msg = messageRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("Message not found"));

    if (msg.getFile() == null) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
                .body("Brak pliku w wiadomości.");
    }

    // msg.getFile() zawiera teraz klucz pliku, np. "1672531234567-nazwa_pliku.jpg"
    String presignedUrl = s3Service.generatePresignedUrl(msg.getFile());

    return ResponseEntity.status(HttpStatus.FOUND)
            .location(URI.create(presignedUrl))
            .build();
}
```