TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Lars Asi 178926IAAB

Creating a Laboratory Environment for the TalTech Course "Introduction to Cloud Technologies" Based on AWS

Bachelor's thesis

Supervisor: Siim Vene
MSc

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Lars Asi 178926IAAB

Labori keskkonna loomine TalTech õppeainele "Sissejuhatus pilvetehnoloogiatesse" AWS teenusepakkuja näitel

Bakalaureusetöö

Juhendaja: Siim Vene
MSc

Tallinn 2021

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Lars Asi

22.04.2021

Abstract

The aim of this thesis is to create a laboratory environment in AWS for the "Introduction to cloud technologies" course in TalTech for the teacher to successfully give lectures and practical assignments for the students and for the students to acquire everything they can from the subject and not create incredibly large bills for the school.

The first section of the work focuses on the problem that is to be solved with the thesis. A short description of the subject the lab environment is created for and the possible safety risks are given.

The second section of the thesis focuses on analysing the services available in the AWS environment and if the services meet the requirements. Services are chosen for the students to use with regards to the requirements. Services are also examined for the use of security, budgeting, monitoring and notifications.

The final section of the thesis describes how the environment is created and how it is usable for teaching. The first subsection covers the creation and implementation of the security policies and monitoring, budgeting, and notification configuration and how the users are added to the environment. In the second subsection the practical exercises are described.

This thesis is written in English and is 59 pages long, including 4 chapters, 5 figures and 0 tables.

Annotatsioon

Antud lõputöö eesmärk on luua labori keskkond õppeainele „Sissejuhatus pilvetehnoloogiatesse“ AWS keskkonnas, et õppejõud saaks efektiivselt anda tudengitele peale loenguid praktilisi ülesandeid ning pakkuda ka keskkonda kus ülesandeid lahendada. Koolile loodud kulud hoida kontrolli all ning tudengitele mitte luua kulusid.

Esimeses pooles kirjeldatakse turvapoliitikale vastava keskkonna üles seadmist, tudengite kasutajakontode loomist, keskkonnas toimuvate tegevuste logimist ning eelarve koostamist vastavate teadaannetega.

Teine osa tööst keskendub olemasolevate lahenduste uurimisele ning valitakse mis teenuseid kasutada ja milliseid mitte. Valitakse teenused millega hallata monitooringut ning logimist keskkonnas. Teenuseid mida kasutatakse õppetöö jaoks on eraldi nõuetena. Uuritakse ka viise kuidas kasutada parimaid turvapoliitikaid ning kuidas seada üles korrektne eelarvestamine AWS keskkonnas mis teavitaks rikkumiste korral vajalikke osapooli.

Kolmandas osas kirjeldatakse kuidas loodi tudengite jaoks mugavalt kasutatav ning õppejõu jaoks kergesti hallatav keskkond. Esimeses pooles kirjeldatakse kuidas loodi keskkond vastavate turvapoliitikate, loodi tudengitele kasutajad, seadistati keskkonnas toimuvate tegevuste jälgimine ja logimine ning eelarve vastavate teadaannetega. Teises alamlõigus käsitletakse tudengite jaoks loodud praktilisi ülesandeid detailselt ning lühidalt ka edasi arenduse võimalusi.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 59 leheküljel, 4 peatükki, 5 joonist, 0 tabelit.

List of abbreviations and terms

Auto Scaling Groups	Subservice of EC2 for creating a fault tolerant and highly available environment. Manages the health checks and provisioning of instances.
AWS	Amazon Web Services (AWS) is a subsidiary of Amazon providing on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis.
Bastion host	Instances for accessing other instances that do not allow SSH connections from the internet.
Buckets	Information containers in the AWS S3 service, similarities can be drawn with a hard drive or an SSD.
CloudTrail	Logging service for all the management actions, creates, deletes and modifications, in the AWS service. Does not log the actions inside a EC2 instance.
Control Tower	Service provided by AWS to configure a landing zone more quickly but with some restrictions.
Elastic Compute Cloud	Elastic Compute Cloud (EC2) service is for creating and managing virtual servers/instances.
Guardrails	Governance rules that can be enabled on an organizational unit to enforce policies or detect violations.
IAM	Identity Access Management (IAM) service which is used for managing the users of the account, their associated groups, policies, and roles.
Landing Zone	A well-architected, multi-account AWS environment that is scalable and secure
Organizational Unit	An Organizational Unit (OU) holds AWS accounts within a root account of an organization.
Policies	Security measures to give or deny access to all services in the AWS environment.
Private IP	IP addresses used to communicate inside the AWS environment, can not be accessed from the outside.
Public IP	Outward facing IP address that can be used to access hosted websites or connect with instances.

Relational Database Service	AWS database service for creating and managing Relational databases, abbreviation is RDS.
Roles	Security measures to allow services to take on certain rights that allow or deny them to communicate with other services.
Security Groups	Stateful firewall rules for EC2 instances.
Simple Notification Service	The Simple Notification Service (SNS) allows users to create notifications for events and sends the notifications via SMS, html, or e-mail.
Simple Storage Service	The Simple Storage Service (S3) is for storing static files and hosting static websites.
Tag	Attachable metadata for AWS resources to manage, identify, organize, search for, and filter resources.
Topic	Subservice of SNS where groups or topics are created which include users to be notified.

Table of contents

1 Introduction 12
1.1 Subjects’ introduction 12
1.2 Services to be taught in the subject. 12
1.3 Possible safety issues 13
1.4 Choice of methodology 13
1.4.1 Analysing and comparing the best practises. 13
1.4.2 Modelling 13
1.4.3 Developing a prototype 14
2 Solution analysis 15
2.1 Requirements for the lab environment 15
2.2 Possible implementations of the lab environment 15
2.2.1 Students create their own accounts. 16
2.2.2 Everything in one account 16
2.2.3 Landing zone with multiple users 17
2.2.4 Landing zone with multiple lab environment accounts 18
2.2.5 Choosing the implementation 18
2.3 Choosing how to set up a landing zone. 19
2.4 Possible ways to create users for students in the environment. 19
2.5 Billing management 20
2.5.1 Pay-per-second. 20
2.5.2 Pay-per-request. 20
2.5.3 Pay-per-storage-unit 20
2.5.4 Hybrid payment models 21
2.6 Budgets in the AWS environment 21
2.7 Minimizing cost generation. 22
2.7.1 Services costing the most. 22
2.7.2 Ways to mitigate costs. 22
2.8 Gathering information on the topic. 23
2.9 What is a landing zone inside an Organizational unit and why is it used? 24

2.10 Choosing services to create the environment. 24
2.10.1 Control Tower 25
2.10.2 IAM 25
2.10.3 IAM Policies 25
2.10.4 CloudTrail 27
2.10.5 SecurityHub 28
2.10.6 GuardDuty 28
2.10.7 CloudWatch 28
2.10.8 AWS Config 28
3 Implementation of lab environment 30
3.1 Creating the lab environment 30
3.2 Implementing security policies 30
3.3 Creating security policies 31
3.4 Creating users for students 32
3.5 Implementing event logging 33
3.6 Budgeting the environment 34
3.7 Implementing cost minimization 35
3.7.1 Managing resource types 35
3.7.2 Managing instance's uptime 35
3.8 Practical labs 35
3.8.1 First exercise 35
3.8.2 Second exercise 36
3.8.3 Third exercise 36
3.8.4 Fourth exercise 37
3.8.5 Fifth exercise 38
3.8.6 Sixth exercise 39
3.9 Lessons learned. 40
3.9.1 Future improvements 40
3.9.2 AWS restrictions 41
3.10 Cleaning the environment 41
4 Summary 43
References 44
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis 46

Appendix 2 - IAM policy condition with service specific variable 47
Appendix 3 - IAM policy condition with AWS variable 48
Appendix 4 – Automatically creating users in AWS 49
Appendix 5 - Macro for sending emails 51
Appendix 6 - Lambda function to stop running instances after an hour 52
Appendix 7 - Second practical exercise 53
Appendix 8 - Third practical exercise 54
Appendix 9 - Fourth practical exercise 55
Appendix 10 - Fifth practical exercise 56
Appendix 11 - Sixth practical exercise 58

List of figures

Figure 1. Visualisation of the created landing zone. 34
Figure 2. Visualisation of the third exercise. 37
Figure 3. Visualisation of the fourth exercise... 38
Figure 4. Visualisation of the fifth exercise. 39
Figure 5. Visualisation of the sixth exercise.. 40

1 Introduction

In this thesis the issues which would occur if there were no correctly managed and monitored lab environment for the course are analysed. All the services with the necessary functionality provided by AWS are examined to build the lab environment. Analyse the implementation of the lab environment and the practical assignments assigned for the students.

1.1 Subjects' introduction

Cloud computing is moving closer to being the norm for most companies in the world with AWS holding the majority of the market share, but it is still not being taught at TalTech. Requirements for creating the environment and actions that are allowed are given by the teacher of the course.

1.2 Services to be taught in the subject.

A student studying cloud computing needs an environment where they can practise all that they have acquired in class, but the environment should also be flexible enough to enable the student to learn by themselves but enabling all the services for the students is not acceptable because there is a possibility that the student forget to turn off or disable the created resources or misconfigure resources to be less cost efficient and create an enormous bill for the school.

