Project plan

A picture containing text

Description automatically generated

Table of Contents

[1. Project Definition: 2](#_Toc899678161)

[1.1 Project background: 2](#_Toc183316636)

[1.2 Problem definition: 2](#_Toc889327145)

[1.3 Project goal: 2](#_Toc427825138)

[1.4 Expected Results: 3](#_Toc1859815290)

[1.5 Way of Working: 3](#_Toc1377869301)

[1.6 Scope: 3](#_Toc611157818)

[Within the scope: 3](#_Toc984402566)

[Outside the scope: 3](#_Toc1557367440)

[2. Project Structure: 3](#_Toc525343290)

[2.1 Development Team: 4](#_Toc830081652)

[2.2 Tutor: 4](#_Toc880018532)

[3. Risk Assessment: 4](#_Toc583793693)

[4. Deliverables: 5](#_Toc207690793)

[Docker container for flask application and node exporter 6](#_Toc1588350905)

[Credentials database 6](#_Toc1485912178)

[Terraform, CDK, CloudFormation automation scripts 6](#_Toc1560683652)

[5. Planning: 6](#_Toc2000238945)

[6. MoSCoW table 8](#_Toc108266711)

[Version Control Table 9](#_Toc71776162)

# Project Definition:

## 1.1 Project background:

We are three Fontys ICT (Information and Communication Technologies) students tasked with automating the deployment of a cloud infrastructure. If we automate the creation of the infrastructure on AWS, we are free to select the type of service implemented.

## 1.2 Problem definition:

We are aiming to address the issue that professors lack a user-friendly environment in which to post recordings of demos and lectures. The lectures and demonstrations are currently submitted to teams, but the problem is that they are not structured in any way, so students must conduct extensive searches to locate the appropriate demo/lecture. The recordings on MS teams will be automatically erased after 120 days, meaning that if a student wishes to review a demo or lecture to prepare for the next semester, he or she would be unable to do so.

## 1.3 Project goal:

The objective is to create an autoscaled, multi-az, serverless infrastructure that can be moved between organizations and currently consists of a webserver that hosts a file converter and multi-cloud uploader.

Here, instructors can quickly submit recordings of their lectures and/or demonstrations. Users (students) can log in using their own credentials to search among the demos/lectures until they locate the one they want.

## 1.4 Expected Results:

Teachers will have the option of uploading their videos to several cloud providers and will be able to post their lessons and films in many sites using a video storage software that we plan to develop. Students will be able to access the videos through the web app by logging in with their own credentials; however, they will not be able to modify the submitted content. This application might be utilized in many schools and offices for comparable purposes.

## 1.5 Way of Working:

We will manage the project utilizing the Agile methodology, which will include weekly delivery of documentation and progress reports through weekly mentor meetings. The project will change over time based on requirements and lessons learnt, the work and study sprints are checked and guided by our mentor; Almost every workday, a Fontys conference room is selected for Case study and homework completion. The average daily duration is between two and three hours, but this can vary depending on the workload. Weekend online conferences are held if there is still work to be completed.

## 1.6 Scope:

### Within the scope:

This project will be completed by three Fontys ICT Eindhoven students. Before implementing everything, we will conduct research and testing utilizing an on-premises servers, AWS, and our laptops.

### Outside the scope:

We will not deliver any needed hardware for this project, nor the licences needed to complete this project.

# Project Structure:

This is a 16-week project with weekly meetings to ensure that we stay on track.

2.1 Development Team:

The developers of this project are:

Jorn Kosterman, [488058@student.fontys.nl](mailto:488058@student.fontys.nl)

Soufiane El-Atmani, [479242@student.fontys.nl](mailto:479242@student.fontys.nl)

Mihai Glodici, [485522@student.fontys.nl](mailto:485522@student.fontys.nl)

We are three students from Eindhoven's Fontys university. We all currently live in the Netherlands and attend the HBO ICT program.

Every week, we aim to ask questions to become more experienced and invested in the project. Multiple people take notes to ensure that nothing is forgotten from the meeting.

