Personal
Development Portfolio

Fontys

HOGESCHOOL ICT

Javier Duran

Student Number: 3567885

Infrastructure Engineering

Table of Contents

Introduction

The Personal Development Portfolio (PDP) are all the activities that have helped me gain or improve certain skills during a certain project or time lapse. Through this PDP descriptions of assignments, skills learned and knowledge that assist with my personal and professional development.

My name is Javier Duran, I am an international student at Fontys University of Applied Sciences, in the Infrastructure Engineering BSc program. Very interested and motivated to keep learning about IT in general but more specifically cybersecurity. As risks keep in this area keep growing, I think it is very important to have a secure infrastructure to give companies the edge advantage. In addition, I want to be part of people who will help build a smarter and better future through technology.

Purpose

The purpose of this assignment is to create Development Portfolio through which skills, techniques, knowledge, and topics learned during lectures will be laid out. In addition, the case studies and multidisciplinary project will help us develop these knowledge and skills learned by putting them into practice. Most importantly, my personal development in professional skills such as teamwork, communication skills, and leadership will be evaluated. Different forms of evidence will support all the above-mentioned characteristics. Critical reflection will be done on the work delivered. This facilitates the progress and gives an insight into my learning process.

Objectives

- Learn new technical and professional skills.
- Getting an insight into my learning process.
- Inform on the skills and knowledge learned during the semester.
- Reflecting on the assignments and case studies.
- Improve through self-evaluating what are my weak and strong areas and/or skills.

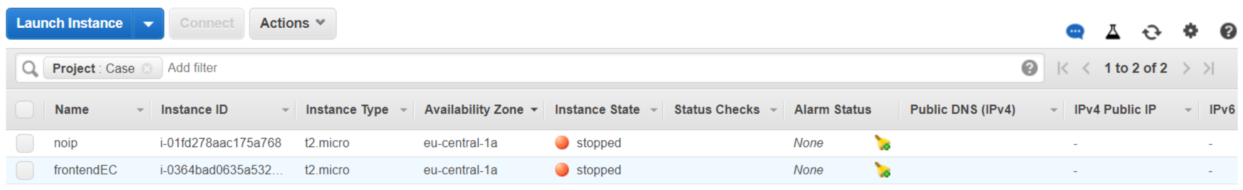
1.0 Orchestration

1.1 EC2 Instances

An Amazon EC2 instance is a virtual server in Amazon's Elastic Compute Cloud for running applications on AWS infrastructure. This can serve as basically unlimited amounts of VMs that will run different programs and tools to keep the infrastructure working as efficiently, secure, and error resilient. These instances are created from Amazon Machine Images (AMI). The different AMIs have different operating systems, CPU power and other components that will depend on the type of instance and AMI used to launch the instance.

In this semester we will make use of EC2 instances to deploy an application and other different monitoring tools, services and security features that will create the infrastructure for our case study and multidisciplinary project. For example, we will use an Ubuntu AMI in an EC2 instance to launch our webserver and application.

1.1.1 Evidence



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6
noip	i-01fd278aac175a768	t2.micro	eu-central-1a	● stopped		None		-	-
frontendEC	i-0364bad0635a532…	t2.micro	eu-central-1a	● stopped		None		-	-

Figure 1.1.1

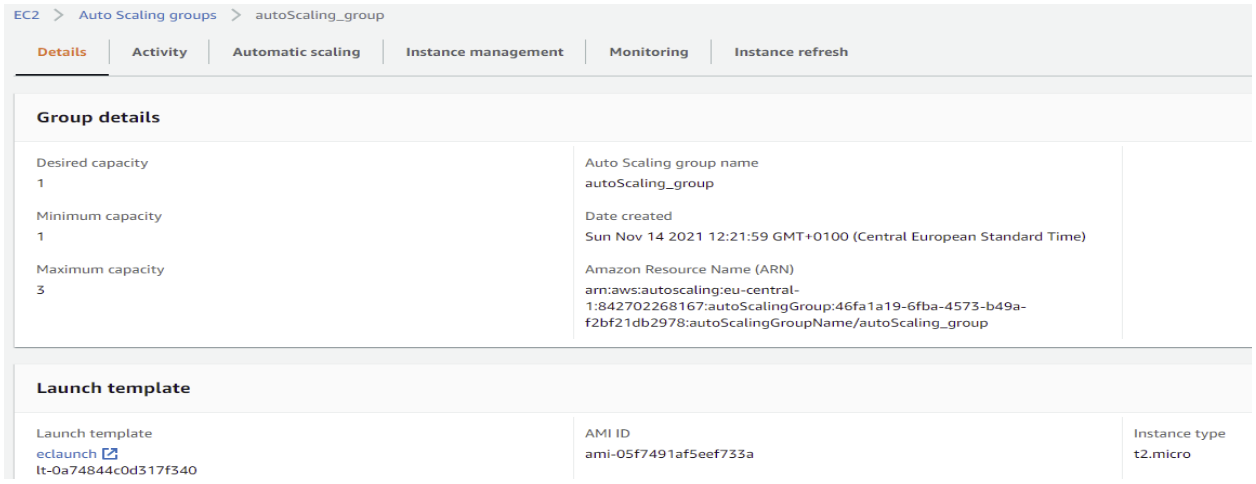
- In figure 1.1.1, we can observe some of the instances the group is going to use for the case study. These instances are allocated in the Public Subnet of our VPC to host the front end of our application.

1.2 Auto Scaling

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. It is possible to choose a maximum, minimum and a desired number of instances in the auto scaling group. It is also possible to attach policies to be able to adjust the autoscaling group accordingly.

We will make use of autoscaling group in our webserver's instances. The autoscaling group will use a template used to create am Ubuntu EC2 instance with Nginx installed. This will auto scale webserver instances in two different availability zones, with a desired capacity of 1 webserver per availability zone, a maximum of 3 per availability zone and the minimum is one per availability zone. CloudWatch alarms will trigger the autoscaling policies depending on the CPU and RAM utilization.

1.2.1 Evidence



Group details

Desired capacity
1

Minimum capacity
1

Maximum capacity
3

Auto Scaling group name
autoScaling_group

Date created
Sun Nov 14 2021 12:21:59 GMT+0100 (Central European Standard Time)

Amazon Resource Name (ARN)
arn:aws:autoscaling:eu-central-1:842702268167:autoScalingGroup:46fa1a19-6fba-4573-b49a-f2bf21db2978:autoScalingGroupName/autoScaling_group

Launch template

Launch template
eclaunch
lt-0a74844c0d317f340

AMI ID
ami-05f7491af5eef733a

Instance type
t2.micro

Figure 1.2.1

- In Figure 1.2.1 it is possible to observe the auto scaling group created. At the bottom left we can see that it uses 'eclaunch' template. This template is created based on the webserver launched since it is auto scaling our webservers. This architecture is possible to observe in Figure 2.1.1, where both webservers in place are part of the autoscaling group.

1.3 AWS Lambda

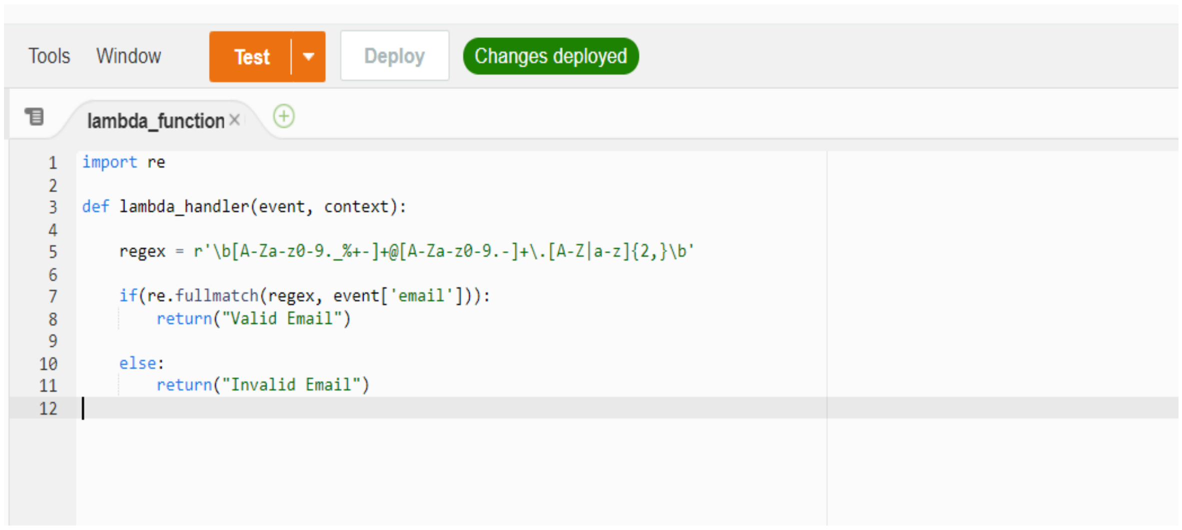
Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging (AWS, 2021). AWS Lambda has the following key features:

- Concurrency and Scaling

- Code signing
- Lambda Extensions
- Functions defined as container images
- Functions blueprints
- Database & File System Access

For the course assignments, the group came up with a Lambda solution to check for the validity of an email. This means that the string input by the user for “email” has to have the correct characters, such as @ and period. If the email string satisfies the “Regex” policies than the email will be taken as valid. In the other hand, if the regex policies are not satisfied then the email will not be accepted as valid.

1.3.1 Evidence



```python
import re

def lambda_handler(event, context):

    regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

    if(re.fullmatch(regex, event['email'])):
        return("Valid Email")

    else:
        return("Invalid Email")
```

Figure 1.3.1

- Figure 1.3.1 shows the Lambda function that was created to check for the validity of an email. The validity of the email will depend on the regex policies stablished in the function.

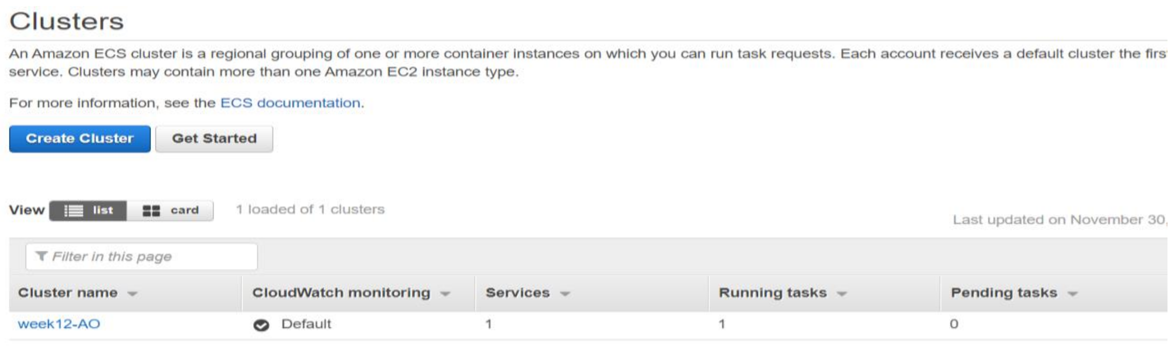
1.4 AWS Elastic Containers Service

Amazon ECS is a fully managed container orchestration service that makes it easy for you to deploy, manage, and scale containerized applications (AWS, 2021). Some advantages of using ECS are:

- You can build and deploy container-based applications on-premises or in the cloud with Amazon ECS.
- Plan, schedule, and execute batch computing workloads across the full range of AWS services.
- Most importantly, and why we will make use of this tool is to automatically scale and run web applications in multiple Availability Zones with the performance, scale, reliability, and availability of AWS.

The team will make use of AWS Elastic Containers Service alongside Dockers and CI/CD Pipelines to build, test, deploy our sample application for the case study. The main use of ECS is to deploy the front-end of our website and the failover redirection web server. AWS ECS will also allow us to have continuous integration in the application. It will make it easier for the team and IT admins in general to manage and maintain the infrastructure and application.

1.4.1 Evidence



Clusters

An Amazon ECS cluster is a regional grouping of one or more container instances on which you can run task requests. Each account receives a default cluster the firs service. Clusters may contain more than one Amazon EC2 instance type.