Since AWS is used mainly to set up cloud-based servers then that must be included in the services allowed for the students. The main service, which is used for managing single server environments, creating, duplicating, backing up and restoring servers can also manage multi server environments with automation and load balancing is the EC2 (Elastic Compute Cloud) service and it uses subcategories to create a more user-friendly experience. The services that are going to be taught in the course are EC2 for server management, S3 for storage, VPC for networking and CloudWatch for monitoring.

1.3 Possible safety issues

The Safety issues must be considered for both parties involved meaning the safety of the students and the teacher/school must be guaranteed. Ensuring safety means that there are no bills created for the students and no unexpectedly large bills for the university, the personal data of the students is kept secure including their banking (credit and debit card) info as well as the banking info of the university and or teacher.
The protection of logs is equally important as they can be used to find students trying to take credit for others work. Without the log messages generated by events in the environment protecting the users and finding what activities were done by whom becomes impossible.

1.4 Choice of methodology

While creating a scalable, secure, monitored, and easily manageable environment for the cloud computing subject, 3 different methods of conduct have been chosen. The methods focus on studying the available information, creating visual models of the created work and creating different environments and practical exercises to test the functionality of the built environment.

1.4.1 Analysing and comparing the best practises.

Studying different classes and webinars on the topics needed to create the environment. Primary sources of information are the Udemy and ACloudguru learning environments and appropriate courses on the topic [1], [2].

1.4.2 Modelling

Modelling is required to get a better understanding of the environment and all the different services working together in the landing zone. The model is necessary when defending the thesis to clarify different processes, give an overview of the permissions and the overall flow of the information inside the landing zone. Models are created for students' practical exercises as well to give them a better overview of the exercise.

1.4.3 Developing a prototype

Two types of prototypes are developed. First focuses on the environment from the administrative side and the second prototype represents students' point of view. Initially the environments are created through different means to test the security, monitoring, budgeting, and scalability. After that, following the set requirements, practical exercises are created to test the environment and the correctness of all the limitations on the services, budgets, monitoring and permissions.

2 Solution analysis

In this chapter the different possibilities to solve the different problems are analysed. Populating the environment with student users through different services. How to configure the budgeting and cost minimization of the lab environment. Which services should be used to create the environment.

2.1 Requirements for the lab environment

Before the environment can be built, requirements must be put in place to guide the creation of the environment.

- The environment needs to be able to scale up to thousands of students if need be. To achieve this the adding of the students to the environment with the proper permissions needs to be automatic.

- Security to protect the personal info of all the users and their banking information is necessary. The resources created by the users must be secure as well meaning that users can not manage other users' resources.

- To find any and all attacks against the environment and its users or the malicious use of the environment or the misconfiguration of other users' resources monitoring of all the actions taken inside the environment and if possible, resources must be enabled.

- In case irreversible damage is done to the environment then the environment must be easily disabled or deleted, and a new environment must be created with minimal effort and minimal downtime for the users.

2.2 Possible implementations of the lab environment

Building a lab environment can be approached from different angles. One way is for each student to create their own AWS account either through normal means or apply for an

AWS educate account. Another way is to create an AWS account and add all the students to the account as users and manage their accesses. The third way is to create a landing zone which has an account where the users for the students are created and additional accounts for accumulating security and auditing information. Fourth way is to create a landing zone which creates the security and auditing account, and personal accounts for each student are created.

2.2.1 Students create their own accounts.

Creating an AWS account requires an email and a credit or debit card for online payments. It is also possible for students to apply for the AWS Educate program, which gives them free credits to spend to cover the first costs that may arise while studying on the platform [3]. The main purpose of the AWS Educate program is to give students a chance to practise using the AWS services for the first level certificate which is the Cloud Practitioner certificate which in turn happens to be the certificate that the "Introduction to cloud computing" course is preparing the students for.

Following this approach, the students would have free reign over the AWS environment and could do as they pleased without asking for additional permissions. From the course perspective plagiarism would in some cases be impossible and in some very trivial.

The downside of the approach is that helping the students troubleshoot their problems becomes exceedingly complicated and there is a possibility that the student may break the environment by deleting certain resources or allowing access to bad actors who may get access to the credit card info used to create the account. The possibility of creating enormous bills for the students is quite high when factoring in the assumption that the students are not adept when it comes to cloud technologies. In addition, AWS does not alert the user about the potential costs if no billing alerts have been set up beforehand. With AWS Educate, the students' chances of exceeding the budget are lower but not eliminated completely. Additionally, the AWS Educate service has preconfigured learning paths that do not match with the scope of the course.

2.2.2 Everything in one account

One possible way of setting up the lab environment is to create a regular account in the AWS environment and add the students as users to the account. The users would have to be managed by custom policies, billing alarms would have to be set up for the account and all the AWS management actions, creating, deleting, and modifying resources, would have to be logged using CloudTrail. While creating the lab environment is similar to the third approach the approach of creating everything into a single account poses serious security risks.

When everything is in one account then the chances of a student or foreign actor getting complete control of the environment does pose a serious threat. The bad actor could steal the credit card info used to pay for all the account, they could disable all the alerts and logging services while also deleting previous log messages to render debugging the problem and finding the intrusion mechanism impossible. Using one account for everything can cause issues when students try to test new services and learn independently but during the process, they configure resources wrongly or even delete some existing resources for example deleting a public subnet from the VPC mainly used for the class renders all the instances inaccessible.

2.2.3 Landing zone with multiple users

Another approach to creating the lab environment is to create a landing zone inside AWS. The schools account is given the role of master account for the landing zone and it manages the security, auditing and lab environment account created inside the landing zone. The accounts are placed inside the Opilased Organizational Unit for ease of management. The three created accounts are distinct and separated from each other which means that if one account is compromised the rest are not affected and the compromised account can be dropped from the organization. All the management logs produced by CloudTrail are sent to the Security account and the billing data is sent to the Auditing and master account and if need be, the logging and auditing data can be redirected to other accounts.

The main issue with this approach is that the students can copy each other's work and delete instances, buckets and even other users if the permissions are not managed correctly. The account feels overcrowded and finding specific resources inside services becomes difficult as more and more students provision resources using different services. The approach also limits the full use of the billing service which enables billing actions to stop instances or even disable a user or account policy for exceeding the allowed budget. With the third solution the only usage of the budget actions would be to strip all the users of their access to the lab environment when the budget has been exceeded.

2.2.4 Landing zone with multiple lab environment accounts

The fourth approach is similar to the third in the way of setting up the landing zone, managing the flow of security and audit data and setting up the accounts in the Students Organizational Unit. The access for the students can be managed through IAM inside the lab account or by using the AWS SSO service to create all the login info for the students which is attached to accounts inside the Students OU, one for each student.

With this approach it is possible to create a script which creates a budget alert and budget action for each account which when triggered takes away all access from the student. Shutting down the instances is not possible because the instances have to be created before the billing action.

The main downside of the approach currently is that it is not scalable, each account created for the students must be manually accessed from the master account and a user must be created inside the newly created account.

With this approach the possibility of the students acquiring administrator access is possible which they can use maliciously or do irreversible damage to their account, both would result in the deletion of their account and if allowed the student would have to start from scratch.

2.2.5 Choosing the implementation

The first approach gives the students the same user experience as does the fourth approach but troubleshooting error messages becomes incredibly difficult and the chance of students creating enormous bills for themselves or the school is too high. The second approach, everything in one account, has too many vulnerabilities for it to be considered a safe environment for the lab environment although it is scalable. The third approach, Landing zone with multiple users, has minor issues considering the workflow of the practical assignments for the students but those issues can be mitigated or at least minimized using IAM policies, and the solution is also highly scalable with the biggest issue being the distribution of user information for the students. The fourth approach, Landing zone with multiple lab environment accounts, does have the best user experience and practical assignment integrity but it is currently not scalable and may generate the school the biggest bills among all the approaches.

With the aforementioned taken into consideration the third approach is the most viable meeting all the requirements of safety, security, budgeting and scalability.

2.3 Choosing how to set up a landing zone.

A Landing Zone is collection of accounts inside an OU (Organizational Unit). Creating a Landing Zone helps the client secure their multi account environment in the cloud. The available options are immense but takes up a tremendous amount of time and requires a great understanding of current services and options inside the AWS environment and overall experience with sys or devops.
Manually creating accounts is not the only way to set up a Landing Zone. The Control Tower service is an easier way to implement a Landing Zone more quickly and with less technical expertise, but it does come with some certain limitations.
The main limitation that exists in Control Tower is that there is no way to transfer an existing Landing Zone over to a Control Tower. The main limitations that apply to the current situations are that Control Tower lacks support for CI/CD, Interactive API-s, the use for CloudFormation templates and Terraform modules [4]. These are advanced topics that are not taught in the course. Control Tower also has self-service provisioning which enables the students to register to the subjects landing zone with a url provided by the teacher. Without this option the teacher has to create every student an account manually.

2.4 Possible ways to create users for students in the environment.

There are currently two different ways to automatically create users for the students, first is using the AWS SSO service which integrates with on-premises domain controllers or can be used with SAML to give the service the students data to create the users. Another way of adding users to the environment is adding them manually using the IAM service or doing the same task programmatically using the AWS CLI. The AWS environment provides all accounts with a CLI capable of performing most of the tasks that can be done in the GUI. The CLI understands the most popular programming languages for example python and java and can be given a script for creating users, their login profiles and assigning them to a group.

