A logo of a cheese head

Description automatically generated

|  |
| --- |
|  |
| Cheesehead hosting  Design document |
| |  |  |  | | --- | --- | --- | | Author: Group 1 | Date: 21/9/2023 | Version: 1.0 | |

Table of Contents

[1. Introduction 3](#_Toc156292613)

[1.1 Project background information 3](#_Toc156292614)

[1.2 Current situation 3](#_Toc156292615)

[2. Requirements 4](#_Toc156292616)

[3. Architecture of the network 7](#_Toc156292617)

[3.1 Diagrams 7](#_Toc156292618)

[3.1.1 Network diagram 7](#_Toc156292619)

[3.1.2 Process orchestration diagram 8](#_Toc156292620)

[3.2 Infrastructure solutions 10](#_Toc156292621)

[3.2.1 IP plan 10](#_Toc156292622)

[3.2.2 EC2 instances 11](#_Toc156292623)

[3.2.3 Lambda functions 11](#_Toc156292624)

[3.2.4 Step functions 14](#_Toc156292625)

[3.2.5 Elastic auto scaling 15](#_Toc156292626)

[3.2.6 Routing and internet gateways 16](#_Toc156292627)

[Internet gateways 16](#_Toc156292628)

[Routing tables 16](#_Toc156292629)

[Peering connections 17](#_Toc156292630)

[3.2.7 Load balancer & Target group 18](#_Toc156292631)

[3.2.7.1 Target group 18](#_Toc156292632)

[3.2.7.2 Load balancer 19](#_Toc156292633)

[3.2.8 Route53 21](#_Toc156292634)

[3.2.9 Elastic Container service 22](#_Toc156292635)

[3.2.9.1 Task definitions 22](#_Toc156292636)

[3.2.9.2 Services 26](#_Toc156292637)

[3.2.10 API gateway 28](#_Toc156292638)

[3.2.11 Other peripherals 29](#_Toc156292639)

[3.2.11.1 EFS 29](#_Toc156292640)

[4. Naming connection 30](#_Toc156292641)

[4.1.1 Username convention 30](#_Toc156292642)

[4.1.2 System naming convention 30](#_Toc156292643)

[5. Authorization plan 31](#_Toc156292644)

[5.1 IAM policies 31](#_Toc156292645)

[5.1.1 ECStoEFS 31](#_Toc156292646)

[5.1.2 ECStoEFSgeneral 31](#_Toc156292647)

[5.1.3 AllowSSMcmd 31](#_Toc156292648)

[5.2 IAM roles 32](#_Toc156292649)

[5.2.1 ecsTaskECStoEFSgeneral 32](#_Toc156292650)

[5.2.2 ecsTaskECStoEFS 32](#_Toc156292651)

[5.2.3 LambdaSSMrole 32](#_Toc156292652)

[6. Security 33](#_Toc156292653)

[6.1 Security groups 33](#_Toc156292654)

[6.2 CloudWatch 35](#_Toc156292655)

[6.3 CloudTrail 36](#_Toc156292656)

[7. Website 39](#_Toc156292657)

[7.1 Website wireframe 39](#_Toc156292658)

[7.2 Database design 40](#_Toc156292659)

[9. References 41](#_Toc156292660)

# 1. Introduction

## 1.1 Project background information

Cheesehead Hosting has established itself as trustworthy hosting company over the years. We take pride in providing our clients with a high amount of reliability and great service. We are a company that stands with our clients, so we aim to save our clients as much money as possible to establish trust between the company and the clients. We know that having a strong online presence is super important, whether you're a small business, a new startup, or a big company. So, we're here to give you top-notch hosting that's all about quality. We offer a variety of hosting options that can be customized to fit your needs, making sure your website or app runs smoothly and efficiently.

## 1.2 Current situation

Today, the infrastructure finds itself at a critical juncture as it recognizes the need to modernize its network infrastructure. Over the years the company has earned a good reputation for hosting website and for cloud storage. However, the existing network infrastructure has become dated and requires big upgrades to meet the evolving demand and expectations from their customers.