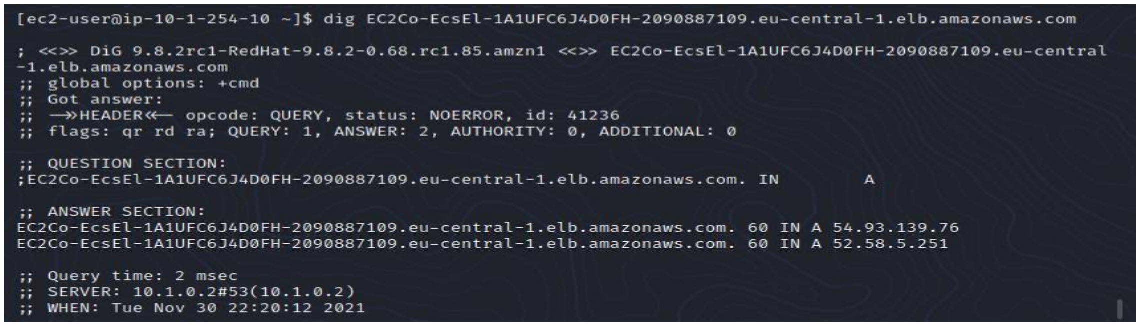
For more information, see the ECS documentation.

1 loaded of 1 clusters

Last updated on November 30,

Cluster name	CloudWatch monitoring	Services	Running tasks	Pending tasks
week12-AO	✓ Default	1	1	0

Figure 1.4.1



```
[ec2-user@ip-10-1-254-10 ~]$ dig EC2Co-EcsEl-1A1UFC6J4D0FH-2090887109.eu-central-1.elb.amazonaws.com

; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.68.rc1.85.amzn1 <<>> EC2Co-EcsEl-1A1UFC6J4D0FH-2090887109.eu-central
-1.elb.amazonaws.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41236
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;EC2Co-EcsEl-1A1UFC6J4D0FH-2090887109.eu-central-1.elb.amazonaws.com. IN        A

;; ANSWER SECTION:
EC2Co-EcsEl-1A1UFC6J4D0FH-2090887109.eu-central-1.elb.amazonaws.com. 60 IN A 54.93.139.76
EC2Co-EcsEl-1A1UFC6J4D0FH-2090887109.eu-central-1.elb.amazonaws.com. 60 IN A 52.58.5.251

;; Query time: 2 msec
;; SERVER: 10.1.0.2#53(10.1.0.2)
;; WHEN: Tue Nov 30 22:20:12 2021
```

Figure 1.4.2

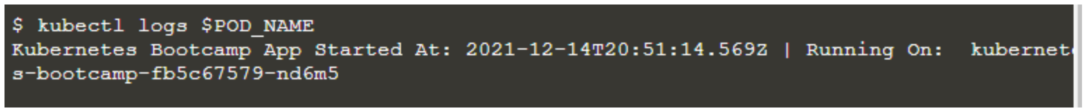
- In Figure 1.4.1 you can see the creation of an ECS Cluster created to complete the assignment. To have a proper functionality, a load balancer was also created to route traffic through the load balancer. In Figure 1.4.2 a Dig command was executed on the load balancer DNS with a successful response.

1.5 Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more. The core of Kubernetes' control plane is the API server. The API server exposes an HTTP API that lets end users, different parts of your cluster, and external components communicate with one another (Kubernetes, 2021). Kubernetes is a versatile tool for automating and simplifying your container workflow.

The team did not make use of Kubernetes in the project due to the time it takes to make Kubernetes work properly and take full advantage of the pros that Kubernetes offers. For the assignments we deployed an application using Kubernetes in a Kubernetes node. We will run the image of the application in a pod. Later, we need to expose the application to the internet so it can be accessible for the users. Once the port of the cluster was found, we opened the port of the machine to test our results. Kubernetes is a useful but complex tool to learn. This will help us scale the applications easier and will speed up some process of continuous deployment and integration.

1.5.1 Evidence



```
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2021-12-14T20:51:14.569Z | Running On:  kubernete
s-bootcamp-fb5c67579-nd6m5
```

Figure 1.5.1

- In Figure 1.5.1 we can observe some information of the pod running the application, *"kubectl logs $POD_NAME"* was the command used for the output seen.

1.6 Elastic Kubernetes Service

Amazon Elastic Kubernetes Service is a managed container service to run and scale Kubernetes applications in the cloud or on-premises. It is possible to create applications that automatically scale and run across multiple Availability Zones. This service will allow IT admins to continuously deploy and integrate Kubernetes applications. makes working with existing container infrastructure or migrations easier. It gives you customizable container or node management with more control. You can manage the applications running on your cluster or adjust individual resources using the Kubernetes Dashboard. Kubernetes can be challenging to learn and make use of but, Amazon EKS simplifies the process of building architecture in AWS.

It is possible to EKS within our project, due to time constrains we decided not to because we already had a working way to deploy our applications. In the other hand, if EKS was learned earlier in the semester or some background experience was had, we could have implemented it and have a more efficient way of deploying our applications in Kubernetes and having continuous integration, deployment and orchestration within the “dockerized” services and applications.

2.0 Network Orchestration

2.1 VPC

Amazon VPC enables you to launch Amazon Web Services resources into a virtual network you've defined. This virtual network resembles a traditional network that would operate in a data center, with the benefits of using the scalable infrastructure of AWS (AWS, 2021). VPC stands for Virtual Private Cloud and enables us to build a complete infrastructure, securely, in the cloud. This allows us to add services and components in the infrastructure and configure them for the clients’ benefits.

The team will create two VPCs for the case study, one which will be our main infrastructure where the application will be hosted and the services will be running, while in the second VPC we will keep the database so it will be secure and available. In figure 3.2.1 it is possible to observe that a VPC was built via Terraform with the main components such as internet gateways, routing tables and a private and public subnet.

2.1.1 Evidence

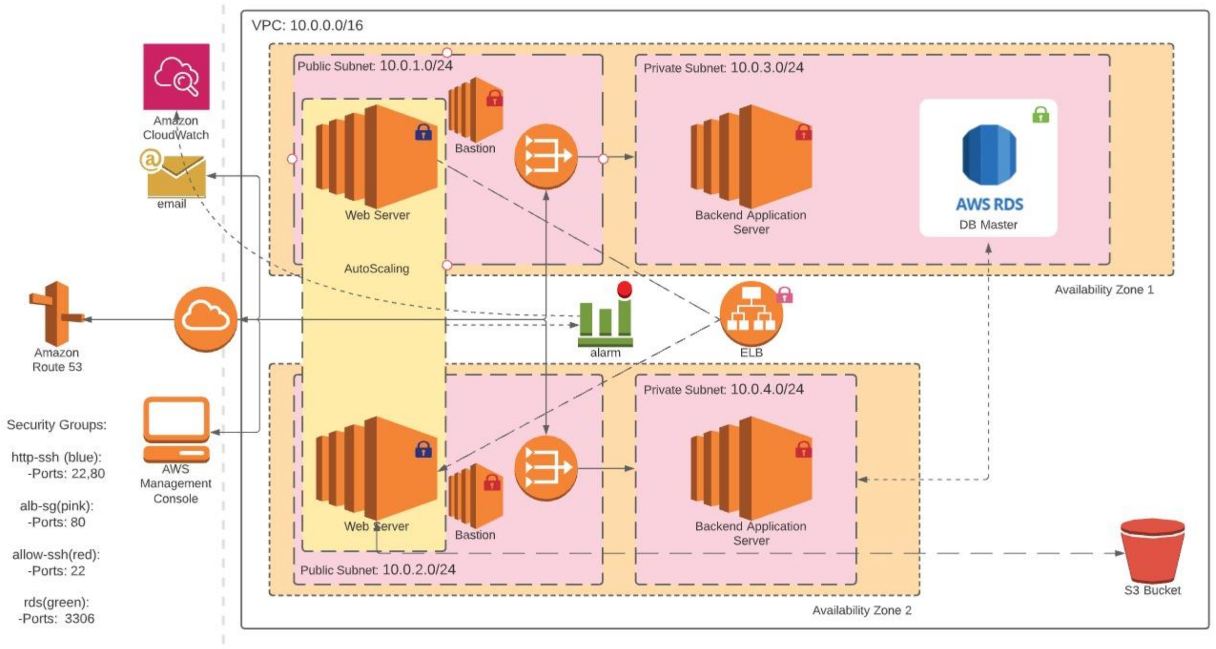


Figure 2.1.1

- In Figure 2.1.1 we can observe the VPC architecture used for our case studies. This architecture is composed of 2 AZ's, with a private and public subnet on each AZ. Many components mentioned in this PDP are also part of the VPC such as RDS, auto scaling, S3 bucket, gateways and more.

2.2 VPC Peering

A VPC peering connection is a networking connection between two VPCs. This enables us to route traffic using IP addresses. The VPC peering connection will help us to transfer data from a public to a private VPC or the other way around, for example. It is possible to establish peering relationships between VPCs across different AWS Regions. This allows VPC resources including EC2 instances, Amazon RDS databases and Lambda functions that run in different AWS Regions to communicate with each other using private IP addresses, without requiring gateways, VPN connections, or separate network appliances.

During the course, we created two different VPCs. One VPC was private, containing the database. In the other hand, the other VPC was public, and it contained the webserver instance. We made use of VPC peering to complete the assignment, to be able to deploy an application using VPC peering, with a database on a private VPC.

2.3 VPN

VPN stands for Virtual Private Network. This service is a technology that can transmit network data over another network. It lets users make use of the network resources in the connected network through VPN. OpenVPN is a virtual private network system that implements techniques to create secure point-to-point or site-to-site connections in routed or bridged configurations and remote access facilities (Yonan).

In the case study, the group will make use of VPN to be able to connect to the AWS cloud environment to the on premises "Infra Lab". This will make possible the secure connection between the two environments and will helps us share data and create on premises back-ups. VPN will also help us connect securely remotely to be able to manage and maintain our infrastructure. We will be using OpenVPN from Pfsense that will be hosted on our Infra Lab environment.

2.3.1 Evidence

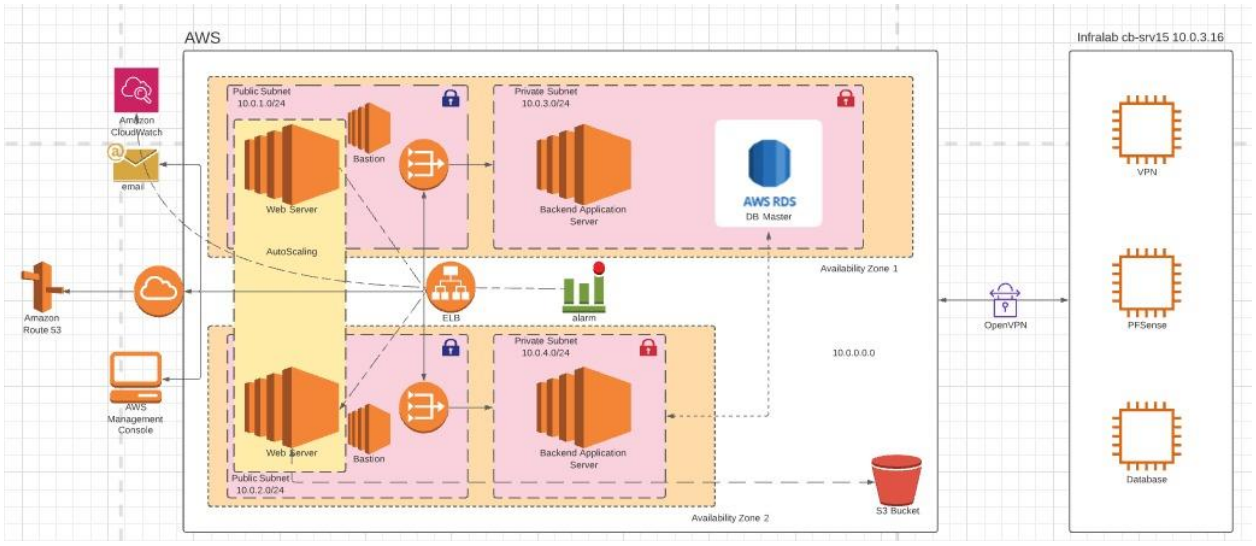


Figure 2.3.1

- In Figure 2.3.1 we can see our network diagram showing how VPN will be working. As you can see, we will connect the VPC with our on-premises server through OpenVPN.