2.5 Billing management

AWS services and resources generate bills mainly through the three following categories: pay-per-second, pay-per-request, pay-per-storage-unit.

2.5.1 Pay-per-second.

The pay-per-second model is the most marketed user-friendly feature for users because they only pay for what they use. The payment model is used mainly by the EC2 and the RDS (Relational Database Service) services where instance work times increase the bill with the maximum accuracy of one and the minimum of sixty seconds, the load on the instances or requests they handle from other services does not contribute to the bill, the users are billed the same amount whether they put the instances under the maximum possible load or none. The size or power of the provisioned instance is what determines how large a bill is generated each second [5].

2.5.2 Pay-per-request.

Paying per requests is mainly used by services that send commands or receive requests from AWS services or from the internet examples being SNS and SQS. SNS sends notifications after being triggered by CloudWatch alerts for example, SQS stores and sends messages after being requested by different services. Each request costs the owner of the account.

2.5.3 Pay-per-storage-unit

AWS S3 and other database services generate profit for AWS using pay-per-storage-unit meaning the bigger the files placed inside AWS the larger the cost. Pay-per-storage-unit payment models use built in functionality to generate an even bigger bill, one example being the encryption of files where users can use AWS Key management service to create private keys meant for encrypting and decrypting files, AWS does the encryption of the files automatically but every encryption costs. The cost of the storage and retrieval of the files differs when using different storage options for example archiving a file in S3

Glacier will cost less per gigabyte compared to S3 Standard, but retrieval fees are added [6].

2.5.4 Hybrid payment models

These are services that integrate multiple payment models into their system with the best example being S3 which generates a bill for every gigabyte stored on the service and if S3 has static website hosting enabled it in turn generates AWS revenue based on each request that the service handles.

2.6 Budgets in the AWS environment

AWS has created the service AWS Budgets for creating and managing budgets. The created budgets can monitor the entire environment, or they can be fine tuned for example only oversee the costs of a single instance inside a specific region. Budgets can be created in two different operating modes, actual or forecasting. An actual budget alerts the user when the specified amount or the percentage of a certain amount has been exceeded. A forecasted budget alerts the user when a resource has been provisioned that will exceed the budget or a percentage of the budget at any point in time of the current month [7]. AWS Budgets is integrated with the SNS service which sends out the alert information.

The service has the functionality to take certain actions when a budget is exceeded. First type of action is IAM policy which can be set to change users' permissions when the budget has exceeded the limit. The second is attaching a service control policy to an OU to limit the actions Accounts inside the OU are able to perform or attaching it directly to an Account. The final budget action is stopping specific EC2 or RDS instances, this budget action can only be deployed on instances that have already been created.

The aforementioned actions have great use cases in a business environment but provide little in the context of the subject. The most useful budget action for the subject is changing the permissions of the students when the budget has been exceeded but with the solution of creating all the users in one account inside the landing zone it is not possible to effectively track the money spent per student.

AWS Budgets works in unison with AWS Billing and AWS Cost Explorer which both help to get a better overview of the costs and sort and group the cost data by different metrics for example grouping costs by service.

2.7 Minimizing cost generation.

After applying different billing alarms, the services, from the presented curriculum, which are going to cost the most have to be verified and find ways to minimize the cost of the AWS service.

2.7.1 Services costing the most.

Of the services going to be taught in the subject the EC2 service is going to be the costliest following with the CloudTrail service. EC2 will be the most costly because each student creates many instances especially in the last practical exercise where the students use Auto Scaling Groups to create multiple instances at the same time with the cost of per-instance per-hour is at 0.0116$ for an Linux t2.micro type instance [8]. CloudTrail is the second most costly service because it records every action of every student and sends the data to an S3 bucket in the security account which if placed in the same region is not going to generate extra copying fees and the CloudTrail service costs cents for every 100,000 events [9]. The other services and configuration tools generate such a low bill for the subject that it is not necessary to take them into consideration.

2.7.2 Ways to mitigate costs.

EC2 instance costs are calculated by the computation, network speed and size of the instance and how long an instance is running for hence the size and run time of the instances must be managed. Management of EC2 instance sizes can be done through IAM policies which allow the use of conditions to only be able to create instances with a certain type or using AWS Config to create rules to automatically change the types of created instances. For the practical assignments the students are allowed to use the instance types of t2.micro which have one virtual CPU and one GB of RAM which is enough to comfortably, without significant lag, complete the practical assignments. With the cost of the instances being fixed the management of instance run times can be done through these three different ways: manually shutting down instances the students forgot to turn off every night, not during the day as it might hinder their work, creating a cron job which triggers a lambda function meant to stop all the instances at midnight, creating a rule inside CloudWatch which triggers a lambda function meant to stop all running instances after a certain time interval for example stopping all the instances that have been up for more than an hour.

Automatically controlling the instances and stopping them after a certain time interval is the best choice because it reduces manual work and will generate less costs than stopping all the instances only at midnight.

The CloudTrail service does not have valid ways of managing costs since the costs are generated by the students using the environment and limiting the hours a student can access the environment is not going to significantly affect the cost. CloudTrail is a region based service which means using AWS services in regions CloudTrail has not been set up would mean that the service would not generate any costs but that in turn would render the logging done by CloudTrail in the main region worthless since it would only log the students switching regions.

2.8 Gathering information on the topic.

Collecting the necessary information from various sources is quite troublesome because although there is a lot of information available on the internet asking the correct questions is often the main issue when solving different problems. The official AWS documentation is very helpful at the start but as the problems got more specific to the problem presented by the thesis the documentation turned out to be too generic or too focused on simple use cases, basically focused towards the beginners.

Different webinars provided a very well-balanced foundation for the AWS environment but they mostly focused on teaching what a service is and what it is used for with some small challenges or exercises and mostly neglected to teach how to debug various issues that may arise in creating policy permissions or automation for instances.

Conversing with different experts proved to be incredibly useful since they have a solid understanding of the AWS environment and have experienced many of the same issues and bugs. Found various experts on the subject with the help of my instructor and through the many connections made during the time in ITÜK.

2.9 What is a landing zone inside an Organizational unit and why is it used?

Control Tower is an automation tool for setting up a multi account environment with the account that the landing zone is set up in becoming the master account of all the child accounts.

Landing zones are used to ensure safety and data integrity for the accounts that is why upon creating a landing zone with the Control Tower service there are three different accounts generated automatically. One is for security and compliance, second one is for auditing purposes where all the billing data is to be collected and third is a regular account where development can take place, adding additional accounts is to be done in the master account under organizations.

Landing zones are created inside organizational units to ensure that accounts belonging to different landing zones do not get mixed up and are not configured wrongly.

2.10 Choosing services to create the environment.

AWS has multiple services meant for security and surveillance, most are for managing the access and/or logging of a single service or for aggregating all the data into one place to give an overview of the entire environment.

Services used for securing the environment are as follows: IAM for managing users, groups, roles and policies meant to manage access in the environment, Control Tower which is used for creating and managing a landing zone, CloudTrail is for tracking all the actions taken by the users in the environment, SecurityHub for aggregating all the data of all the enabled security services into one place and analysing them to find if AWS best practises have been used, GuardDuty is an AI as a service that finds odd behaviour in the environment and warns when for example a EC2 is farming bitcoin or root user has been accessed, GuardDuty also finds less threatening actions for example granting public access to a S3 bucket, CloudWatch to give a visual representation of the data for ease of analyzation and for periodically triggering lambda functions, AWS Config can be used to create rules for all the users in the account to follow

The services used for the implementation of the lab environment are as follows: Control Tower, IAM, CloudTrail, SecurityHub, GuardDuty, CloudWatch, AWS Config.

2.10.1 Control Tower

The AWS service meant to automate the process of creating a landing zone. Control Tower creates a security, audit, and a shared services account inside an OU in the master account, existing accounts can be added to the landing zone after Control Tower has finished creating it. In addition, Control Tower is linked with AWS SSO which is responsible for managing user permissions inside multiple sub-accounts of the master account. Users can be added manually or by connecting an existing domain controller to the service. In Control Tower Guardrails created by AWS are automatically enabled that follow AWS best practises.

2.10.2 IAM

IAM (Identity and Access Management) is an AWS service meant for managing users, groups, policies, and roles within an AWS account. AWS best practises suggest adding users to groups meant for distinct activities i.e., developer and admin, the groups in turn have policies attached to them, either created by AWS or the administrators of the environment, which give the users inside the groups access to or deny them certain services. Roles in layman terms are users for services - for example, a rule allowing administrative access to the S3 service can be attached to an EC2 instance which, when communicating with an S3 bucket assumes the attached role. This allows access to the S3 buckets and to manage their content, to name a few possibilities. Without attaching the role, the EC2 instance receives a permission denied message.

2.10.3 IAM Policies

Policies are written in json with the mandatory parameters of version with a certain date to give AWS information on which rules to follow and a parameter of statement which houses all the information for the policy. In the statement parameter permissions are separated into groups having 3 required parameters, to allow or deny services, and permits the use of 2 extra parameters which are used for conditions and naming the groups. The three required parameters being effect, action and resource with resource being changeable with principal only in specific use cases which is not tackled in this work and the extra parameters being Sid and condition. Action, resource, and principal are interchangeable with notAction, notResource and notPrincipal to allow all the actions excluding the ones inside the aforementioned parameters, usage of the parameters is considered advanced and is not used to give a clearer view of the allowed services.