# 2. Requirements

The MoSCoW table below offers a clear overview of the network functionalities and capabilities.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |  |
| Network diagram | v |  |  |
| 2 Clouds are used |  |  | v |
| Automation |  |  |  |
| App uses Ansible 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 |  |  |  |
| Created 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 |
| 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 |  |  |
| Analysis of CloudTrail security events with native CloudWatch or Athena tools, generate notifications and alerts for service management (e.g. ITIL: Incident process) | v |  |  |
| balanced and failsafe design between several public front-end instances (use of Route53, application/network loadbalancer or/and auto-scaling group) | v |  |  |
| private Route53 host zone design used 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 |  |  |
| Security management analysis (Threat, Risk Analysis and mitigations) | 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 logs should be regularly shipped to some central storage/monitoring service for maintaining visibility into the health and performance of your infrastructure and applications |  | v |  |
| Logs contents are parsed (if needed) and analyzed to, for instance, enabling triggering alerts, or for better observability (a dashboard implemented) |  |  | v |
| Load testing performed, monitored and recommendations provided as a result |  | v |  |
| The provisioned components are automatically registered in the monitoring service/tool | v |  |  |
| Cost/Usage alerts are implemented | v |  |  |
| Some resource health alerts are implemented | v |  |  |
| The provisioned components are automatically tagged, so that it’s possible to determine which case/project/team/customer they belong to. This information should be exposed to the monitoring system |  | v |  |
| Supporting Services |  |  |  |
| Multi-AZ deployment of components | v |  |  |
| Timely backups configured | v |  |  |
| If an application is re-deployed, it should restore the relevant data from the backup | v |  |  |
| Monitoring is embedded into the portal, so that users can observe the status of their applications from one central point of access. |  | v |  |
| The costs should be thoroughly analyzed and there should be justification given for any component/service selected. | v |  |  |
| A simple service desk is implemented (the minimum of 4 ITIL processes are implemented) \* | v |  |  |
| An advanced service desk is implemented (all six ITIL processes are implemented) \* |  | v |  |
| Main portal has embedded service desk functionality |  | v |  |
|  |  |  |  |
| Problems (critical alerts) are automatically registered in the service desk |  |  | v |
| A created and executed test plan for a given infrastructure based on Security Management analysis. | v |  |  |

# 3. Architecture of the network

## 3.1 Diagrams

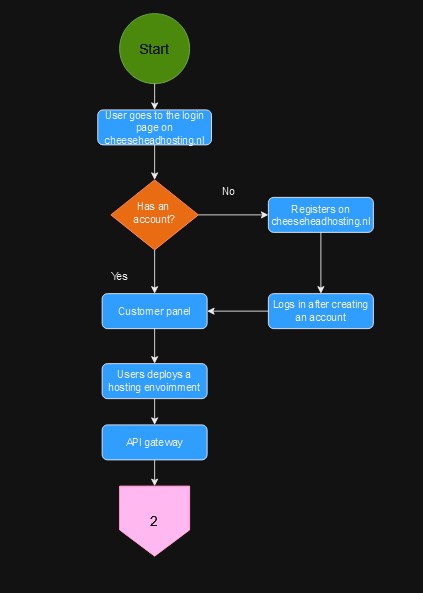
### 3.1.1 Network diagram

In the figure below you can see the overall network diagram. It shows the infrastructure we have created to host our own website and show the way how the customers enviornment are getting deployed.

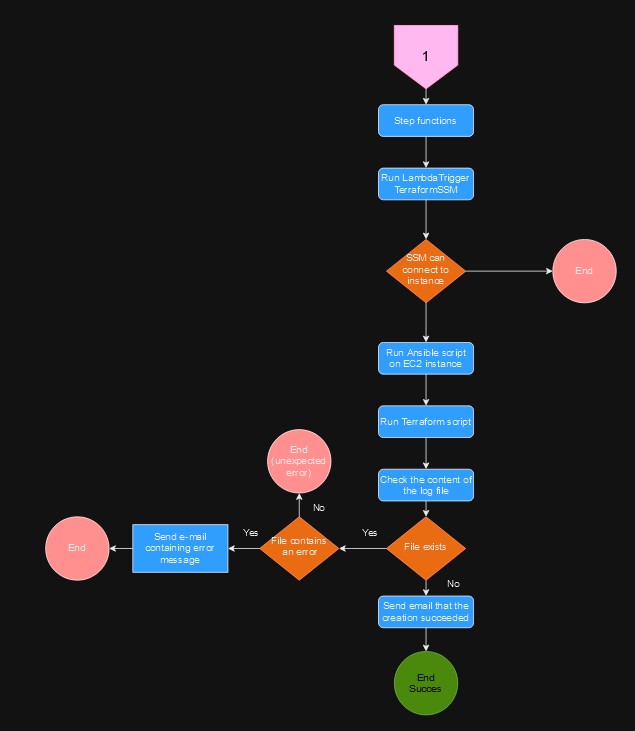
A screenshot of a computer

Description automatically generated

### 3.1.2 Process orchestration diagram

In the figure below you can see the start of the process orchestration diagram. The diagram goes through all the steps from the user initiating a deployment to the result. The figure continues the next page.