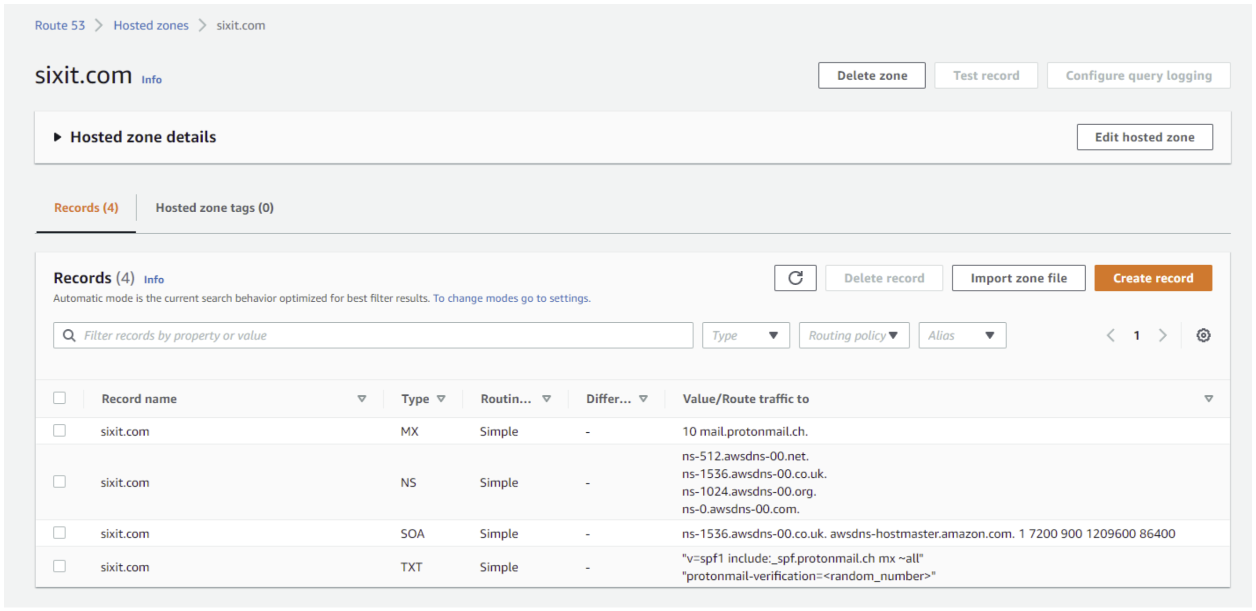
We will use Pfsense as our on-premises firewall and as our VPN server. Facilitation data and file sharing from one environment to the other.

2.4 Route 53

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. Amazon Route 53 connects users and requests to the infrastructure we are going to deploy. In addition, Amazon Route 53 has Traffic Flow feature. it easy for you to manage traffic globally through a variety of routing types, all of which can be combined with DNS Failover in order to enable a variety of low-latency, fault-tolerant architectures (AWS, 2021).

The purpose of using Route 53 in our case studies is to give the infrastructure an access point from outside the AWS infrastructure. This will help IT admins and developers connect to the resources and services in the infrastructure, such as SSH connections. In addition, Route 53 will let clients connect to our application and service using the DNS ‘sixitnews.com’.

2.4.1 Evidence



Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record name	Type	Routin...	Differ...	Value/Route traffic to
sixit.com	MX	Simple	-	10 mail.protonmail.ch.
sixit.com	NS	Simple	-	ns-512.awsdns-00.net. ns-1536.awsdns-00.co.uk. ns-1024.awsdns-00.org. ns-0.awsdns-00.com.
sixit.com	SOA	Simple	-	ns-1536.awsdns-00.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
sixit.com	TXT	Simple	-	"v=spf1 include:_spf.protonmail.ch mx ~all" "protonmail-verification=<random_number>"

Figure 2.4.1

- In Figure 2.4.1 you can see that domain “sixit.com” is being used by us with Route53, in addition we have used Route53 to configure a Proton email server. The email server can be seen from the MX and TXT records.

2.5 Resiliency

Resiliency is the ability for a system to recover from a failure induced by load, attacks, and failures. A resilient workload has the capability to recover when stressed by load, attacks, and failure of any component in the workload's components. A resilient workload not only recovers but recovers in an amount of time that is desired (ServerCentral Turing Group, 2020). Some ways to have a resilient system can be:

- Workloads across different availability zones
- Backups
- Incident management processes

In our case studies, the group will deploy the clients' applications in a resilient infrastructure. Some features that were added to the infrastructure to increase resiliency is that the instances will be hosted in two different availability zones. In addition, the group will add an elastic load balancer and Route53 to balance workloads between instances. Backups will also take place in the infrastructure. For the case study, backups will be saved in the on-premises database. As seen on Figure 2.1.1, the application will be hosted in two different AZ's to balance the workload. In addition, backups from the RDS will be done and stored on our on-premises server.

2.6 API Gateway

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud. Some of the most important features of AWS API Gateway are the following:

- Support for stateful (WebSocket) and stateless (HTTP and REST) APIs.
- Powerful, flexible authentication mechanisms, such as AWS Identity and Access Management policies, Lambda authorizer functions, and Amazon Cognito user pools.
- Developer portal for publishing your APIs.
- CloudTrail logging and monitoring of API usage and API changes.
- CloudWatch access logging and execution logging, including the ability to set alarms.
- Support for custom domain names.
- Integration with AWS WAF for protecting your APIs against common web exploits.
- Integration with AWS X-Ray for understanding and triaging performance latencies.

To learn about and how to make use of API Gateway, we have used a Lambda function and have used the API as a trigger to execute the Lambda function. The API will send a GET request to execute the Lambda function and giving the expected response after being triggered.

2.6.1 Evidence

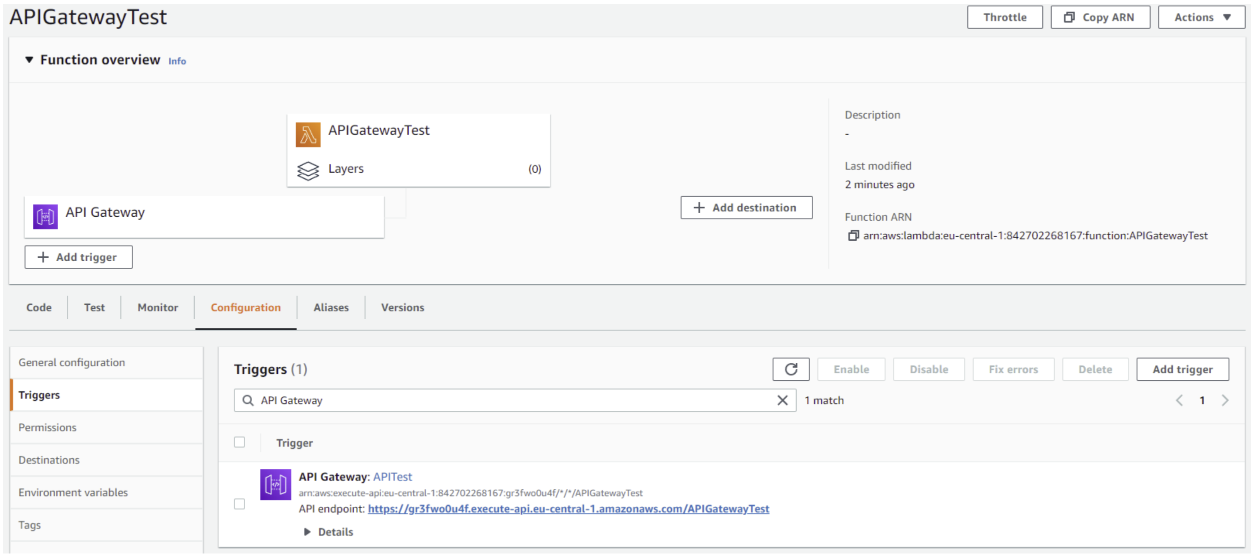
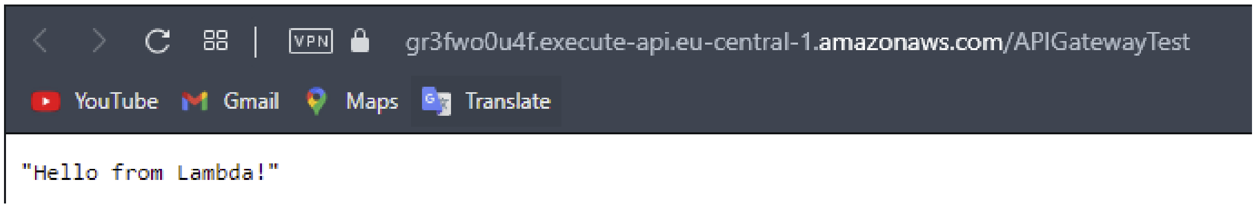


Figure 2.6.1


gr3fwo0u4f.execute-api.eu-central-1.amazonaws.com/APIGatewayTest
YouTube
Gmail
Maps
Translate

```
"Hello from Lambda!"
```

Figure 2.6.2

- In Figure 2.6.1, it is possible to see that the Lambda function and the API have been created and configured properly for the API to trigger the Lambda function. The response expected is “Hello from Lambda!”. In Figure 2.6.2 we can observe the successful response from the Lambda function after being triggered by the API when opening the API URL into the web browser.

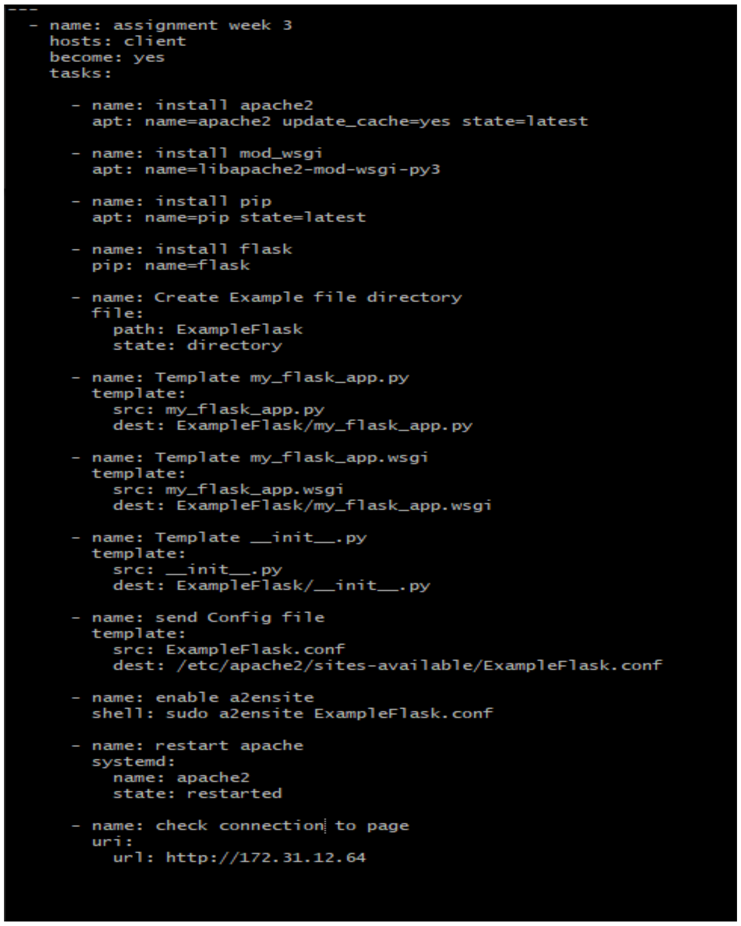
3.0 Automation

3.1 Ansible

Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs (Red Hat, 2021). With Ansible it is possible to create playbooks to execute more than one command or action in one go.

We will make use of Ansible in our case study and multidisciplinary project as it will help us deploy new applications and services faster, manage our infrastructure more efficiently, and improve provisioning, orchestration, and security and compliance. By creating playbooks, we will manage to launch the application in a more efficient way, while orchestrating our infrastructure.