The effect parameter can be either “Allow” or “Deny” which if attached to a user either through a group or directly allows or denies the use of the services mentioned in the action parameter [10].

Inside the action parameter the actions are stored inside a list separated by commas and follow the pattern of service and performable action for example to allow the action of starting a stopped instance the action would be the following “ec2:startinstance”. Inside AWS the actions are grouped mainly by the services. List and read actions allow the users to visually see the data in the console, tagging allows the user to add or delete tags, write is used for changing the configuration, creating and deleting resources, for EC2 there is one more group named permission management which fundamentally is the same as write and for most of the other services in AWS is inside the write group [11].

The resource parameter allows an extremely fine-grained and simple method of managing the resources an action is allowed to access. Syntax for the resource parameter has 2 mandatory fields, arn and aws, with the rest being optional and can be left blank or marked with wildcards in the form of an asterisk. After the aws the pattern is as follows: service to be used, region of the service, not all services are region based therefore being marked with an asterisk in some cases, account id, if left empty the users can access services in other accounts but they would have to be given access beforehand from said accounts and the last field is service specific for example if the users are only allowed to access the S3 bucket named production-webpage then the resource parameter would be the following "Resource": "arn:aws:s3:::production-webpage". Resources can be inserted as a list separated by commas. The resource parameter allows the use of variables inside the resource which can be used the limit the access even further [12].

Optional parameters for the statement are Sid and conditions. Sid is used to name a group of the policy to ease the reading of the policies. Conditions can be applied to all services but not all services can be controlled by all conditions. Some actions can be limited with tags or have distinct conditions meant only for them, example of that is used for creating an EC2 instance with only the allowed type of t2.micro in Appendix 2 , all services allow the use of AWS variable conditions as shown in Appendix 3 [13].

2.10.4 CloudTrail

CloudTrail is the service responsible for tracking all the actions the users make while using the lab environment. The service enables compliance, governance, operational auditing, and risk auditing. The service works by capturing and recording all the activity as CloudTrail events that can be viewed and downloaded in the CloudTrail event history or stored inside buckets using the S3 service in the account that created the logs or sent to other accounts, in our case the data would be sent to the security account for added safety. AWS is constantly integrating services and has integrated CloudTrail event history with S3 and Athena with the latter providing a sequel query based environment to query the event data and find specific events done by certain users in certain timeframes more quickly and easily.

The main use of CloudTrail is inactive student monitoring meaning the service is always logging information, but the information is not accessed until any malicious behaviour is recognized with the help of CloudWatch and GuardDuty which is covered in detail in the following section or when students report that their resources have been misconfigured or terminated without their prior knowledge.

Lambda has as well been integrated with CloudTrail and can be used to trigger rules during certain events which in turn send SNS notifications to alert the required parties. Use case of this integration would be to catch creation of resources not permitted by the environment parties which in the subjects' case are restricted by IAM policies but since the teacher wanted to have a lab teaching the students the creation of custom IAM policies there is a risk that a student creates for themselves administrator access policies and creates not allowed resources which in turn may generate enormous costs for the school. To solve the problem the actions of the students are highly monitored during the practical exercise and the rights are revoked after the deadline for the practical exercise.

2.10.5 SecurityHub

The service is created to help users secure their environment better by following well known security standards like PCI DSS (Payment Card Industry Data Security Standard), CIS (Centre for Internet Security) and AWS own best practises. The service constantly scans for changes in the environment and gives warnings of misconfigured resources and assigns them a severity level and gives an overall safety rating for the entire environment [14].

With the lab environment the best use of the service comes at the very beginning when users have not been allowed to use the environment yet because the labs will trigger many alarms mainly through allowing resources to communicate with the internet.

2.10.6 GuardDuty

AWS service that uses artificial intelligence to find suspicious and hurtful events inside the AWS environment. All the events are logged inside the service and can be connected with CloudWatch and SNS to send notifications when suspicious behaviour has occurred. Best use cases for the lab environment would be for example GuardDuty finding instances that are being used to mine bitcoin or Ethereum or informing the administrator when somebody has accessed the root account of the environment [15].

2.10.7 CloudWatch

Solution created by AWS to help users monitor their environment and set up alarms or rules in AWS. When the alarms are triggered, they can perform actions on the various resources in the environment, call lambda functions either periodically or when a value is exceeded for example an instances CPU is under 90% load and notifies certain users through SNS topics [16].

2.10.8 AWS Config

The service is meant to record the changes in the configuration and can be used to audit the environment. Notifications to changes can be set up with the help of the SNS service. The service is per-region but can be aggregated between regions or even accounts. To enable AWS security best practises the log messages created by the service are aggregated to the security account [17].

AWS has created many rules to help with the managing of the account and custom rules can be created using Lambda for example finding instances that do not use the type t2.micro. Rules can not only notify when they are broken but can also manage the resources monitored in the rule as well for example when a student creates a t2.large instance the instance type is downgraded to t2.micro. Rules can be evaluated for each config change and/or at regular time intervals.

3 Implementation of lab environment

This section describes how the lab environment for the introduction to cloud computing subject is created. It covers how the practical exercises are created. The final points in this section cover ideas for future improvements, the restrictions imposed by AWS, how to lift the restrictions and how the environment is going to be deleted and why.

3.1 Creating the lab environment

AWS Control Tower service enables creating the landing Zone as a simple task that requires few actions. The creation of two additional email addresses is required for the logging and auditing accounts. The creating of the additional emails turned out to be the most difficult part of the entire process, because the help of the TalTech helpdesk is needed to create email addresses within the TalTech domain but the request is ignored. The necessary emails are created in gmail. The creation of the landing zone which is done entirely by AWS took about 1 hour to complete.

3.2 Implementing security policies

Creating different permission policies that allow the user to only use instances, databases, load balancers etc that they themselves have created is complex because AWS is built to help businesses grow and no business sees value in protecting the servers that workers have created to test something and only policy those servers that generate value because the policies the administrator should apply to a production server can be found under the EC2 IAM examples [18]. To meet the requirements permission policies are created that give the users only access to their created resources. Custom permission policies are needed for the following services: EC2, S3, Backup, RDS, Networking & Content Delivery and CloudWatch, everything else is blocked for the student.
Managing access to different AWS resources can be performed through four different means, first being management through tags, second being assigning access for users to only use certain resources, the third being allowing certain actions while the user is performing another action for example adding tags to instances is only allowed when

creating a new instance, managing the tags of already created instances is not allowed because it would allow students to take credit for other students work. The fourth course of action is allowing certain actions to only be taken by certain services.

3.3 Creating security policies

After the landing zone is created the next step is to create custom security policies which grant the students permissions to use the AWS lab environment. The teacher set the overall requirements which are fine tuned as the practical exercises are created, one of the requirements is that the students must see everything in the allowed services which in turn saved a huge amount of time creating the policies because the use of wildcards for List and Read actions is allowed.

The creation of the policies is simplified in part by the AWS policy creator since all the actions are listed but it does get overwhelming due to the sheer number of actions allocated to a service with the EC2 service being the most complicated in this work because it also entails all the networking actions. A major encumbrance is that the actions do not always follow a certain naming convention for example the action for creating security groups is “ec2:createSecurityGroup”, for creating a S3 bucket the action is “S3:CreateBucket” but the action for creating a EC2 instance is“ec2:RunInstance”. Another obstruction is again the inadequacy of the naming convention because the subservice the action manages is often not included in the name of the action for example if a user wants to change the volume or the security group of an instance they have to have the action “ec2:ModifyInstanceAttribute” allowed. All the policies created for the thesis are in GitHub in the policies folder [19].

After achieving main functionality in the environment managing the conditions of services is next. The teacher agreed to use tags to identify the resources of users with the format of User: <username> and it allowed the creation of the condition that if a user wants to manage an existing resource, then the tag on the resource has to refer to the user. On certain resources the condition that a tag must be entered can be applied but it will create permission errors with resources with lots of dependencies with no helpful error messages. The only service that does not allow to use tags in the desired way in this work is S3. The substitution for tags is using the student’s username as a prefix in the S3 bucket’s name. Conditions allow the management of resource types as well for example allowing the users to use only a certain type of instance, this is covered in implementing cost minimization.

One more use case for conditions is the allowing of certain actions only when being executed by a certain service example would be creating a spot instance which requires the action of "iam:CreateServiceLinkedRole".

To monitor the students' activities better the condition is implemented that only allows actions to be taken inside the us-east-1 region because CloudTrail and VPCs' are region based and grading the students work across multiple regions will become strenuous.

The created policies are attached to the group Students and all the students are added into said group. The students are also added to the IAMBasicPermissions group to be able to complete the second practical exercise.

3.4 Creating users for students

Creating the users with AWS SSO using SAML is not successful. During the creation process the option to use SAML is not presented but the documentation still presents it as a viable source [20]. Importing the user data from the schools domain controller is not attempted because communication with the help desk is incredibly slow and inefficient.

The users are created for the students using the CLI and the IAM service. The user data for all the students that registered for the subject is acquired in csv format from Moodle and a python script is written to generate commands to be pasted into the AWS CLI, the program code is under Appendix 4. The approach is chosen over creating a script that does everything by itself because the Author at the time did not have the expertise on how to generate the commands and save the passwords used in the commands to create users into a S3 bucket so that they could later be distributed to the students.