In the image below you can find the second part of the process orchestration diagram. This continues from the API gateway.



## 3.2 Infrastructure solutions

## 3.2.1 IP plan

#### 3.2.1.1 Operation IP-plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operational IP-plan | | | | |
| Region | **VPC** | **Subnet** | | **Node IP** |
| Ireland | Public VPC 172.16.0.0/16 | Web\_sub | 172.16.1.0/28 | 192.16.1.4 |
| Ireland | Private VPC 172.17.0.0/16 | DB\_sub | 172.17.1.0/28 | 172.17.1.4 |
|  |  | Prod\_sub | 172.17.2.0/28 | 172.17.2.5 |
|  |  | NAT\_sub | 172.17.254.0/28 | 172.17.254.4 |
| Frankfurt | Public VPC 172.18.0.0/16 | Web\_sub | 172.18.1.0/28 | 182.18.1.4 |
| Frankfurt | Private VPC 172.19.0.0/16 | DB\_sub | 172.19.1.0/28 | 172.19.1.4 |
|  |  | Prod\_sub | 172.19.2.0/28 | 172.19.2.5 |
|  |  | NAT\_sub | 172.19.254.0/28 | 172.254.4 |

#### 3.2.1.2 Customer IP-plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Client IP-plan | | | | | |
| Region | **AZ** | **VPC** | **Subnet** | | **Node IP** |
| Ireland | AZ 1 | Public VPC 10.0.0.0/16 | Web\_sub | 10.0.1.0/28 | 10.0.1.4 |
| Ireland | AZ 1 | Private VPC 10.1.0.0/16 | DB\_sub | 10.1.1.0/28 | 10.1.1.4 |
|  |  |  | NAT\_sub | 10.1.254.0/28 | 10.1.254.4 |
| Ireland | AZ 2 | Public VPC 10.0.2.0/16 | Web\_sub | 10.0.2.0/28 | 10.0.2.4 |
| Ireland | AZ 2 | Private VPC 10.1.3.0/16 | DB\_sub | 10.1.3.0/28 | 10.1.3.5 |
|  |  |  | NAT\_sub | 10.1.253.0/24 | 10.1.253.4 |
| Frankfurt | AZ 1 | Public VPC 10.2.0.0/16 | Web\_sub | 10.2.1.0/28 | 10.2.1.4 |
| Frankfurt | AZ 1 | Private VPC 10.3.0.0/16 | DB\_sub | 10.3.1.0/28 | 10.3.1.4 |
|  |  |  | NAT\_sub | 10.3.254.0/28 | 10.3.254.4 |
| Frankfurt | AZ 2 |  | Web\_sub | 10.2.2.0/28 | 10.2.2.4 |
| Frankfurt | AZ 2 |  | DB\_sub | 10.3.2.0/28 | 10.3.2.5 |
|  |  |  | NAT\_sub | 10.3.254.0/28 | 10.3.253.3 |

## 3.2.2 EC2 instances

|  |  |  |  |
| --- | --- | --- | --- |
| Operational IP-plan | | | |
| Region | **Instances** | **Public IP** | **Private IP** |
| Frankfurt | CH-NAT01 | 54.155.99.119 | 172.17.254.3 |
| Frankfurt | CH-OP01 | None | 172.17.2.4 |
|  |  |  |  |
| Ireland | CH-NAT02 | 63.34.50.30 | 172.17.254.4 |
| Ireland | CH-OP02 | None | 172.17.2.5 |

## 3.2.3 Lambda functions

Purposes of our Lambda function and what they will do:

**LamdaTriggerTerraformSSM**

* Reads JSON requests for instance creations on the website which are listed below:
  + Region
  + Redundancy
  + projectName
  + programmingLanaguage
  + dbType
  + clientEmail
* Ansible
  + SSM is used to call the ansible playbook. The ansible playbook is then run using the user inputted variables. Ansible will communicate with the EC2 instance.
* Loop
  + The lambda function enters a loop constantly pulling information from the SSM service to get the status of the command execution. In the script, this is used by the “get\_command\_invocation” method and it keeps going until it gives a status of “Success” or “Failed”.