3.1.1 Evidence



```
---
  - name: assignment week 3
    hosts: client
    become: yes
    tasks:

      - name: install apache2
        apt: name=apache2 update_cache=yes state=latest

      - name: install mod_wsgi
        apt: name=libapache2-mod-wsgi-py3

      - name: install pip
        apt: name=pip state=latest

      - name: install flask
        pip: name=flask

      - name: Create Example file directory
        file:
          path: ExampleFlask
          state: directory

      - name: Template my_flask_app.py
        template:
          src: my_flask_app.py
          dest: ExampleFlask/my_flask_app.py

      - name: Template my_flask_app.wsgi
        template:
          src: my_flask_app.wsgi
          dest: ExampleFlask/my_flask_app.wsgi

      - name: Template __init__.py
        template:
          src: __init__.py
          dest: ExampleFlask/__init__.py

      - name: send Config file
        template:
          src: ExampleFlask.conf
          dest: /etc/apache2/sites-available/ExampleFlask.conf

      - name: enable a2ensite
        shell: sudo a2ensite ExampleFlask.conf

      - name: restart apache
        systemd:
          name: apache2
          state: restarted

      - name: check connection to page
        uri:
          url: http://172.31.12.64
```

Figure 3.1.1

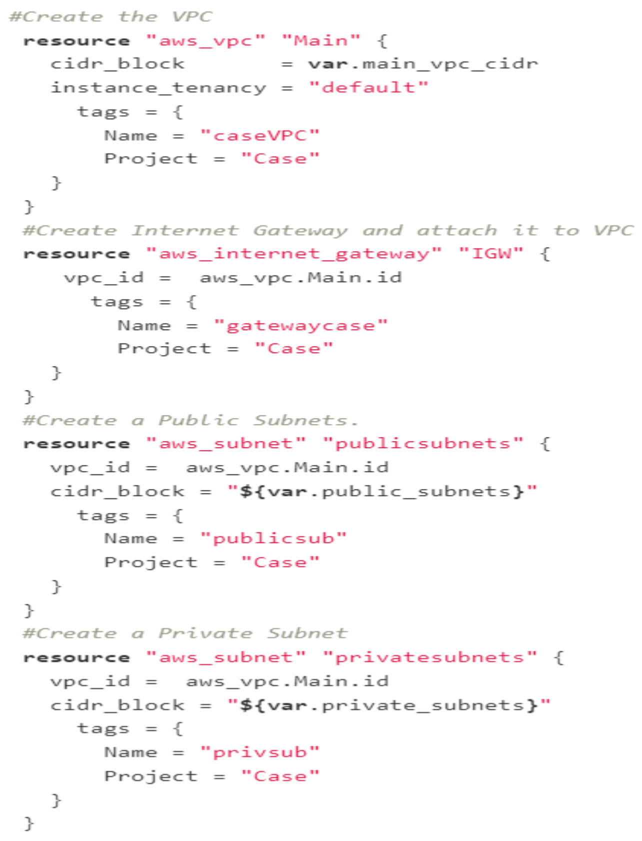
In figure 3.1.1, it is possible to observe an Ansible playbook that will install Apache2 webserver and deploy a simple Flask application.

3.2 Terraform

Terraform is an open-source infrastructure as code software tool that provides a consistent CLI workflow to manage hundreds of cloud services. That means declaring infrastructure components in configuration files that are then used by Terraform to provision, adjust, and tear down infrastructure in various cloud providers (Bensky, 2020). Terraform also uses modules to enable different services other than computing instances and network. This is very useful because it will create pieces of reusable and configurable code to implement, update or upgrade different services and computing instances.

We will use Terraform in the case study and multidisciplinary project to create, manage, upgrade, update or delete instances, VPCs and other network elements and services such as a webserver. This will enable the team to work in a more efficient and faster way as the code can be configured, shared and reuse to keep changing the infrastructure for it to be the best possible.

3.2.1 Evidence



```
#Create the VPC
 resource "aws_vpc" "Main" {
   cidr_block         = var.main_vpc_cidr
   instance_tenancy = "default"
     tags = {
       Name = "caseVPC"
       Project = "Case"
   }
 }
 #Create Internet Gateway and attach it to VPC
 resource "aws_internet_gateway" "IGW" {
    vpc_id =  aws_vpc.Main.id
      tags = {
        Name = "gatewaycase"
        Project = "Case"
   }
 }
 #Create a Public Subnets.
 resource "aws_subnet" "publicsubnets" {
   vpc_id =  aws_vpc.Main.id
   cidr_block = "${var.public_subnets}"
     tags = {
       Name = "publicsub"
       Project = "Case"
   }
 }
 #Create a Private Subnet
 resource "aws_subnet" "privatesubnets" {
   vpc_id =  aws_vpc.Main.id
   cidr_block = "${var.private_subnets}"
     tags = {
       Name = "privsub"
       Project = "Case"
   }
 }
```

Figure 3.2.1

A part of the Terraform code that will create a VPC with a private and a public subnet, an internet gateway and the corresponding routing tables can be observed in figure 3.2.1. The code can be seen in our github: https://git.fhict.nl/I408431/casestudy-group6.git

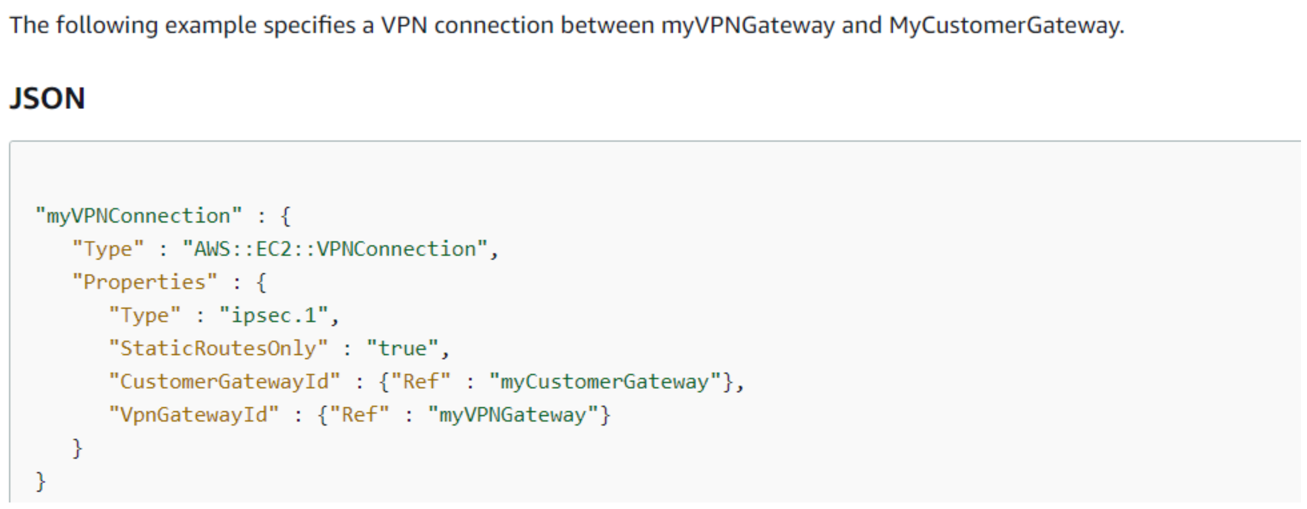
3.3 CloudFormation

AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. CloudFormation can save us time and effort because we do not have to create and configure each instance, network component or any other AWS service manually. This service is essentially like Terraform but for AWS Cloud only. Some use cases for CloudFormation can be to simplify infrastructure management and configuration, also facilitates replication of resources and infrastructure and easily control, track and do changes to the infrastructure. Some advantages of CloudFormation are:

- Is possible to use multiple languages to write a CloudFormation script as json or yaml.
- When you add new features to the script it will update all the infrastructure and create the new items.
- If a feature or resource is no longer needed the CloudFormation will delete this last.
- As it is an AWS feature is much easier to create different resources and adding new items to the infrastructure.

During the lectures, the group completed an assignment to create a VPN connection using CloudFormation. We used CloudFormation to create the VPN connection between our VPC and our on-premises environment. In addition, for the projects we created a Lambda function that was scheduled to be executed once per day. The Lambda function was used to check for the unused elastic IPs in our AWS console. This way, we can reduce some costs and check which IPs from this service are not being used.

3.3.1 Evidence


The following example specifies a VPN connection between myVPNGateway and MyCustomerGateway.

JSON

```
"myVPNConnection" : {
   "Type" : "AWS::EC2::VPNConnection",
   "Properties" : {
      "Type" : "ipsec.1",
      "StaticRoutesOnly" : "true",
      "CustomerGatewayId" : {"Ref" : "myCustomerGateway"},
      "VpnGatewayId" : {"Ref" : "myVPNGateway"}
   }
}
```

Figure 3.3.1

- In Figure 3.3.1 we can see an example of a VPN connection created with CloudFormation. The connection is stablished from the VPN gateway (in AWS) and “Costumer Gateway” (on-premises server).

3.4 AWS Step Functions

Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications. Through Step Functions' graphical console, you see your application’s workflow as a series of event-driven steps. Step Functions is based on state machines and tasks. A state machine is a workflow. A task is a state in a workflow that represents a single unit of work that another AWS service performs. Each step in a workflow is a state (AWS, 2021).

Unfortunately, the team did not implement a Step Function into the projects due to time restrains. Even though we did not use it, we brainstormed some uses. We concluded that one of the best uses for Step Functions would be to use different triggers such as CloudWatch alarms, events or requests that will send SNS notifications. We use SNS notifications in two cases, so automating this would have been a plus.

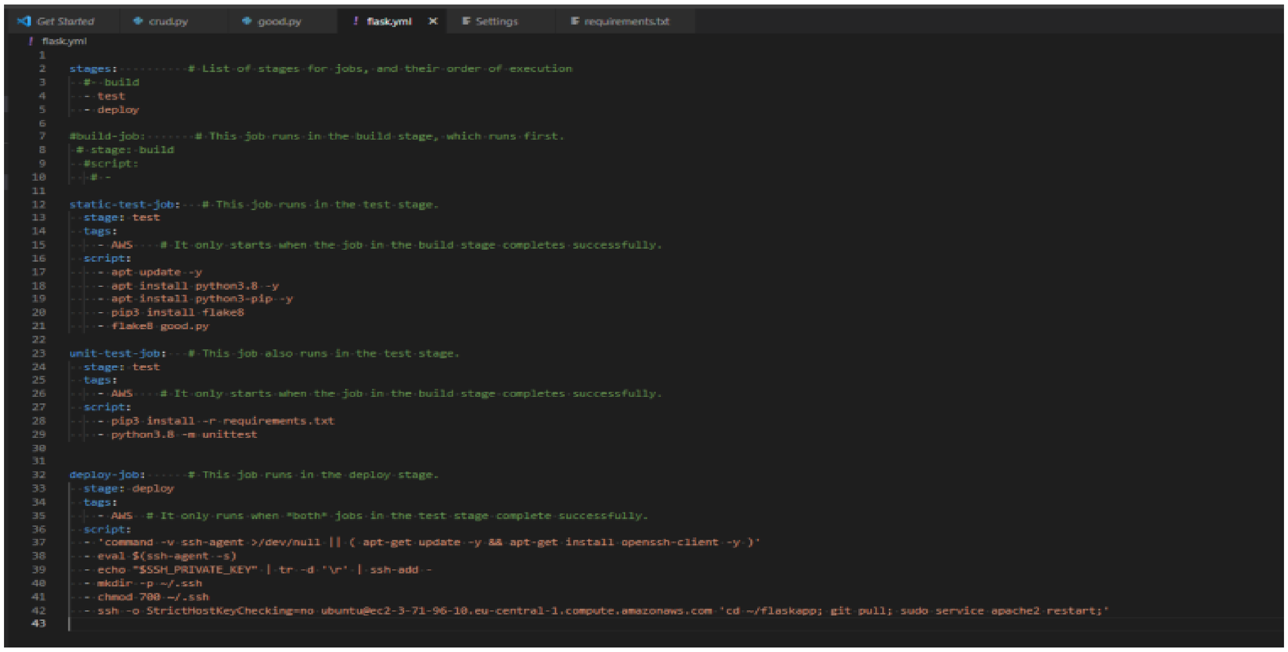
3.5 CI/CD Pipelines

A CI/CD pipeline is a series of steps that must be performed in order to deliver a new version of software. A CI/CD pipeline introduces monitoring and automation to improve the process of application development, particularly at the integration and testing phases, as well as during delivery and deployment (Red Hat, 2019). For pipelines to work and execute properly, it consists of different stages that perform different tasks. The before mentioned stages are divided into:

- **Build**: The stage where the application is compiled.
- **Test**: The stage where code is tested. Automation here can save both time and effort.
- **Release**: The stage where the application is delivered to the repository.
- **Deploy**: In this stage code is deployed to production.
- **Validation and compliance**: The steps to validate a build are determined by the needs of your organization.