Because the school is on lockdown the user data has to be sent for each student separately. For that the data with the students emails is gathered into a google sheet file and a macro is written that sends the user login data to every student individually, the program code for the google sheet macro is in Appendix 5. The macro worked when sending data to one student at a time but when sending the data to all the students at once the emails got stuck in Google’s spam filter or the schools email filter. The login information is given manually to everyone.

3.5 Implementing event logging

To configure proper logging and auditing data be sent to the security account the services responsible for logging and auditing, GuardDuty, CloudTrail, Security Hub and AWS Config, need to be enabled. The services, except for AWS Config, set themselves up and merely provide insight into the environment to the users, they do not stop instances for example, but their data can be used either through CloudWatch and Lambda to stop said instances. AWS Config logs the findings and has the availability to configure rules which if triggered can respond, for example in the context of the subject when a student launches a type of instance that is not a type of t2.micro then AWS Config can be configured to automatically change the type to t2.micro. Except for changing the types of instances AWS Config did not provide anything else beneficial and is not implemented.

To ensure the safety of the logs of all the aforementioned services sending the data to the security account is configured manually. To send the data to the destination account the services must be enabled and the policies of the S3 buckets where the data is stored is updated to allow access from the account sending the data. CloudTrail in the sender account is enabled and the storage location is the security accounts S3 bucket.

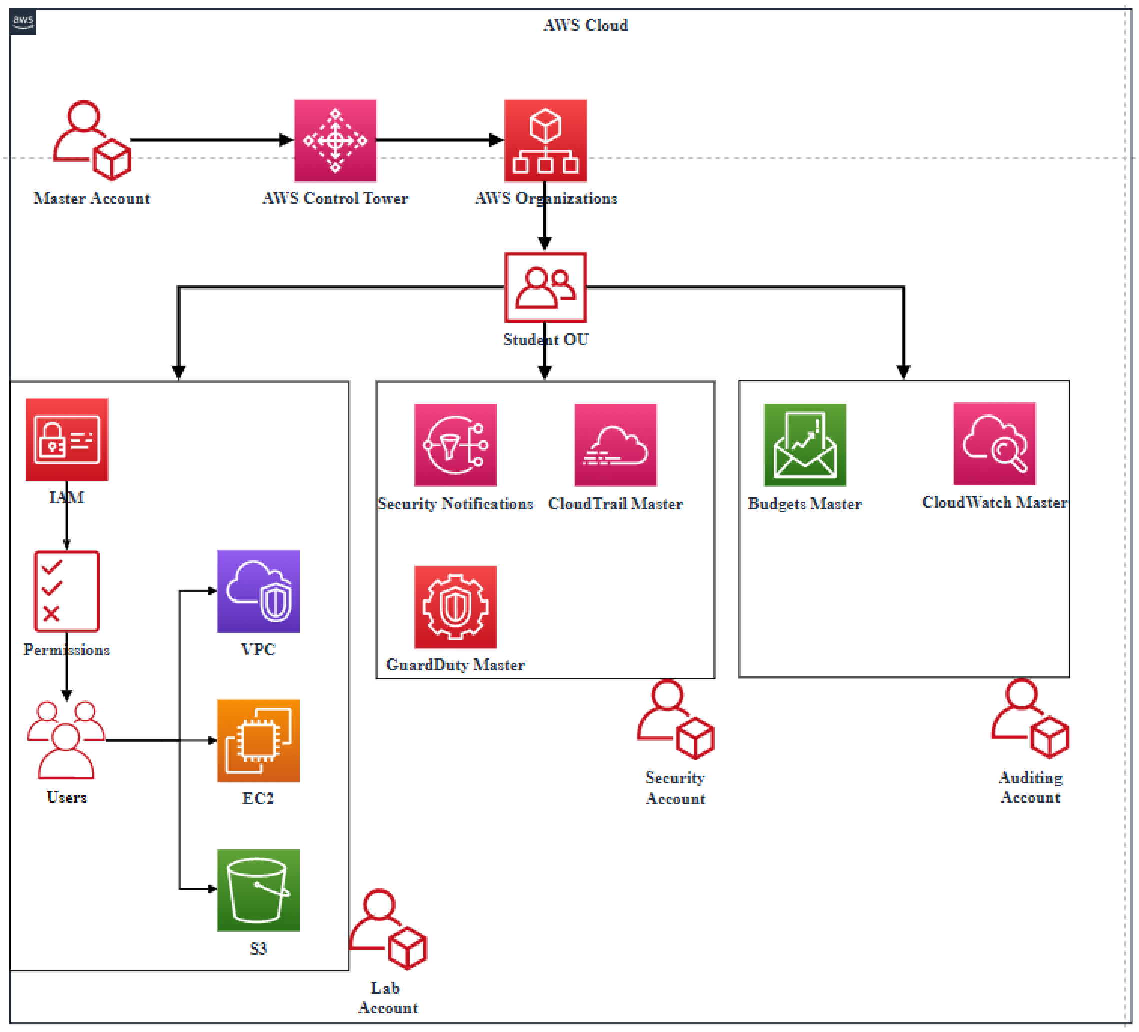

Figure 1. Visualisation of the created landing zone.

3.6 Budgeting the environment

Through AWS Budgets a forecasted budget is created with a SNS alert to notify the required parties through email when the budget is exceeded. This type of budget is chosen to help find anomalies or unpermitted costly activities in the environment.

Cost explorer is used to monitor daily costs and the services that generate the most cost. A misconfiguration that is done while setting up the environment increased the costs but went unnoticed because at the time students are working on the third and fourth practical exercise and the Author and the teacher are aware of the costs but Cost Explorer showed the costs to be of the same service where the misconfiguration happened. After implementing instance up time rules and the costs remaining unchanged the teacher finds the resource generating the costs which is a misconfigured snapshot.

3.7 Implementing cost minimization

Instances generate cost based on the size and run time of the instance hence the first step is to manage the size of the instance the students can choose and secondly the run time of said instances.

3.7.1 Managing resource types

The management of resource types can be enabled with AWS Config with the options of alerting, manually changing resource type and automatically changing resource type but when changing the type of the instance is stopped and if the students choose the wrong type frequently the service generates a noticeable cost. Making a single type of resource mandatory through policies eliminates instance down times and helps students be more mindful of what they use while creating instances. The appendix 2 refers to the EC2 service but resource types can be managed in the RDS service as well.

3.7.2 Managing instance's uptime

Instance uptime management is done through a Lambda function, where a Python script using the boto3 library for managing AWS services and resources collects the information of the running instances and calculates how long they have been running for, program code visible in Appendix 6 [21]. If the uptime of an instance exceeds a certain threshold, one hour, it is stopped. A CloudWatch rule with a cron job is created to execute the Lambda function every five minutes. The timeout period for the Lambda function is increased from 3 seconds to a minute to avoid failure with hundreds of instances.

3.8 Practical labs

To measure the knowledge acquired from the theoretical lessons and to consolidate the knowledge practical labs are created. Labs are designed from trivial to complex, latter ones using knowledge and components of previous assignments.

3.8.1 First exercise

The first lessons give the students an overview and history of the cloud technologies that are in use today hence the exercise consists of distributing the login data to the students, giving them a brief introduction of the AWS environment, how to find different services, showing what services are mainly going to be used and enabling multi factor authentication for added security.

There is no intention on checking whether or not the students enabled multi factor authentication, because the policies are set up in a way that makes it impossible to complete the fourth exercise without multi factor authentication.

3.8.2 Second exercise

The second lesson focuses on user, group, role, and policy management inside the AWS environment which is all done through the IAM service hence the second practical exercise consists of the students creating their own groups and assigning custom policies to said groups. The requirements for the policies are practical and come up in most firms at the start of implementing AWS into their systems and processes.

The students have to create a policy that allows the creation of EC2 instances and security groups for everyone but the users that this policy is applied to are only allowed to manage, start, stop and terminate, instances and security groups that have the tag with the key value pair of Environment and Test, the guide for the students is in Appendix 7.

3.8.3 Third exercise

With the third lessons and practical exercise the students start creating their own instances in AWS using the EC2 service, they also create key pairs, security groups and their own AMI image of the server for future practical exercises.

Since the process of creating an instance is incredibly simple the students also have to create their own security groups to manage inbound and outbound access, inbound allowing SSH, HTTP and ICMP, the latter two being for controlling the completion of the lab and outbound allowing everything. In addition, the students create key pairs to access the instances and attach a bootstrap script that installs a web server and create an index file to confirm the completion of the lab. Allowing inbound ICMP connections helps troubleshoot possible issues and to automate the grading of the students' work, the guide for the students can be found under Appendix 8.

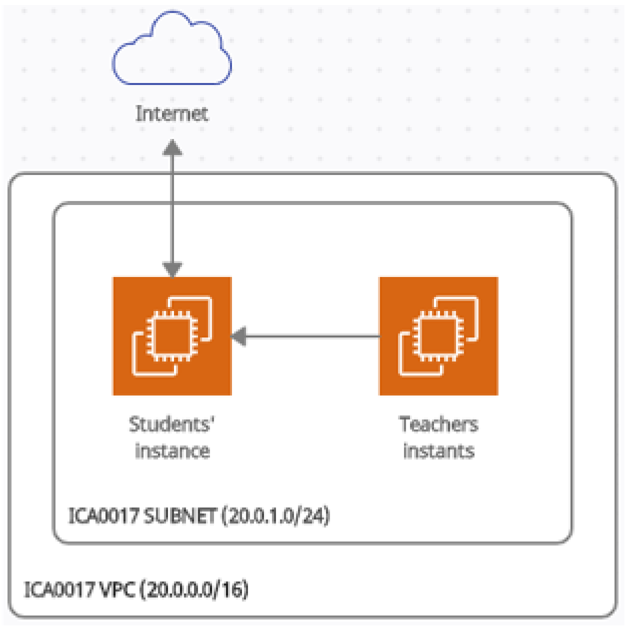

Figure 2. Visualisation of the third exercise.