**Project-CheckFile-EC2**

* Triggers SSM
  + SSM is used to investigate the instance that runs the Terraform script if it has a file named randomnaam.tf.
* Check content of file
  + After retrieving the files content, it checks the content of the file. The file contains the status of what happened to the terraform script the file can have the following statuses: Succes, Failed, Cancelled.
* Output
  + If the output is success, it gives the output ‘Succes’ if the status is failed or cancelled it returns the status failed.

**Project-Send-Email**

* Input parameters
  + Event contains information about the mail to be set such as the subject, body, and recipients.
* Extracting email information
  + Extracts the subject body and recipients from the event parameters.
* Fetching information
  + Retrieves the sender’s email address and password from environment variables.
* Sending email
  + Calls the send email function with the correct information.
  + Uses the SMTP\_SSL class from the SMTP library to establish a connection to the SMTP server from Gmail.
* Return response.
  + If the email is sent successfully the function returns a success message.

**SNOweek16**

* Initialization:
  + Sets up an S3 client and configures a logger for information logging.
* Source and Destination Buckets:
  + Defines source (snoweek8) and destination (snoweek16cloudtrailbackup) S3 buckets.
* Backup Loop:
  + Initiates a loop for listing and copying objects from the source to the destination bucket.
* List Objects:
  + Uses list\_objects\_v2 to retrieve a list of objects from the source bucket.
* Object Copy:
  + If there are objects in the response, iterates over each object.
  + Creates a copy source and copies the object to the destination bucket.
  + Logs the success of each backup.
* Continuation Token:
  + Checks if there are more objects to list.
  + If true, updates the continuation token to proceed with the other object(s).
* Loop Termination:
  + Breaks out of the loop when it is finished processing all objects.
* Completion Message:
  + Returns a message saying the backup is completed

## 3.2.4 Step functions

We are making use of one general state machine that is executing various steps. The purpose of this step functions workflow is to automate the process of creating the infrastructure and sending a failure or success mail containing the user credentials to make use of the infrastructure.

Below you can find a more detailed breakdown of what all the steps in the state machine do.

1. Run Terraform
   * Triggers the workflow by invoking a Lambda function to trigger Ansible & Terraform execution.
   * Handles retries in case of Lambda service exceptions or there are too many requests.
   * Stores the result of the Terraform execution in the path $.Terraform.
2. CheckFileExists
   * Invokes a Lambda function to check the existence of a file on the EC2 instance.
   * Stores the lambda function’s result in $.lambdaResult.
3. CheckFileResult
   * Peforms a conditional check on the status returned by the file-check lambda function.
   * If the status is success it goes to FileExists if failed it goes to FileDoesNotExist otherwise it goes to FileCheckError.
4. FileExists
   * If the file exists on the EC2 instance, passes a message to SendEmail-Fail.
   * Stores the success message in the path $.outputFile.
5. SendEmail-Fail
   * Invokes a Lambda function to send an email mentioning the failure of environment creation.
   * Handles retries for Lambda service exceptions or to many requests at the same time.
   * Transitions to "File Fail" after email notification.
6. File Fail
   * Represents a failure state with the error TerraformFileFound
7. FileDoes Not Exist
   * If the file does not exist on the EC2 instance, it passes a message to SendEmail-Success.
   * Stores the message in the path $.outputFile
8. SendEmail-Succes
   * Invokes a Lambda function to send an email indicating the successful deployment of the environment.
   * Handles retries for lambda exceptions or too many requests.
   * Transitions to success after email notification.
9. Succes
   * Represents a successful completion state.
10. FileCheckError
    * Represents a failure state with the error Failed to use SSM in case of an unknown error during file existence check.

## 3.2.5 Elastic auto scaling

**CH-Frontend-AutoScaling (ECS Fargate)**

* + Policy name: CH-Frontend-AutoScaling
  + Minimum number of tasks: 1
  + Maximum number of tasks: 3
  + Scaling policy type: Target Tracking
  + ECS service metrics: ECSServiceAverageCPUUtilization
  + Target value: 70%
  + Scale-out cooldown period: 300
  + Scale-in cooldown period: 300

**CR-(Project name)-Autoscaling (ECS AGS)**

* VPC zone identifier:
  + CR-(Project name) public subnet AZ-1-2-1 (eu-west-1a)
  + CR-(Project name) public subnet AZ-1-2-2 (eu-west-1b)
* Desired capacity: 1
* Max size: 3
* Min size: 1
* Launch template.
  + IDL: ECS\_LT.ID (it changes each time the terraform script ran).
  + Version: Latest
  + Tags:
    - Key: AmazonECSManaged
    - Value: True
    - Propagate at launch: True