Pipelines will have a lot of focus in our projects. In the multidisciplinary assignment, we will be using pipelines to satisfy the client needs of continuous integration and deployment. It is very important for the client to have pipelines since it is a new and in development product. For this reason, implementing pipelines was a "no brainer". In addition, pipelines will also be part of the case study, as we want to make our GUI application and the infrastructure itself as simple, as possible for the end users.

3.5.1 Evidence



```yaml

stages:          # List of stages for jobs, and their order of execution
  #- build
  - test
  - deploy

#build-job:       # This job runs in the build stage, which runs first.
 # stage: build
  #script:
   # -

static-test-job:   # This job runs in the test stage.
  stage: test
  tags:
    - AWS    # It only starts when the job in the build stage completes successfully.
  script:
    - apt update -y
    - apt install python3.8 -y
    - apt install python3-pip -y
    - pip3 install flake8
    - flake8 good.py

unit-test-job:   # This job also runs in the test stage.
  stage: test
  tags:
    - AWS    # It only starts when the job in the build stage completes successfully.
  script:
    - pip3 install -r requirements.txt
    - python3.8 -m unittest


deploy-job:      # This job runs in the deploy stage.
  stage: deploy
  tags:
    - AWS  # It only runs when *both* jobs in the test stage complete successfully.
  script:
  - 'command -v ssh-agent >/dev/null || ( apt-get update -y && apt-get install openssh-client -y )'
  - eval $(ssh-agent -s)
  - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
  - mkdir -p ~/.ssh
  - chmod 700 ~/.ssh
  - ssh -o StrictHostKeyChecking=no ubuntu@ec2-3-71-96-10.eu-central-1.compute.amazonaws.com 'cd ~/flaskapp; git pull; sudo service apache2 restart;'

```

Figure 3.5.1

- In Figure 3.5.1 we can observe a piece of code that will perform different tasks in a certain order to deploy a sample Flask application to an EC2 instance using CI/CD Pipelines. On static-test job we install the necessary packages and libraries to run the application. In the unit-test-job tasks, we install the requirements for the flask app to run without any problems, and install the libraries used by the app. On the last section, deploy-job, the application is deployed to the webserver and restarts to have the latest version of the code.

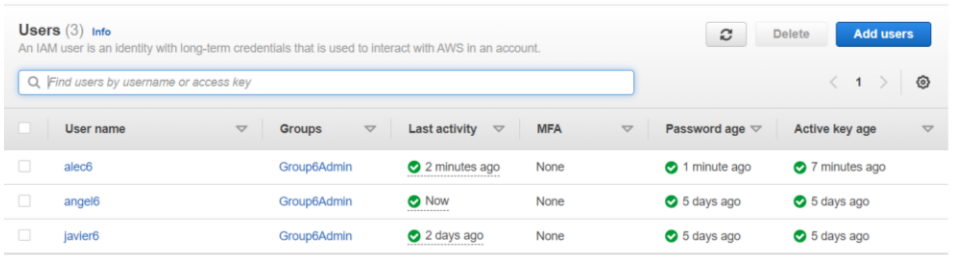
4.0 Security

4.1 IAM User Policies

After getting a root user, the group created different IAM users for each one of the members. We assigned policies to these users such as a change password policy. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an IAM principal (user or role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. AWS supports six types of policies: identity-based policies, resource-based policies, permissions boundaries, Organizations SCPs, ACLs, and session policies (AWS, 2021).

We also used policies to give the group member's IAM users administrator policies because all the members will have to work in different features and components at different times, this will make it easier for us to complete the projects and assignments.

4.1.1 Evidence



Users (3)

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

User name	Groups	Last activity	MFA	Password age	Active key age
alec6	Group6Admin	✓ 2 minutes ago	None	✓ 1 minute ago	✓ 7 minutes ago
angel6	Group6Admin	✓ Now	None	✓ 5 days ago	✓ 5 days ago
javier6	Group6Admin	✓ 2 days ago	None	✓ 5 days ago	✓ 5 days ago

Figure 4.1.1

- In figure 4.1.1 we can observe the 3 users created with the policies in place.

4.2 Securing Access to Object Storage

Amazon S3 offers access policy options broadly categorized as resource-based policies and user policies. Access policies that you attach to your resources (buckets and objects) are referred to as resource-based policies. For example, bucket policies and access control lists (ACLs) are resource-based policies. You can also attach access policies to users in your account. These are called user policies. You can choose to use resource-based policies, user policies, or some combination of these to manage permissions to your Amazon S3 resources (AWS, 2021).

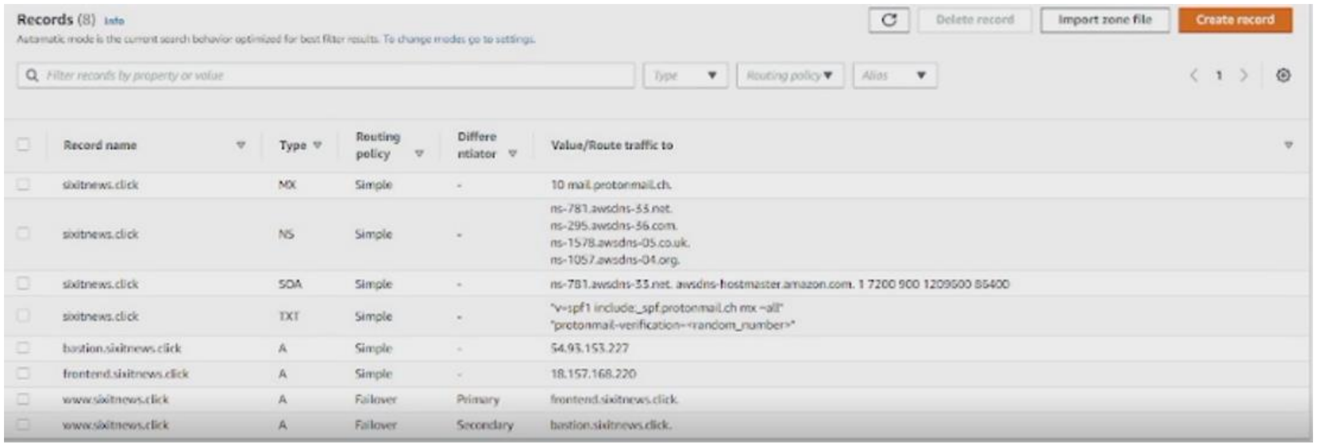
S3 Buckets policies will be set in place to give the right permissions only to the group users because all the group member's users are admins. This will create a secure bucket to store objects and data.

4.3 Route53 Failover

When having more than one resource performing the same function, it is possible to configure Route53 to check health of your resources and then have the DNS queries only responding to the instances that are healthy. In simple configurations, you create a group of records that all have the same name and type. For example, we can have different A type records to "sixitnews.click". You then configure Route 53 to check the health of the corresponding resources. If one of the instances is not healthy then the DNS will reroute traffic to the failover record.

In both projects, case study and multi-disciplinary, the team implemented Route53 failover records. We decided to use simple routing to route all the HTTP traffic through the DNS "sixitnews.click". The team then implemented a failover record for the web server and redirect the traffic to the bastion. The bastion will then be hosting a webpage that will indicate the user that the web application is down. In simple words, if the webserver is not healthy, we will use the bastion as a webserver to report that problems with the web application are happening.

4.3.1 Evidence



Records (8) Info

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record name	Type	Routing policy	Differentiator	Value/Route traffic to
sixitnews.click	MX	Simple	-	10 mail.protonmail.ch.
sixitnews.click	NS	Simple	-	ns-781.awsdns-33.net. ns-295.awsdns-36.com. ns-1578.awsdns-05.co.uk. ns-1057.awsdns-04.org.
sixitnews.click	SOA	Simple	-	ns-781.awsdns-33.net. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
sixitnews.click	TXT	Simple	-	"v=spf1 include:_spf.protonmail.ch mx ~all" "protonmail-verification=<random_number>"
bastion.sixitnews.click	A	Simple	-	54.93.153.227
frontend.sixitnews.click	A	Simple	-	18.157.168.220
www.sixitnews.click	A	Failover	Primary	frontend.sixitnews.click.
www.sixitnews.click	A	Failover	Secondary	bastion.sixitnews.click.

Figure 4.3.1

- In Figure 4.3.1 we can observe the records in sixitnews.click DNS. We have email records for an email server. In addition, we have the failover records. We can observe that the primary failover record is on the front-end web server, while the secondary failover record is on the bastion host, this way if the health check in the front-end webserver is not successful then the bastion will replace it.

4.4 Flow Logs

VPC Flow Logs is a feature that enables you to capture information about the traffic going to and from network interfaces in the VPC. Flow log data can be published to Amazon CloudWatch Logs. It is possible to create Flow Logs for specific subnets or a whole VPC. Nevertheless, all the interfaces in that subnet or VPC will be monitored with Flow Logs. We can also choose to monitor certain type of traffic such as accepted, rejected or all the traffic in the network. With the use of Flow Logs we can see how the network is behaving and what can the team do to improve the efficiency of the infrastructure.

For both projects the team implemented Flow Logs to monitor the VPC for all network traffic, including reject requests by the infrastructure. One of the main reasons we decided to implement these was in case something unusual happened with our infrastructure. For example, sometimes the API of our application looked like it was not properly working, and we checked these errors from the Flow Logs. We decided to monitor all the traffic and use SNS to send the logs to the team's email. We tested the this by sending multiple HTTP 'GET' requests and check the logs received to the email by SNS.

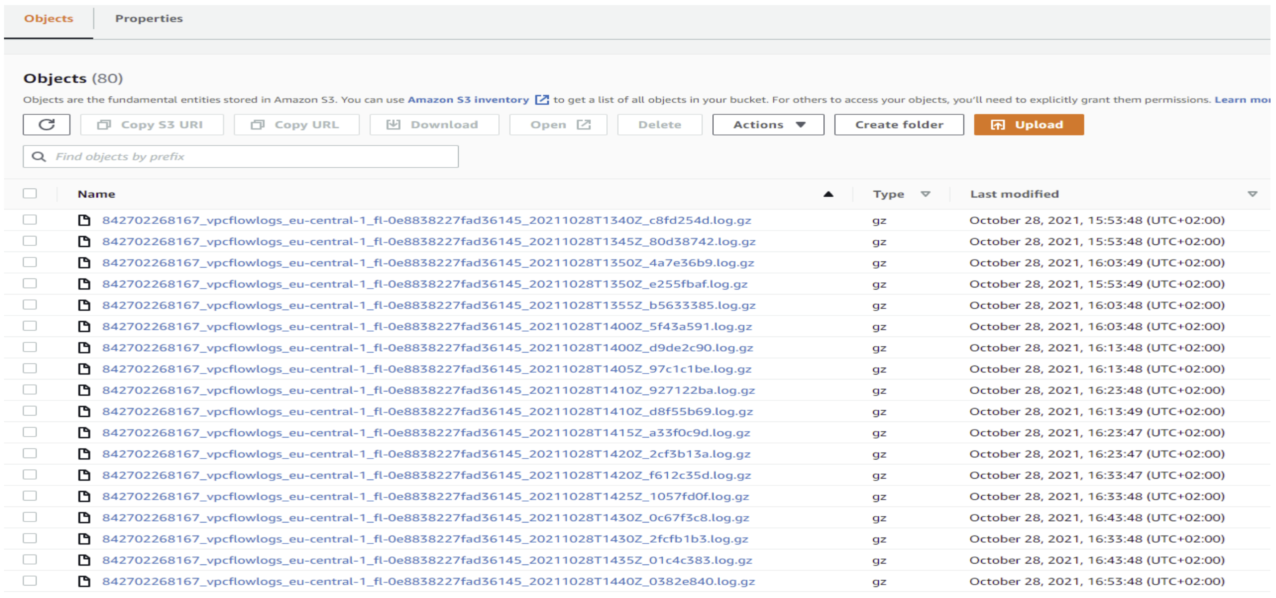
5.0 Supporting Services

5.1 S3 Objects

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. S3 buckets are where the objects are stored. To store your data in Amazon S3, you first create a bucket and specify a bucket name and AWS Region. Then, you upload your data to that bucket as objects in Amazon S3. Each object has a key, which is the unique identifier for the object within the bucket (AWS, 2021)

We used S3 Bucket to host a static website describing our expectations for this semester. In the other hand, we will use S3 Buckets to store data in the cloud for our case and multidisciplinary projects. We are also planning on creating a one-page contact form so that clients can contact us in case of errors, bugs or other problems of the infrastructure. In addition, we will use S3 buckets to save our infrastructure logs.