3.8.4 Fourth exercise

In the fourth lesson the students are introduced to S3 which is as well their first serverless service. They are taught how AWS stores all the files uploaded in the buckets inside S3 and how different services can be used to communicate with S3 for example EC2.

In the exercise the students focus on learning the functionality of the S3 service with creating a custom role for their EC2 instances to be able to communicate with S3 to manage the buckets contents and create a static website. The uploading of the files is conducted through the EC2 instance and can be done using the GUI but to make the exercise more interesting the GUI is not used. The assignment for the students is visible under Appendix 9.

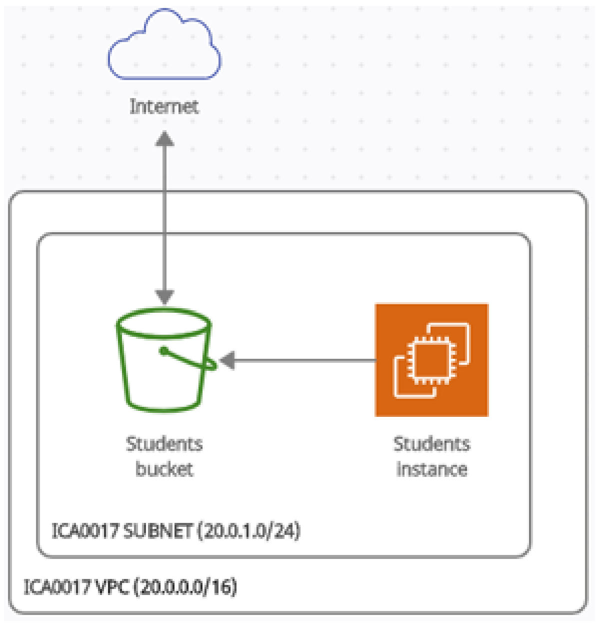

Figure 3. Visualisation of the fourth exercise.

3.8.5 Fifth exercise

The fifth exercise uses the networking and Content delivery service to teach the students how to set up and configure a secure network and launch instances in separate subnets and SSH accesses only permitted through a bastion host.

The students are not able to entirely create the environment illustrated in the appendix number 12 because the number of VPCs' is limited inside an account and that is why the internet gateway and the VPC with the VPC router has been created for the students as well as the Network ACL and Management subnet.

The students start by creating subnets, one public and one restricted, which they add to the corresponding routing tables, the term public is used when internet access is allowed. The following step is to create security groups that only allow SSH connections coming from the management security group because the management instance is in this context a bastion host. On the public security group HTTP and MySQL ports are allowed, with the former open to the internet and the latter only accepting communication from instances in the private security group. Final step of the lab is to confirm the correct configuration of the security groups with installing web hosting software on the public instance and showing the database version of the database installed in the database instance located in the private subnet using the php code provided by the teacher. The assignment for the students is visible under Appendix 10.

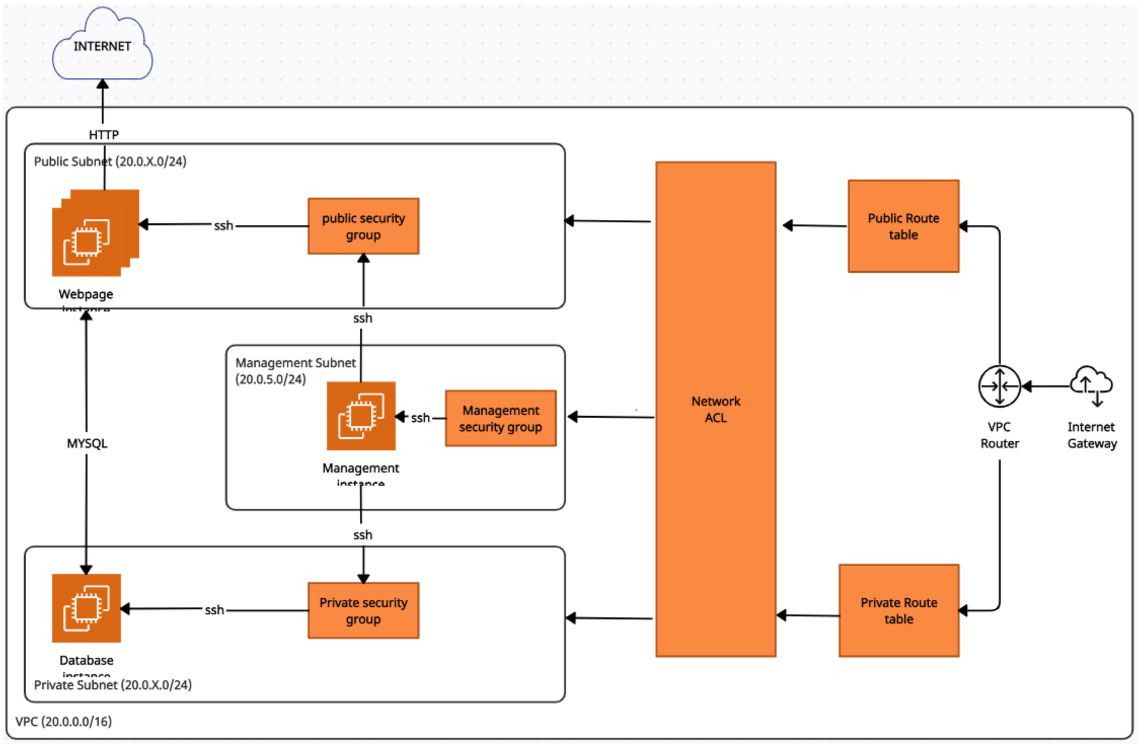

Figure 4. Visualisation of the fifth exercise.

3.8.6 Sixth exercise

The goal of the last exercise is to elevate the environment created in the previous exercise to be a highly available and fault tolerant environment. This is achieved by using Load balancers and Auto Scaling Groups in the AWS EC2 service.

The first step in creating a highly available environment is creating a target group for the load balancer, target groups register all the instances inside a load balancer. The next step is to create a load balancer. The instances are broadcasting a regular website to the internet and because of that an application load balancer is chosen which also enables routing traffic based on headers. The next step is to create another subnet because a Load balancer requires a minimum of 2 instances inside different availability zones. After the load balancer is created a launch configuration which gives the Auto Scaling Group directions on how to create an instance is next. The final step in setting up the highly available environment is creating the Auto Scaling Group. Students need to attach all the previously created resources: target group, load balancer and launch configuration, they also need to choose the VPC and subnets the Auto Scaling Group is permitted to create instances in. The requirements for the scaling policy are to launch a new instance when the CPU exceeds a load of 20%. The assignment for the students is visible under Appendix 11.

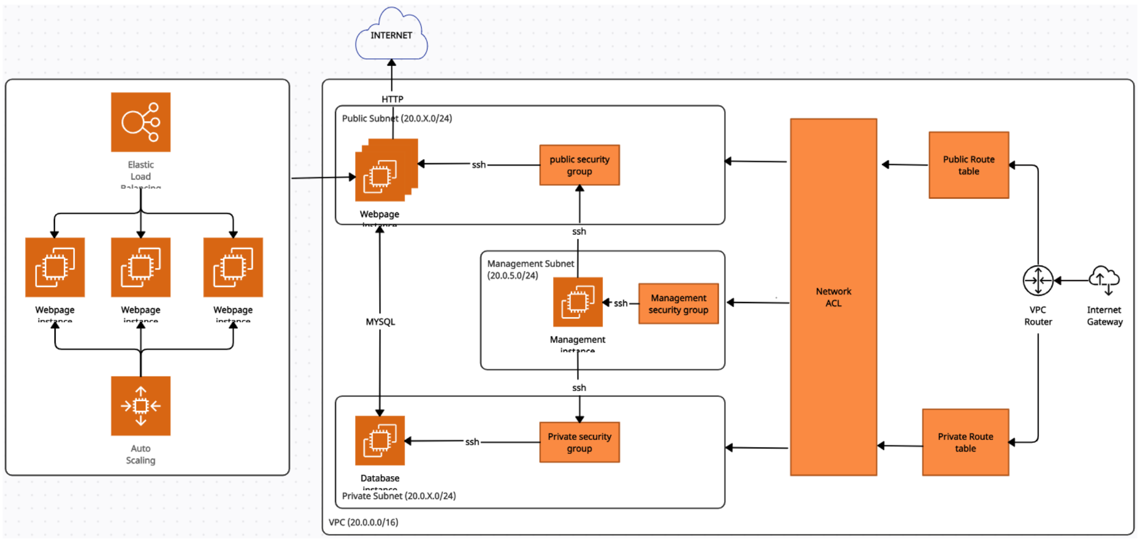

Figure 5. Visualisation of the sixth exercise.

3.9 Lessons learned.

During the creation of the lab environment the improvements are marked down that could be enabled in the future which are not implemented currently either due to the time constraint or the complicated communications with the schools IT help desk. AWS also sets restrictions which are found while creating the environment and the practical exercises.