**CR-(Project name)-ECS-capacity-provider**

* Auto scaling provider group
  + AWS\_autoscaling\_group.ecs\_asg.arn (it changes each time the terraform script ran).
  + Managed scaling
    - Maximum scaling step size: 1000
    - Minimum scaling step size: 1
    - Status: Enabled
    - Target capacity: 1

CR-(Project name)-ECS-cluster-capacity-provider

* Capacity providers: aws\_ecs\_cluster.clinet\_cluster.name (it changes each time the terraform script ran).
* Default capacity provider strategy
  + Base: 1
  + Weight 100
  + Capacity provider: aws\_ecs\_capacity\_provider.ecs\_capacity\_provider.name (it changes each time the terraform script ran).

## 3.2.6 Routing and internet gateways

### Internet gateways

|  |  |  |
| --- | --- | --- |
| Internet gateways | | |
| Name | **State** | **VPC ID** |
| Customer\_IGW\_public | Attached | Customer\_VPC\_Public |
| Cheesehead VPC public igw | Attached | Cheesehead hosting VPC public |
| Cheesehead VPC public NAT igw | Attached | Cheesehead hosting VPC private |
| Customer\_IGW\_Private | Attached | Customer\_VPC\_Private |

### Routing tables

|  |  |  |
| --- | --- | --- |
| NAT routing | | |
| Destination | **Target subnet** | **Target name** |
| 0.0.0.0/0 | 0.0.0.0/0 | Internet gateway, cheesehead VPC Private NAT igw |
| 172.16.0.0/16 | 172.17.0.0 | VPC peering connection, cheesehead public to private |
| 172.17.0.0/16 | Local | - |

|  |  |  |
| --- | --- | --- |
| Web routing | | |
| Destination | **Target subnet** | **Target name** |
| 0.0.0.0/0 | 0.0.0.0/0 | Internet gateway, cheesehead VPC Private NAT igw |
| 172.16.0.0/16 | Local | - |
| 172.17.0.0/16 | 172.17.0.0 | VPC peering connection, cheesehead public to private |

|  |  |  |
| --- | --- | --- |
| Database routing | | |
| Destination | **Target subnet** | **Target name** |
| 172.16.0.0/16 | 172.17.0.0 | VPC peering connection, cheesehead public to private |
| 172.17.0.0/16 | Local | - |

|  |  |  |
| --- | --- | --- |
| CR – (project name) NAT routing | | |
| Destination | **Target subnet** | **Target name** |
| 0.0.0.0/0 | 0.0.0.0/0 | Internet gateway, Customer\_IGW\_private |
| 10.1.0.0./16 | 10.1.0.0 | VPC peering connection, client-public-to-private |
| 10.0.0.0./16 | Local | - |

|  |  |  |
| --- | --- | --- |
| CR – (project name) Web routing | | |
| Destination | **Target subnet** | **Target name** |
| 0.0.0.0/0 | 0.0.0.0/0 | Internet gateway, Customer\_IGW\_private |
| 10.1.0.0./16 | Local | - |
| 10.0.0.0./16 | 10.0.0.0 | VPC peering connection, client-public-to-private |

|  |  |  |
| --- | --- | --- |
| CR – (project name) Database routing | | |
| Destination | **Target subnet** | **Target name** |
| 10.1.0.0./16 | Local | - |
| 10.0.0.0./16 | 10.0.0.0 | VPC peering connection, client-public-to-private |

### Peering connections

|  |  |  |  |
| --- | --- | --- | --- |
| Cheesehead public to private | | | |
| Requester VPC | **Request CIDRs** | **Accepter VPC** | **Accepter CIDRs** |
| Cheesehead hosting VPC public | 172.16.0.0/16 | Cheesehead Hosting VPC private | 172.17.0.0/16 |

|  |  |  |  |
| --- | --- | --- | --- |
| Customer public to private | | | |
| Requester VPC | **Request CIDRs** | **Accepter VPC** | **Accepter CIDRs** |
| Customer\_VPC\_Public | 10.1.0.0/16 | Customer\_VPC\_Private | 10.0.0.0/16 |

## 3.2.7 Load balancer & Target group

### 3.2.7.1 Target group

#### Target Group Configuration Cheesehead website:

##### Target Type:

* **Target Type:** Instances
* **Protocol Port:** 80
* **VPC:** Cheesehead Hosting public

##### Health Checks:

* **Health Check Protocol:** HTTP
* **Health Check Path:** /

##### Targets:

* ECS 172.16.3.14 (eu-west-1c)

#### Target Group Configuration Customers:

##### Target Type:

* **Target Type:** Instances
* **Protocol Port:** 80
* **VPC:** Customer VPC public

##### Health Checks:

* **Health Check Protocol:** HTTP
* **Health Check Path:** /

##### Targets:

* Adds the instances that are created IP varies.