5.1.1 Evidence



Objects (80)

Objects are the fundamental entities stored in Amazon S3. You can use **Amazon S3 inventory** to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. **Learn mo**

Name	Type	Last modified
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1340Z_c8fd254d.log.gz	gz	October 28, 2021, 15:53:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1345Z_80d38742.log.gz	gz	October 28, 2021, 15:53:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1350Z_4a7e36b9.log.gz	gz	October 28, 2021, 16:03:49 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1350Z_e255fbaf.log.gz	gz	October 28, 2021, 15:53:49 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1355Z_b5633385.log.gz	gz	October 28, 2021, 16:03:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1400Z_5f43a591.log.gz	gz	October 28, 2021, 16:03:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1400Z_d9de2c90.log.gz	gz	October 28, 2021, 16:13:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1405Z_97c1c1be.log.gz	gz	October 28, 2021, 16:13:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1410Z_927122ba.log.gz	gz	October 28, 2021, 16:23:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1410Z_d8f55b69.log.gz	gz	October 28, 2021, 16:13:49 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1415Z_a33f0c9d.log.gz	gz	October 28, 2021, 16:23:47 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1420Z_2cf3b13a.log.gz	gz	October 28, 2021, 16:23:47 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1420Z_f612c35d.log.gz	gz	October 28, 2021, 16:33:47 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1425Z_1057fd0f.log.gz	gz	October 28, 2021, 16:33:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1430Z_0c67f3c8.log.gz	gz	October 28, 2021, 16:43:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1430Z_2fcfb1b3.log.gz	gz	October 28, 2021, 16:33:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1435Z_01c4c383.log.gz	gz	October 28, 2021, 16:43:48 (UTC+02:00)
842702268167_vpcflowlogs_eu-central-1_fl-0e8838227fad36145_20211028T1440Z_0382e840.log.gz	gz	October 28, 2021, 16:53:48 (UTC+02:00)

Figure 5.1.1

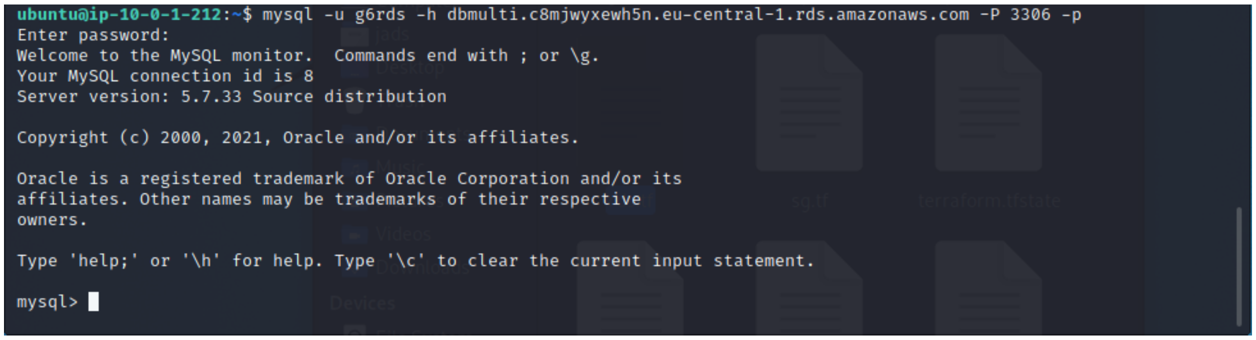
- In Figure 5.1.1 we can see that we are using an S3 Bucket to save the network logs that were monitored. This will be done to take action in case any unusual activities happen in the network.

5.2 RDS

Amazon RDS is Amazon's Relational Database Service. This service makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups (AWS, 2021). There are several options to launch different types of RDS instances, and provides different types of database engines including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server.

The team will use RDS instances to host the application in our case study as well as launching one if the multidisciplinary client requires a relational database. This will save 5 different tables that will save user and application information for our blog to perform properly. For the projects we will make use of 2 types of databases. The case study will be using MySQL as is the most popular database system and the user can find support and information about it. In the other hand, the multi-disciplinary project will use, by requirement of the client, PostgreSQL database.