3.9.1 Future improvements

To better the experience for the students AWS SSO should be connected with the schools domain controller in order for students to use their Uni-ID for logging into the AWS environment, due to poor communication with the TalTech IT helpdesk this solution did not get implemented. Another improvement is for each student to have their own account inside the landing zone, this approach is excluded because currently there is not a way to create the accounts automatically without having to go into an account to create a user which can then be accessed by the student, there may be a way to solve this issue with CloudFormation but due to the time constraint this solution did not get researched further. To implement the solution of each students having their own account more securely an Organizational Trail must be created, this is a CloudTrail created in the master account applied to the Students OU. Applying the trail to an OU will record all the actions of all the accounts in the OU, removing an account from the OU will not delete the logs [22].

The tagging policy created to only allow the users to tag EC2 service resources while creating them is too strict. The restriction is added to disallow users to add a tag to another's work when the original creator forgets to add a tag linking them to the created resource. This restriction results in users having to relaunch their resources because they forgot the tag and are unable to manage the created instance. The action of creating a tag should be allowed always, but to ensure the integrity of the resources the action for deleting tags should not be enabled.

For this semester, the grading is mostly manual or semi-automatic due to time constraints. The grading of the second exercise is not necessary because it can be combined with the third exercise, for example the students would have to complete the third exercise using their own security policy. The total automation of grading the practical exercises that host a webpage can be achieved through creating Lambda functions that analyse the webpages hosted on the EC2 instances or S3 buckets. For each practical exercise, the students are given a specific phrase that must be written in their index.html file which when found will create a file with the students' username in a specific bucket, one for each assignment. This reduces the need for the teacher to manually check IP addresses or export data to csv files and the students get feedback on the completion of the lab within minutes.

3.9.2 AWS restrictions

During the creation of the lab environment and of the practical exercises' restrictions are found which limit and change the solution. The first restriction appears while testing the landing zone with multiple lab environment accounts. The master account where the OU and the accounts are located is by default limited to five accounts. This limitation is lifted after communicating with the AWS helpdesk. Another restriction that appeared while creating the fifth lab is that the number of elastic or dedicated IP addresses, VPCs and internet gateways per region is limited to five of each. The quotas can be increased by communicating with the AWS helpdesk, but the restrictions are found too late and the problem is solved by changing the fifth task a little [22].

3.10 Cleaning the environment

After the semester has ended and all the students have had their work graded the lab environment needs all the used resources and users to be deleted and all the monitoring and security services need to be disabled. Deleting and disabling everything either through the GUI or the CLI takes a considerable amount of time because services often have a lot of dependencies which prompt errors when deleted in the wrong order. Often while launching resources the resources themselves create resources to be able to function for example when launching an EC2 instance and manually adding additional volumes to the instance then when deleting the EC2 instance only the Root volume is deleted, and the additional volume has to be deleted manually if not configured differently.

A more effective way of cleaning the lab environment is to delete the account itself. Deleting the old and creating a new account with enabling all the security and monitoring and implementing the IAM policies takes considerably less time than deleting resources manually. To delete an account first it has to leave the Organizational Unit and to do that payment information has to be entered into the account. After payment info is added and the Organizational Unit has been left a support plan must be chosen after closing the account [23].

4 Summary

The goal of the work is to create a scalable, secure, monitored, and easily manageable environment for the “Introduction to cloud technologies” course that minimizes manual work for the teacher and allows students to access different services in the AWS environment.

Scalability and ease of management of the environment is achieved in the thesis. Students are added into the system into the correct groups with the required permissions and login data automatically with the number of students taking the subject having no effect on the process. The instances provisioned by the students automatically shut down after running for an hour to save costs. All the costs generated by various services and resources are accumulated into the auditing account and can be visible in the lab account to get an overview of the costs and find possible misconfigurations.

Security of the environment is achieved by giving the users only the permissions necessary to complete the practical assignments. To ensure the safety of the users and the environment all the actions taken inside the environment are recorded by CloudTrail and sent to the security account.

In the thesis the Author provides information and data on how the environment is created and answers how the environment is scalable, secure, monitored and easily manageable.

References

[1] N. Davis, “AWS Certified Cloud Practitioner Exam Training 2020,” ACloudGuru, [Online]. Available: https://www.udemy.com/course/aws-certified-cloud-practitioner-training-course/. [Accessed 10 01 2021].

[2] M. R. Ryan Kroonenburg, “AWS Certified Solutions Architect Associate SAA-C02,” ACloudGuru, [Online]. Available: https://learn.acloud.guru/course/aws-certified-solutions-architect-associate/dashboard. [Accessed 13 01 2021].

[3] AWS, “AWS Educate,” AWS, [Online]. Available: https://aws.amazon.com/education/awseducate/. [Accessed 15 01 2021].

[4] cloud.vn, “AWS Landing Zone vs AWS Control Tower,” cloud.vn, [Online]. Available: https://cloud.vn/blog/aws-landing-zone-vs-control-tower/. [Accessed 15 01 2021].

[5] A. Larkin, “What You Need to Know About AWS’s Per-Second Billing,” [Online]. Available: https://cloudacademy.com/blog/what-you-need-to-know-about-aws-per-second-billing/. [Accessed 30 03 2021].

[6] AWS, “Amazon S3 pricing,” [Online]. Available: https://aws.amazon.com/s3/pricing/. [Accessed 10 05 2021].

[7] AWS, “AWS Budgets,” [Online]. Available: https://aws.amazon.com/aws-cost-management/aws-budgets/. [Accessed 15 01 2021].

[8] “AWS EC2 pricing,” [Online]. Available: https://www.awsprices.com/. [Accessed 30 03 2021].

[9] AWS, “AWS CloudTrail pricing,” [Online]. Available: https://aws.amazon.com/cloudtrail/pricing/. [Accessed 30 03 2021].

[10] AWS, “IAM JSON policy elements: Effect,” [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_effect.html. [Accessed 04 04 2021].

[11] AWS, “IAM JSON policy elements: Action,” [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_action.html. [Accessed 04 04 2021].

[12] AWS, “IAM JSON policy elements: Resource,” [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_resource.html. [Accessed 04 04 2021].

[13] AWS, “IAM JSON policy elements: Condition,” [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition.html. [Accessed 04 04 201].

[14] AWS, “AWS Security Hub,” [Online]. Available: https://aws.amazon.com/security-hub/. [Accessed 17 01 2021].

[15] AWS, “Amazon GuardDuty,” [Online]. Available: https://aws.amazon.com/guardduty/. [Accessed 17 01 2021].

[16] AWS, “Amazon CloudWatch,” [Online]. Available: https://aws.amazon.com/cloudwatch/. [Accessed 17 01 2021].

[17] AWS, “AWS Config,” [Online]. Available: https://aws.amazon.com/config/. [Accessed 17 01 2021].

[18] AWS, “Control access to EC2 resources using resource tags,” [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/control-access-with-tags.html. [Accessed 21 01 2021].

[19] L. Asi, “minedetector/CloudComputingSubject,” [Online]. Available: https://github.com/minedetector/CloudcomputingSubject. [Accessed 29 04 2021].

[20] D. H. a. J. Ruiz, “How to bulk import users and groups from CSV into AWS SSO,” [Online]. Available: https://aws.amazon.com/blogs/security/how-to-bulk-import-users-and-groups-from-csv-into-aws-sso/. [Accessed 23 01 2021].

[21] AWS, “EC2,” [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html#EC2.Client.describe_instances. [Accessed 12 04 2021].

[22] AWS, “Creating a trail for an organization,” [Online]. Available: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/creating-trail-organization.html. [Accessed 16 05 2021].

[23] AWS, “Amazon VPC quotas,” [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html. [Accessed 17 04 2021].

[24] E. Shanks, “Close an AWS Account Belonging to an Organization,” [Online]. Available: https://theithollow.com/2018/09/17/close-an-aws-account-belonging-to-an-organization/. [Accessed 17 04 2021].