## 2.2 Tutor:

Our tutor is Andrius Kuprys

Email: [a.kuprys@fontys.nl](mailto:a.kuprys@fontys.nl)

# Risk Assessment:

These are some risks that can occur during the project:

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Risk | Probability | Countermeasure |
| 1 | As a result of someone shutting down the servers (because they are shared), we may lose a significant amount of progress and data. | Unlikely | Effective communication of who is doing what and when |
| 2 | A person can become ill (especially because of Covid-19). | Likely | We may utilize multiple platforms, such as Teams and Discord, to continue our project's development. |
| 3 | Someone unable to attend Fontys due to the public transportation strikes | Likely | We can use various platforms such as teams or discord to keep working on our project. |
| 4 | Price issues. Not checking how much we spend and end up with a considerable number of fines. | Likely | shutting-down our ECS instances at night. |
| 5 | Not being able to finish the project our because of unfamiliarity with the software. | Unlikely | Ask teachers for feedback and help |

# Deliverables:

***No hardware will be delivered by us\****

The deliverables are as follows:

* Ansible/Terraform file
* Source code files webapp
* User & Technical manual

Non-deliverables

* personal hardware used
* AWS cloud account

Infrastructure specifications:

### Docker container for flask application and node exporter

Flask application runs on a gunicorn webserver, which uses a nginx proxy for secure internet discoverability.

This flask application has a subprocess that operates the Prometheus node exporter. We utilized the Popen.subprocess command to run the bash script in deamon so that our website would continue to function.

### Credentials database

This is the database containing all the login information of users. It is getting input from the webapp. Planned to be in a separate closed off from the internet VPC.

Monitoring ECS instance

To collect logs, we will use Prometheus in conjunction with the Cloudwatch data source. In the same subnet, visualization is performed on a separate Grafana container. On the webapp Dockerfile, a node exporter is configured to share data with Prometheus. This guarantees that distinct auto-scaled Fargate instances are also monitored.

Ip range is restricted to 14x3 (triple multi-availability ECS VPC) hosts that are included to the Prometheus scrape file.

### Terraform, CDK, CloudFormation automation scripts

 Terraform, CloudFormation stacks, and the Cloud development kit are used to automate the creation of infrastructure. We will separate it into multiple independent scripts to generate various infrastructure components.

# Planning:

|  |  |
| --- | --- |
| Week | Activities |
| 1 | N/A |
| 2 | Brainstorm for ideas and discussing them with our tutor |
| 3 | Brainstorming about ideas and starting on Project Plan  Creating a GitLab environment  Uploading current documentation to GitLab |
| 4 | Finishing project plan and discussing it with our tutor  Starting on Design Document  Starting with first draft of flask web-app  -End of first sprint- |
| 5 | Finishing Design Document and discussing it with our tutor  Continuing with the development of our flask web-app |
| 6 | Implementing video converter in webapp  Late start on URS (Creating personas)  -End of second sprint- |
| 7 | Researching load balancing and autoscaling  Fixing sound video converter  Researching connection between infralab and AWS  Starting on implementation of this connection |
| 8 | Creation of first build URS  Implementing database in web-app  VPN connection to infralab from ec2 VPC  Continuing URS (Product vision board and roadmap) |
| 9 | Implementing load balancing  Hosting webapp in docker on ECS  Creating AWS network  Preparing for presentation |
| 10 | Presentation MVP2  Hosting webapp on ECS Fargate manually  MultiCloud Azure Blob storage research |
| 11 | Webapp added to Elastic Container repository  Amazon Machine image for ec2 instances  Multi Availability zone research for VPC and LoadBalancer |
| 12 | -End of third sprint- |
| 13 | Limited Cloudformation stack hosting webapp on Fargate  VPN tunnel addition for Fontys on-premise storage |
| 14 | Implemented auto-scaling, ecs, elb, iam, route53, HTTPS automation scripts  Added database, PXC, VPN, backend VPC, S3 to Terraform script |
| 15 | Prometheus + Cloudwatch monitoring  Grafana Cloudformation stack and Dashboard configuration  Locust CDK deployment and load testing |
| 16 | -End of fourth sprint- |
| 17 | Finishing our project and documentation  Giving a demo |