5.2.1 Evidence



```
ubuntu@ip-10-0-1-212:~$ mysql -u g6rds -h dbmulti.c8mjwyxewh5n.eu-central-1.rds.amazonaws.com -P 3306 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.33 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

Figure 5.2.1

- In Figure 5.2.1, an example of the connection to our MySQL database server running in the private subnet. The connection happens from our bastion host. We use MySQL client to be able to connect to the RDS endpoint through port 3306.

5.3 Elastic Load Balancer

Elastic Load Balancing automatically distributes incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to most workloads. Elastic Load Balancing supports the following load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers (AWS, 2021).

The group will make use of a load balancer to increase the availability of the applications hosted. The target group of the application load balancer will consist of our webservers, this way we will be able to balance the workload in them and attach the load balancer to the health checks implemented for the CloudWatch alarms. In Figure 2.1.1, we can observe the use of the elastic load balancer. We will connect the elastic load balancer to our autoscaling group so we can balance and get health checks from the instances that are part of the autoscaling group. In addition, we will make use of the load balancer to make the back end and front end connected through the load balancers open port.

6.0 Monitoring

6.1 CloudWatch

CloudWatch is a monitoring tool that helps developers and infrastructure engineers by providing data and insight about changes and activities in the application, communication protocols, or the infrastructure itself. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. This will help us avoid or detect certain abnormal activities that can be malicious to the infrastructure.

The team will be using CloudWatch to monitor our infrastructure and application by collecting logs and analyzing them. Alerts for CloudWatch have also been put in place. CloudWatch will also help us see how the application and infrastructure will respond once we start stress testing it.

6.1.1 Evidence

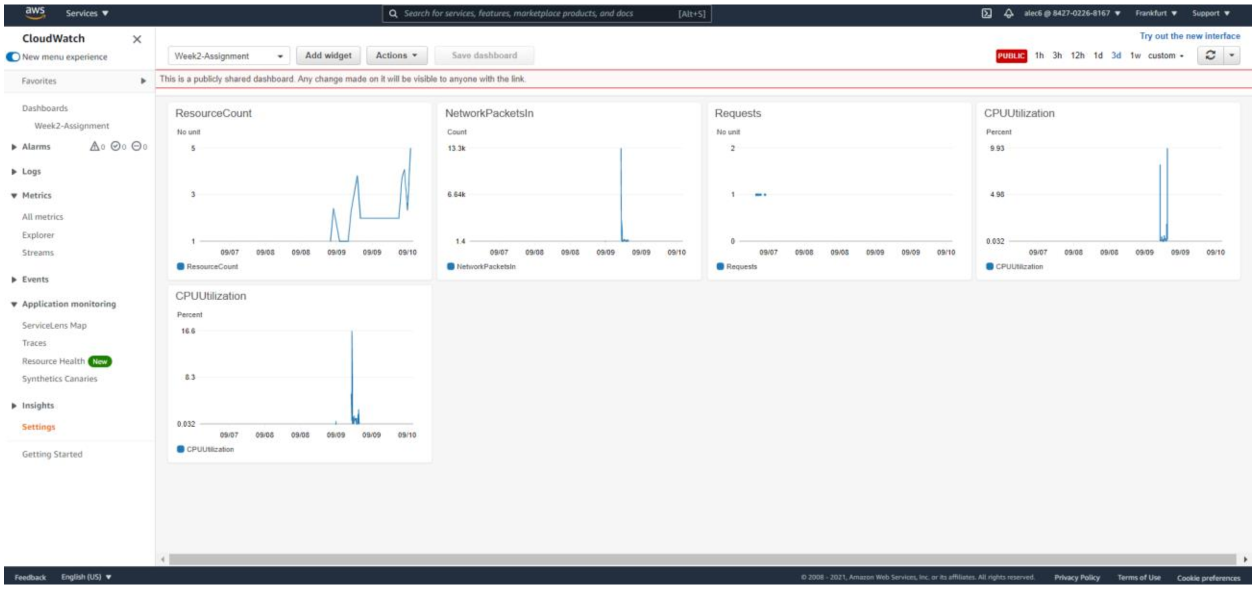


Figure 6.1.1

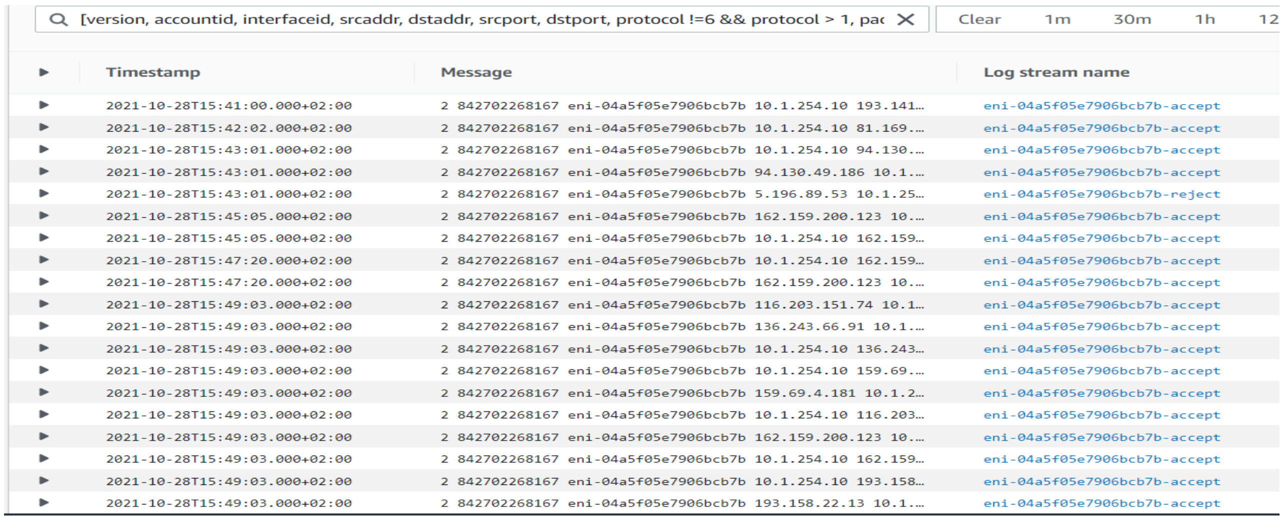
- In figure 6.1.1 we can observe the different metrics in CloudWatch for one of our instances.

6.2 Monitoring & Analysis Network Traffic Log Files

Amazon CloudWatch Logs allow us to monitor, store, and access log files from the instances, Route 53, and other sources. CloudWatch Logs enables you to centralize the logs from all the infrastructure, applications, and AWS services that are being used. CloudWatch Logs enables you to see all the logs, as a single and consistent flow of events ordered by time. It is also possible to filter and query them as desired. It is easy to view the logs as we can search them by these filters or queries and store them in a S3 Bucket.

During the lectures we completed the assignment regarding monitoring and analyzing the network logs. First, we created the logs through CloudWatch with the proper conditions. Later, as seen on Figure 5.1.1 we stored these log files in an S3 Bucket for further analysis if needed. We will also make use of Athenea to help us monitor and send notifications regarding malicious activities such as port scans.

6.2.1 Evidence



[version, accountid, interfaceid, srcaddr, dstaddr, srcport, dstport, protocol !=6 && protocol > 1, pac

Timestamp	Message	Log stream name
2021-10-28T15:41:00.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 193.141…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:42:02.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 81.169.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:43:01.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 94.130.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:43:01.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 94.130.49.186 10.1.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:43:01.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 5.196.89.53 10.1.25…	eni-04a5f05e7906bcb7b-reject
2021-10-28T15:45:05.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 162.159.200.123 10.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:45:05.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 162.159…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:47:20.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 162.159…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:47:20.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 162.159.200.123 10.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 116.203.151.74 10.1…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 136.243.66.91 10.1.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 136.243…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 159.69.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 159.69.4.181 10.1.2…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 116.203…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 162.159.200.123 10.…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 162.159…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 10.1.254.10 193.158…	eni-04a5f05e7906bcb7b-accept
2021-10-28T15:49:03.000+02:00	2 842702268167 eni-04a5f05e7906bcb7b 193.158.22.13 10.1.…	eni-04a5f05e7906bcb7b-accept

Figure 6.2.1

- In Figure 6.2.1 we can see the log events for two different log streams (Accept and Reject) from our VPC according to the conditions in place.

6.3 CloudWatch Alarms

As mentioned before, CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. We will make use of this data to monitor and create metric alarms based on the data and utilization of our EC2 instances. A metric alarm watches a single CloudWatch metric, or the result of an expression based on CloudWatch metrics. The alarm performs one or more actions based on the value of the metric or expression relative to a threshold over time periods.

In both of our projects we will make use of CloudWatch metric alarms to monitor the CPU, RAM utilization and network traffic and trigger the auto scaling group based on the thresholds specified. For example, two alarms are in place for CPU utilization in the webserver. The alarms will cause the auto scaling group to create or destroy an instance based on the threshold 10 to 70 percent. In addition, the alarms will also send a notification together with SNS if the auto scaling group creates or destroys an instance.

6.3.1 Evidence

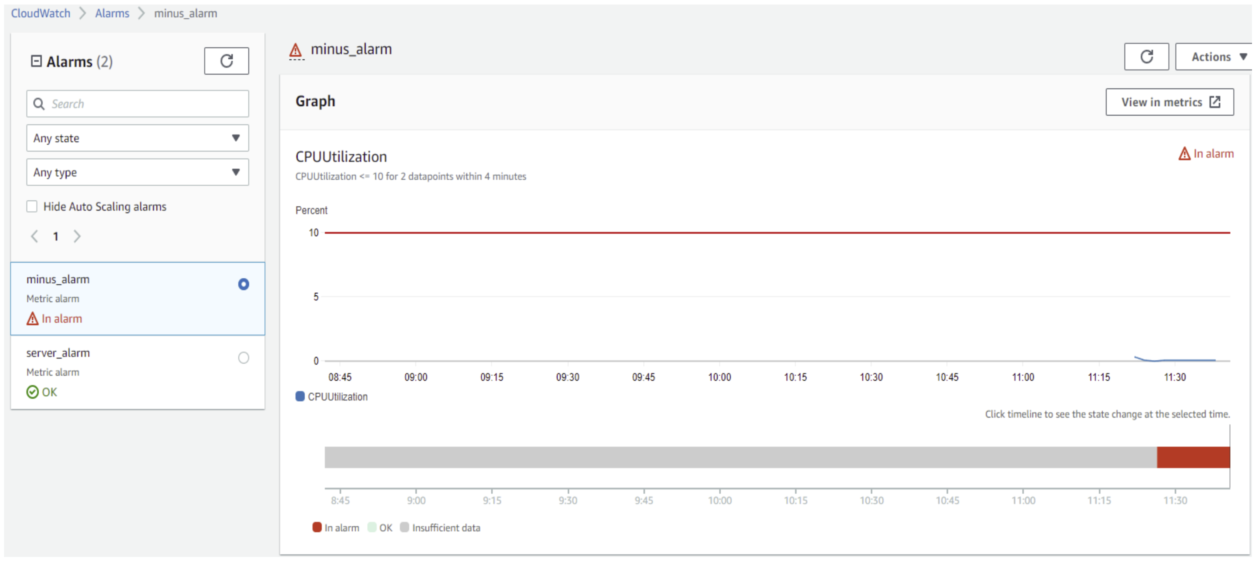


Figure 6.3.1

- In Figure 6.3.1, the alarms stablished can be appreciated. Two alarms are in place, taking health checks every 4 minutes. The “minus_alarm” will destroy an instance from the auto scaling group if this is above the desired capacity. In the other hand, “server_alarm” will create an instance from the auto scaling group template. The alarms will also work with SNS to send us notifications every time an instance is created or destroyed.

6.4 Elastic Stack (MetricBeats & Kibana)

Elastic Stack is a group of open-source products from Elastic designed to help users take data from any type of source and in any format and search, analyze, and visualize that data in real time. The products in this group are:

- **Elastic Search:** search engine built on top of Apache Lucene and released under an Apache license. It is Java-based and can search and index document files in diverse formats.
- **Logstash:** data collection engine that unifies data from disparate sources.
- **Beats or MetricBeats:** “data shippers” that are installed on servers as agents used to send different types of operational data to Elasticsearch either directly or through Logstash
- **Kibana:** data visualization and exploration tool from that is specialized for large volumes of streaming and real-time data.

In this section, we learned how to extract and analyze data from our EC2 instances running in our environment. To perform these tasks we used Elastic Stack, more specifically Elastic

Search, MetricBeats and Kibana. The group managed to create a dashboard using Elastic Stack to monitor the data from the web server hosted in our environment. The dashboard is used to analyze different types of data such as: bytes usage, the CPU usage, the network incoming and the process PID.

6.4.1 Evidence

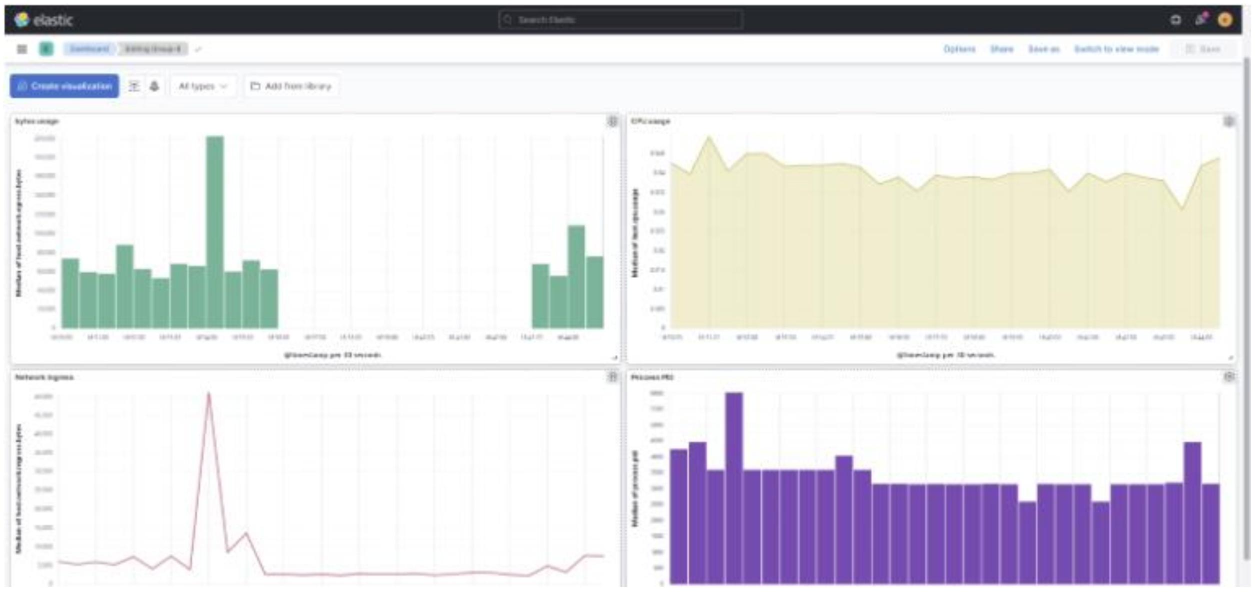


Figure 6.4.1

- In Figure 6.4.3, you can observe the dashboard that was created by the team in order to monitor the web server instances running in AWS. The data collected is bytes usage, the CPU usage, the network incoming and the process PID.

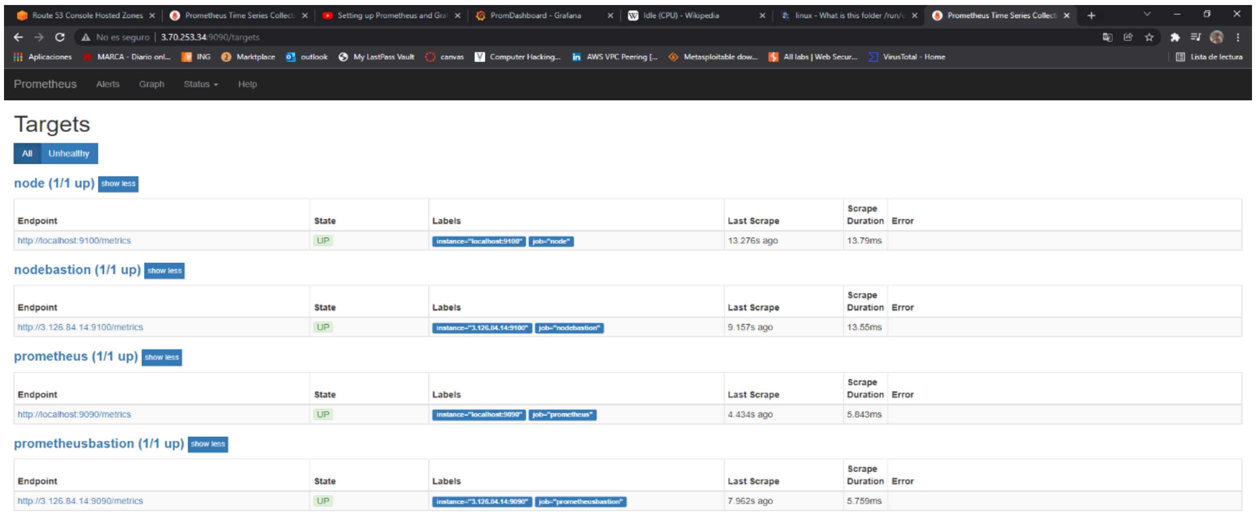
6.5 Prometheus & Node Exporter

Prometheus is an open-source software that facilitates the monitoring of servers, applications, services, and databases by recording and processing numeric data extracted, along with timestamps, from the virtual machine or server. This solution collects metrics from the targets declared in the Prometheus server configuration file. The primary method of data collection is scraping metrics from instrumented applications and services, which expose metrics in a text format via HTTP endpoints about the health, performance, and traffic. The Prometheus server handles the scraping of metrics in the infrastructure. A configuration file needs to be modified in order to add target hosts, schedule tasks, alarms, etc.

The Prometheus node exporter is an exporter for virtual machine and physical server's metrics from hardware and kernel metrics. This will allow the team to monitor more data extracted from the targets.

The team implemented Prometheus and node exporter in the case study project. We chose Prometheus and node exporter to extract data from the front-end of the application deployed and the server itself. In the other hand, we also monitored the network traffic in our bastion host because is the host that will make it possible to connect remotely to the network. This are the two instances that are exposed on the internet and monitoring the traffic and behavior is very important because these instances are the most vulnerable to attacks. In addition, the bastion is also serving as the Prometheus server listening in port 9090 and node exporter exposes port 9100.

6.5.1 Evidence



Targets

node (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node"	13.276s ago	13.79ms	

nodebastion (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://3.126.84.14:9100/metrics	UP	instance="3.126.84.14:9100" job="nodebastion"	9.157s ago	13.55ms	

prometheus (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	4.434s ago	5.843ms	

prometheusbastion (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://3.126.84.14:9090/metrics	UP	instance="3.126.84.14:9090" job="prometheusbastion"	7.962s ago	5.759ms	

Figure 6.5.1

- In Figure 6.5.1 we can observe the 4 targets that are being monitored by Prometheus. 2 node exporter and 2 Prometheus targets are extracting numeric data for analysis.

6.6 Grafana

Grafana is an open-source service that allows you to collect, visualize, alert, and graph metrics from different storage methods. It is possible to use sample dashboards or create your own to visualize data in an organized manner. Grafana has different features such as:

- **Visualizing**: Takes data and it is possible to create clear graphs with preferred legends and data.
- **Dynamic Dashboards**: Create dashboards or use open-source dashboards that will help you monitor the infrastructure in an organize manner.
- **Explore Metrics**: Explore metrics from different targets with timestamps that will allow us further analysis in case of unusual activities.