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - IAM policy condition with service specific variable

```
"Condition": {
       "StringLikeIfExists": {
              "ec2:InstanceType": [
                     "t2.micro"
                     ]
              }
       }
```

Appendix 3 - IAM policy condition with AWS variable

```
"Condition": {
       "StringEquals": {
              "ec2:ResourceTag/User": "${aws:username}"
              }
        }
```

Appendix 4 – Automatically creating users in AWS

```
import csv
import xlsxwriter
import random

workbook = xlsxwriter.Workbook('send-info.xlsx')
worksheet = workbook.add_worksheet()
worksheet_row = 0
worksheet_column = 0

creation_file = open('user-creation-commands.txt', 'a')
deletion_file = open('user-deletion-commands.txt', 'a')
with open('tudengid.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        for i in row:
            user_row = i.split(';')
            if line_count == 0:
                line_count += 1
            else:
                username = user_row[1]
                email = user_row[4]
                pw_nr = random.randint(1,99999)
                password = username + str(pw_nr) + 'A'
                group = 'Tudengid'
                arn = 'arn:aws:iam::311962906195:policy/student-permissions'
                create_user = (f'aws iam create-user --user-name 
{username}\n')
                create_user_pw = (f"aws iam create-login-profile --user-name 
{username} --password '{password}' --password-reset-required\n")
                add_user_to_group = (f'aws iam add-user-to-group --user-name 
{username} --group-name {group}\n')
                creation_file.write(create_user)
                creation_file.write(create_user_pw)
                creation_file.write(add_user_to_group)
                remove_user_from_group = f'aws iam remove-user-from-group --
user-name {username} --group-name {group}\n'
                delete_user_login_profile = f'aws iam delete-login-profile --
user-name {username}\n'
                delete_user = f'aws iam delete-user --user-name {username}\n'

                deletion_file.write(remove_user_from_group)
                deletion_file.write(delete_user_login_profile)
                deletion_file.write(delete_user)
                worksheet.write(line_count, 0, email)
                worksheet.write(line_count, 1, username + '; ' + password)
                line_count += 1
    print(f'{line_count - 1} users created.')
creation_file.close()
deletion_file.close()
workbook.close()
```

Appendix 5 - Macro for sending emails

```
function sendEmails() {
  var sheet = SpreadsheetApp.getActiveSheet();
  var startRow = 1; // First row of data to process
  var numRows = 1; // Number of rows to process
  var dataRange = sheet.getRange(startRow, 1, numRows, 48);
  var data = dataRange.getValues();
  for (var i in data) {
    var row = data[i];
    var emailAddress = row[0]; // First column
    var message = row[1]; // Second column
    var subject = 'ICA0017 konto loomise andmed';
    MailApp.sendEmail(emailAddress, subject, message);
  }
```

Appendix 6 - Lambda function to stop running instances after an hour

```
import boto3
from datetime import datetime, timedelta

ec2 = boto3.client('ec2')
response = ec2.describe_instances(
    Filters=[
        {
            'Name': 'instance-state-code',
            'Values': ['16']
        }
    ]
)
for i in range(100):
    try:
        format = "%Y-%m-%d %H:%M:%S"
        instance_launch_time = 
response['Reservations'][i]['Instances'][0]['LaunchTime'].strftime(format)
        date_time_instance = datetime.strptime(instance_launch_time, '%Y-%m-
%d %H:%M:%S')
        now = datetime.now() - timedelta(hours=3)
        time1 = now.strftime(format)
        date_time_now = datetime.strptime(time1, '%Y-%m-%d %H:%M:%S')
        print(date_time_instance)
        print(date_time_now)
        uptime = (date_time_now - date_time_instance).total_seconds() / 60
        print(uptime)
        if uptime >= -120.0:
            instance_id = 
response['Reservations'][i]['Instances'][0]['InstanceId']
            print(instance_id)
            stop_response = ec2.stop_instances(
                InstanceIds=[
                    instance_id
                ],
                Force=True
            )
    except:
        continue
```

Appendix 7 - Second practical exercise

Create a security policy where only you can edit, create and delete security groups and EC2 instances. Everybody can view them. Policy name has to be the username - tag it with the key User and value <your-username>

To check that the policy works, create your own security group, the name has to start with your username, add the created policy to the group and then remove yourself from the Tudengid group.

If the policy works then write it after your username in the classes google sheet and we will check your policy and give feedback.

If you have gotten positive feedback remove your user from your created group and add yourself back to the Tudengid group.

Appendix 8 - Third practical exercise

1. Create an EC2 Security group that allows inbound connections from the internet to SSH, HTTP.
2. Create a Key pair, use your username and add a tag User: <username>
3. Create an EC2 instance using the security group from point 1.
4. Suggest using Amazon AMI.
    1. Instance type has to be t2.micro, other instance types are not allowed.
    2. Check that auto-assign public IP is Enabled and that you are using the ICA0017 VPC and the ICA0017 subnet
    3. Using a bootstrap script update the server and install httpd and create an index.html that displays your username in the web page.
    4. You have to add a Tag to the server, otherwise you will not be able to manage the server
        1. Key = User
        2. Value = <your username>
5. Using your instance, create yourself an AMI image.
    1. Name it for example username-ami
    2. Add a tag
6. Make sure that your website is visible, if it is then copy your instance ID in the classes google sheet under the third sheet
7. Stop the instance

TIPS:
When viewing if the website is up be sure to not use https
Instance connect has been disabled for safety concerns, please use SSH instead.
Add a tag with the key: Name and Value: <username> to find your resources more easily.

Appendix 9 - Fourth practical exercise

This lab will not work if you don't have MFA enabled from the first exercise.

1. Create a public S3 bucket, name has to start with your username and the rest can be whatever you want
2. Create an IAM Role for your EC2 instance to get access to S3 buckets. You can use pre-made roles by AWS
    1. The role name should be your username and be sure to add the tag
    2. Attach the role to the instance (if stuck, google, how to add role to an ec2 instance)
    3. inside the instance write the command
        1. aws s3 ls
    4. If you see the buckets that have been made in course AWS account S3 service, you can move on to the next step
3. Create a basic index.html and error.html file in the EC2 server and add them to your S3
    1. google the command: aws s3 sync
    2. If you get a permissions error then you have configured your role wrong
4. In the S3 settings allow static website hosting and test that the website is up
5. STOP the instance that you used

6. Copy the s3 url to the google sheet.

Appendix 10 - Fifth practical exercise

In this exercise you will have to host a webpage with a mysql database, both hosted in an air-gapped AWS environment, taking extra precautions to protect the systems containing user's data.
This approach is good to use when dealing with highly sensitive user information.

1. Create a Restricted subnet
2. Create a Public subnet
    a. After creation, change it's auto-assign IP settings and enable auto assigning of public IP
3. Create a public route table
    a. Name tag - leave this empty
    b. VPC - use ICA0017
    c. Add two tags: User: <username>, Name: <username>
    d. Add a route to the internet
        i. Destination should be 0.0.0.0/0 and target should be an internet gateway (the internet gateway has been created for you)
    e. Attach your public subnet to the public route table
4. Create a restricted route table
    a. Attach restricted subnet to it
5. Create a management security group that only allows SSH from the internet
6. Create a restricted security group that allows SSH only from the management security group, and MYSQL connection only from your public security group
7. Create an EC2 instance
    a. Use the public subnet so you can SSH into it
    b. Install MariaDB/MySQL on it
    c. configure user with password for the database
    d. Create an AMI image of the instance
        i. This will be the image you will use for your restricted instance
    e. Terminate the instance
8. Modify your existing public security group to only allow SSH connections from the management security group and make sure it allows only HTTP traffic from the internet
9. Create 3 EC2 instances (Do not create a spot instance)
    a. The first server must be in the Public subnet using the public security group created in lab 3, and have httpd & php (version 7+) installed
    b. The second server must be in the Restricted subnet using the restricted security group
        i. The instance must be launched using the AMI image created in the step 8
        ii. If you do not have a pem version of the key for the SSH connection available create a new key for the Restricted Instance.
    c. Third server is going to be the management server that you will use to access the public and restricted instances created before. It will be placed in the management subnet created for you.

    d. Open the .pem key in a text editor and create a file on your management instance for example privatekey.pem and paste the contents into it

10. SSH into your EC2 instance in the restricted subnet containing database from your management instance, code example below
    a. Ssh ec2-user@<non-public-IP> -i privatekey.pem
    b. Check if mysql is running
11. SSH into your public EC2 instance from your management instance
    a. Configure index.php to connect to the mysql database created in the private instance

12. When everything is finished copy the instance id-s, public instance IP address and a screenshot of the public server web page, that is showing the restricted server's database version, into the google sheet.

Appendix 11 - Sixth practical exercise

1. First create a target group for the load balancer
    1. Here will be all the instances that the load balancer chooses from
    2. The target type will be for instances
    3. The target group name has to begin with your username
    4. Protocol has to be 80
    5. VPC is ICA0017
    6. Use HTTP1 and give a tag at the end
    7. Do not register any targets at the moment
2. In the previous exercise you created a public subnet, make another one exactly like that but be sure that it is in a different availability zone
3. Next create an application load balancer
    1. Make it internet-facing and add a HTTP listener
    2. Select your 2 subnets
    3. Select the public security group created in the previous lab
    4. Select the target group created in the first step
    5. Create
4. If you have not done this create a image of your public instance
5. Create a launch configuration - this is what the auto Scaling will use to create new instances
    1. Name has to start with your username
    2. Use the public instance image or AMI
    3. Type has to be t2.micro
    4. Select your public security group
    5. Choose your key pair
6. Create a Auto Scaling group
    1. Name has to start with your username
    2. Select launch configuration and select the one you created in the previous step
    3. VPC is ICA0017 and subnets are your 2 public subnets
    4. Attach your load balancer
    5. Group size will be minimum 1 maximum 2
    6. Scaling policy will be target tracking
        1. Change name to start with your username
        2. CPU utilization
        3. Target value = 20
    7. Be sure to add a tag
7. Now if you look under instances you will find a instance with your name in the tag terminate it and go look into your auto scaling groups Activity to see if it has started to create a new instance
8. SSH into the instance and put it under a high load so that ASG will launch another instance.
    1. Instance load can be monitored when clicking on the instance and opening Monitoring

9. After the auto scaling group has created another instance then you can be sure that everything is working correctly on the Auto Scaling group side
10. Start your management and Restricted instances and SSH into one of your public instances and change the index.php file a little
11. Open the DNS name of your Load Balancer and hit refresh multiple times and see you it changes between the two instances.
12. Take screenshots with two different webpages under the same URL and paste them into the google sheet.
13. Delete your Auto Scaling Group do not delete anything else.