***NOTE: There might be some changes to our planning during our project!***

# MoSCoW table

The table below will give an overview of the importance/priority for the requirements of this project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Proficient** | **Advanced** | |
|  | **M**ust | **S**hould | **C**ould |
| App (automated script) with a GUI | v |  |  |
| App creates an infrastructure with 1 DB and 1 WEB server | v |  |  |
| App creates an infrastructure with a choice from 2 DBs and 2 WEB servers |  | v |  |
| Create a VPN connection between a Cloud environment and on-premises server | v |  |  |
| App generates user credentials to connect to DB and/or WEB server | v |  |  |
| Generated user credentials are sent to a user automatically |  | v |  |
| Create a Network diagram | v |  |  |
| 2 Clouds are used |  |  | v |
|  |  |  |  |
| Automation |  |  |  |
| App uses Terraform to create an infrastructure | v |  |  |
| Create a Docker container | v |  |  |
| Use of AWS Lambda | v |  |  |
| Use of AWS Gateway |  | v |  |
| Orchestration |  |  |  |
| Use of Ansible and Terraform to orchestrate a complex task |  |  | v |
| Create a Process diagram (automation steps in the App) | v |  |  |
| Use of AWS Step functions | v |  |  |
| Create multiple Docker containers, that communicate with each other |  | v |  |
| Use of Kubernetes environment |  |  | v |
| Security |  |  |  |
| Create IAM users / group policy / key-pair authentication. | v |  |  |
| Server-side or customer provided key encryption for S3 bucket files. | v |  |  |
| Use of secrets management for database instances. | v |  |  |
| Public front-end and private back-end instances in separate subnets. | v |  |  |
| Public front-end and private back-end instances in separate VPCs. |  | v |  |
| Use of NAT instance design for private back-end instance(s). |  | v |  |
| Use of Transit gateway with BGP routing protocol to access several back-end instances in different subnets (different projects) |  |  | v |
| Demonstrate the analysis of VPC flow log files with native CloudWatch or Athena tools, generate notifications and alerts for service management ( e.g. ITIL: Incident process ) | v |  |  |
| Demonstrate the analysis of CloudTrail security events with native CloudWatch or Athena tools, generate notifications and alerts for service management (e.g. ITIL: Incident process ) | v |  |  |
| Demonstrate a balanced and failsafe design between several public front-end instances (use of Route53, application loadbalancer or auto-scaling group) | v |  |  |
| Demonstrate a private Route53 host zone design use by public instance(s) |  | v |  |
| Demonstrate the upgrade of HTTP to HTTS connection without any changes to EC2 instances (use of Certification Manager and load balancer) | v |  |  |
| Network Orchestration |  |  |  |
| Automate public and private instances in separate subnets design with automated scripts (Powershell, Terraform) or CloudFormation. | v |  |  |
| Automate public and private instances design in separate VPCs design (including NAT instance) with automated scripts (Powershell, Terraform) or CloudFormation. |  | v |  |
| Automate public and private instances in separate VPCs design (including NAT instance) and in multiple regions with automated scripts (Powershell, Terraform) or CloudFormation. |  |  | v |
| Monitoring |  |  |  |
| The product should generate logs, which should be regularly shipped to some monitoring service, so that the contents of these logs could be observed/analyzed |  |  | v |
| Load testing performed, monitored and recommendations provided as a result |  | v |  |
| The provisioned components are automatically registered in the monitoring service/tool | v |  |  |
| Supporting Services |  |  |  |
| Multi-AZ deployment of components | v |  |  |
| Files versioning turned-on (if you’re using S3) | v |  |  |
| Timely backups configured | v |  |  |
| Failover arranged |  | v |  |
| The costs should be thoroughly analyzed and there should be justification given for any component/service selected. | v |  |  |

# Version Control Table

|  |  |  |
| --- | --- | --- |
| *Version* | *Date* | *Changes* |
| *0.1* | *10/09/2022* | *First draft* |
| *0.2* | *15/09/2022* | *Rephrased text formally* |
|  |  |  |
| *0.3* | *13/10/2022* | *rephrased “mainly” to “unlikely"* |
|  |  | *Added “non deliverables”* |
| *0.4* | *14/10/2022* | *Added Version Control Table* |
|  |  | *Detailed Way of working* |
|  |  | *Detailed Project Goal* |
| *0.5* | *21-10-2022* | *Edited sprint/planning goals* |
| *0.6* | *08-01-2023* | *Changed description of “Monitoring”* |
| *1.0 release* | *12/01/2023* | *Rephrasing/rewriting most text*  *Updating technical explanations and planning; entire rework of document* |