- **Explore Logs**: Logs can also be exported into Grafana to monitor and/or track logs from applications or services for example Apache error logs.
- **Alerting**: It is possible to visually define alert rules for your most important metrics. Grafana will continuously evaluate and send notifications.
- Mixed Data Sources: Mix different data sources in the same graph.

During the semester we used Grafana to visualize data extracted using Prometheus and node exporter. We created a dashboard that consists of three different sections. First was basic metrics. In this section we displayed the basic metrics such as CPU and Memory usage. In the second section, we monitored Network Traffic in the bastion and web server instance and displayed the data with different types of graphs and different protocols. In the last section we monitored the nodes from Prometheus and node exporter to check if nodes did not reach or took too long, as a sign of unusual.

6.6.1 Evidence

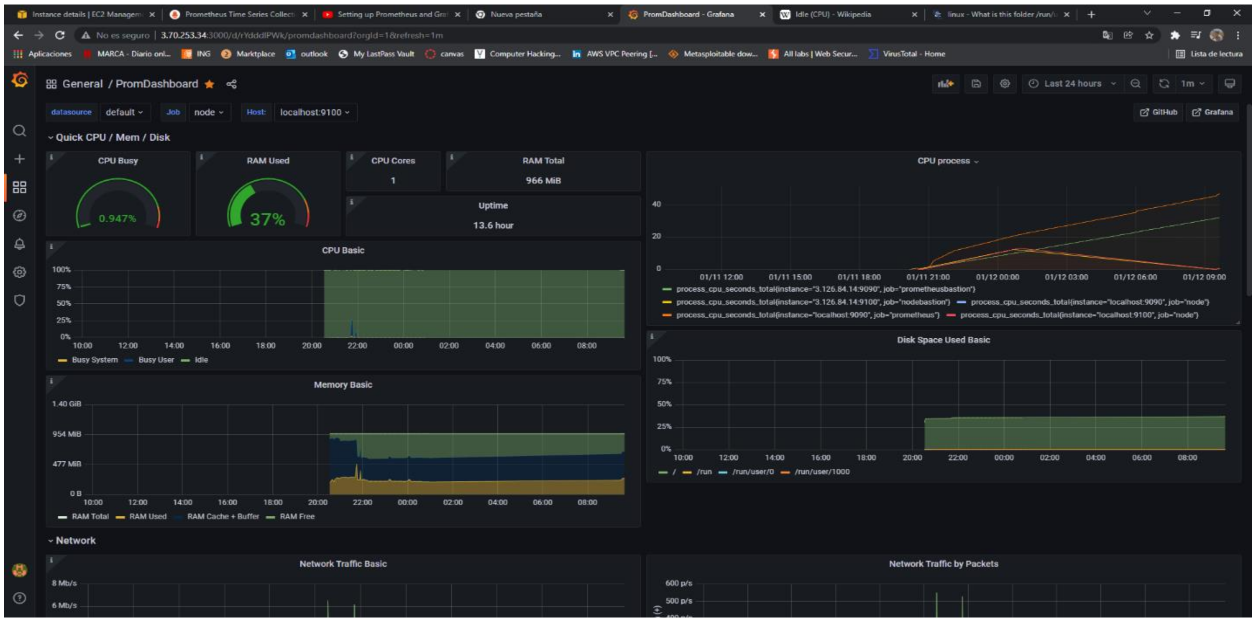


Figure 6.6.1

- In Figure 6.7.1 you can observe the dashboard that was created by the team to analyze and monitor the data extracted from the targets using Prometheus and Node Exporter. The dashboard contains basic metrics and a special section for Network Traffic.

7.0 Personal Leadership

Being entrepreneurial regarding the ICT assignments and personal development, while being aware of own learning capacity and keeping in mind what ambitions that drive ICT professionals and/or which types of positions.

In my personal experience this semester, although we have named someone else 'Leader', I have been putting effort to keep my teammates motivated to keep pushing ourselves and to keep learning so that we can give the best possible solution to our clients. In addition, I sense that everyone is doing their part because of this motivation. At the end of the semester, we helped each other stay motivated, since it was a lot of work and stress.

8.0 Problem Solving

Critically consider ICT assignments from various perspectives, identify problems, finding an effective approach and coming up with appropriate solutions.

Problem solving is crucial in ICT. You will encounter problems almost daily, many which will be different each day. Planning and having certain protocols or procedures against some of the possible problems can be very helpful and time saving. It is important to know how to recognize certain problems, not only in the infrastructure, but in the team, you work with too. These problems will be met almost daily in any ICT area due to the continuous changes in software, hardware and techniques used and it is important to do proper research on the recognized problems to solve them the best way possible. This semester was a lot of problem solving, especially because we had to do lots of testing and take into consideration several situations that could happen in our projects.

9.0 Future-oriented Organization

The organizational context of ICT assignments explores making corporate, sustainable, and ethical considerations and managing all aspects of carrying out the assignment.

It is important we consider ethical, corporate, and regulatory implications for every case study we tackle. As an infrastructure engineer, most likely, costumer data will be handled in many different scenarios. Some of this data can be critical which is why it is important to consider these aspects. Considering the risks and problems to be faced regarding security or project development is also very important to consider regarding corporate and ethical considerations. In addition, this semester we had to work within a budget because we were working with a client. It is important to take into consideration the different situations and possibilities for a client withing a budget.

10.0 Targeted Interaction

Determine which partners play a role in the ICT assignment, constructively collaborate and fitting communication aimed at achieving the desired impact.

As a group we considered all our skills and knowledge to plan and divided tasks accordingly to our best area and/or skills. We also took into consideration the time available weekly for each group member (ex. People who work and study). This way we divided tasks accordingly so they would be completed in the time established. Some problems can be met, but these are mentioned in the risk assessment and solutions were defined if someone had problems completing their tasks. I also consider that it's recommended that all the members participate in at least most of the tasks to learn and understand about the infrastructure. This way there will be no misunderstandings on what it is being done.

Case Study

The case study consists of working in a group of 3 people to improve in our teamwork, communication skills and professional skills in general, because it is more likely that in an IT environment we will be working in teams. The main goal of the project is to create an application, with a GUI that can deploy infrastructure to the cloud. The process must be as friendly as possible and mostly automated since we are aiming for a target group with little experience in IT. To achieve this goal, we are going to be using AWS cloud, automation tools such as Ansible, Terraform, Pipelines, etc. In addition, the team must also take into consideration certain requirements in the infrastructure such as security, resiliency, availability, and automation. By the end of the project, we should have a fully functional GUI application that will help us deploy web applications and/or other services.

Reflection

During the semester lots of challenges were given to us individually and as a group. Every challenge required different technical and professional skills that were put into practice. Individually, I have learned a lot about cloud services and about the most popular tools in professional IT environments such as Ansible, Terraform, resiliency and continuous deployment and integration. In the other hand, we as a group, have worked together very smooth. The team has good chemistry, and we took advantage of each of our strengths. We had great communication, and everyone worked accordingly.

Work Division

Javier Duran	**Alec Lacle**	**Angel Molina**

• Terraform Script • Route53 • CloudWatch Alarms • SNS • Load Balancer • Documentation • RDS • Ansible • GUI • Auto Scaling • S3 Bucket	• Documentation • Terraform Script • Web server • Ansible • Auto Scaling • Presentation	• Event Logs • Terraform Script • VPC • Documentation • VPN • On-premises • Ansible • Lambda • GUI • Monitoring Tools • Route53 failover • Flow Logs

Multidisciplinary Project

The multidisciplinary project consists of creating an automated infrastructure a client. The client is Aim, which is a platform that will be implemented to automate and assist in different school tasks such as registrations, course information, etc. The main goal of this project is to create an automated infrastructure for the developers of Aim. The team had to build the infrastructure according to the requirements and budget given by the client, also, taking into consideration security and resiliency. The infrastructure should be easy to use for developers and the documentation should contain step by step information on how to make use of it so the client can have the best working solution possible.

Reflection

The project was confusing at first, this was due to the lack of client meetings we had. After the second meeting, the requirements of the client were fully known, and we could work properly. Some challenges we faced was the budget given to us. The team came up with great solutions but most of them exceeded the budget. The team had excellent professional skills and through brainstorming and communicating between each other we came up with a reasonable solution, which are client accepted. In the other hand, we worked well as a group and understood what each team member's role was so that we could finish the work efficiently.

Work Division

Javier Duran	Alec Lacle	Angel Molina
• Terraform Script • Route53 • CloudWatch Alarms • SNS • Load Balancer • Auto Scaling • Documentation • RDS • S3 Bucket	• Documentation • Terraform Script • Web server • Email Server • Auto Scaling • S3 Bucket	• Event Logs • Terraform Script • RDS • VPC • Documentation • Route53 failover • Flow Logs

Conclusion

In conclusion, I think we can do better with a little bit more of planning because we are a very good group that understands and enjoys the career. During the first half of the semester, I can say that I have learned a lot, but I have noticed that I can learn so much more. For this reason, I believe that if I put a little extra, the group and I can come up with a very interesting solution for both case studies. During the second part of the semester, we worked harder because of the time constrains we were running into due to the lack of planning we had at the beginning. I met challenges throughout the semester. For example, some services that were learned were very difficult to 'master' took us a while to implement in the infrastructure without having to change our previous solutions. All in all, I think it has been a successful semester personally because we have achieved a professional solution for both projects. The solutions embed most, if not all, learning outcomes while putting into practice the professional skills such as leadership, problem solving and team communication.

References

AWS. (2021). *Amazon ECS.* Retrieved from AWS: https://aws.amazon.com/ecs/

AWS. (2021). *Amazon Relational Database Service.* Retrieved from AWS: https://aws.amazon.com/rds/

AWS. (2021). *Amazon Route 53.* Retrieved from AWS: https://aws.amazon.com/route53/

AWS. (2021). *Amazon S3 Buckets.* Retrieved from AWS: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

AWS. (2021). *Amazon Virtual Private Cloud.* Retrieved from AWS: https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/vpc-tkv.html

AWS. (2021). *AWS Lambda.* Retrieved from AWS: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

AWS. (2021). *AWS Step Functions.* Retrieved from AWS: https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html

AWS. (2021). *Identity and access management in Amazon S3.* Retrieved from AWS: https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-access-control.html

AWS. (2021). *Load Balancer.* Retrieved from AWS: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

AWS. (2021). *Policies and Permissions IAM.* Retrieved from AWS: https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

Bensky, B. (2020, July 14). *Terraform.* Retrieved from Fairwinds: https://www.fairwinds.com/blog/what-is-terraform-and-why-is-it-important

Kubernetes. (2021, Novemeber 23). *Kubernetes API.* Retrieved from Kubernetes: https://kubernetes.io/docs/concepts/overview/kubernetes-api/

Red Hat. (2019, January 8). *What is CI/CD Pipelines?* Retrieved from RedHat: https://www.redhat.com/en/topics/devops/what-cicd-pipeline

Red Hat. (2021). *How it Works?* Retrieved from Red Hat Ansible: https://www.ansible.com/overview/how-ansible-works

ServerCentral Turing Group. (2020, Oct 7). *5 best practices for resiliency planning using AWS*. Retrieved from AWS: https://aws.amazon.com/blogs/publicsector/5-best-practices-resiliency-planning-using-aws/

Yonan, J. (n.d.). *OpenVPN.*

Appendix

- Github Group 6 for case study: https://git.fhict.nl/I408431/casestudy-group6.git
- Github Group 6 for Multi-disciplinary project: https://git.fhict.nl/I408431/multi-disciplinary-group6.git