### 3.2.7.2 Load balancer

#### Load balancer configuration CH-LoadBalancer

##### Basic Configuration:

* **Scheme:** Internet facing
* **IP Address Type:** IPv4

##### Network Mapping:

* **VPC:** Cheesehead Hosting VPC public.
* **Mappings:**
  + cheesehead-website-subnet-1b (eu-west-1b)
  + cheesehead-website-subnet-1c (eu-west-1c)

##### Security Group:

* **Security Group Name:** CheeseHeadweb-SG

##### Listeners and Routing:

* **Protocol:** HTTPS
* **Port:** 443
* **Target Group:** CH-Target group

##### Secure Listener Settings:

* **Policy Name:** ELBSecurityPolicy-TLS13-1-2-2021-06
* **Default SSL/TLS Server Certificate:** From ACM
* **Certificate:** [www.cheeseheadhosting.nl](http://www.cheeseheadhosting.nl)

#### Load balancer configuration CR-(project name)-Loadbalancer

##### Basic Configuration:

* **Scheme:** Internet facing
* **IP Address Type:** IPv4

##### Network Mapping:

* **VPC:** Cheesehead Hosting VPC public.
* **Mappings:**
  + CR-(Project name) public subnet AZ-1-2-1 (eu-west-1a)
  + CR-(Project name) public subnet AZ-1-2-2 (eu-west-1b)

##### Security Group:

* **Security Group Name:** CR-(project name)-web-SG

##### Listeners and Routing:

* **Protocol:** HTTPS
* **Port:** 443
* **Target Group:** CH-Target group

##### Secure Listener Settings:

* **Policy Name:** ELBSecurityPolicy-TLS13-1-2-2021-06
* **Default SSL/TLS Server Certificate:** From ACM
* **Certificate:** [\*.cheeseheadhosting.nl](http://www.cheeseheadhosting.nl)

## 3.2.8 Route53

#### 3.2.8.1 Public Hosted zones

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Records in cheeseheadhosting.nl | | | | | |
| Record name | Type record | Differentiator | Value | Alias | Routing Policy |
| cheeseheadhosting.nl | A | - | www.cheeseheadhosting.nl | Yes | Simple |
| cheeseheadhosting.nl | NS |  | ns-1881.awsdns-43.co.uk  ns-1438.awsdns-51.org  ns-962.awsdns-56.net  ns-34.awsdns-04.com | No | Simple |
| cheeseheadhosting.nl | SOA |  | ns-1881.awsdns-43.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400 | No | Simple |
| \*.cheeseheadhosting.nl | A | - | www.cheeseheadhosting.nl. | Yes | Simple |
| web.cheeseheadhosting.nl | A | - | s3-website.eu-central-1.amazonaws.com. | Yes | Simple |
| www.cheeseheadhosting.nl | A | Primary | dualstack.ch-loadbalancer-514737101.eu-west-1.elb.amazonaws.com. | Yes | Failover |
| www.cheeseheadhosting.nl | A | Secondary | web.cheeseheadhosting.nl | Yes | Failover |

#### 3.2.8.1 Private Hosted zones

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Records in project-database | | | | | |
| Record name | Type record | Differentiator | Value | Alias | Routing Policy |
| project-database | NS | - | ns-1881.awsdns-43.co.uk  ns-1438.awsdns-51.org  ns-962.awsdns-56.net  ns-34.awsdns-04.com | No | Simple |
| cheeseheadhosting.nl | SOA | - | ns-1536.awsdns-00.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400 | No | Simple |
| \*.cheeseheadhosting.nl | A | - | 172.17.1.9 | Yes | Multivalue |

## 3.2.9 Elastic Container service

### 3.2.9.1 Task definitions

**Web-cluster-database-image**

* Container definition (MYSQL)
  + Defines the docker image "insently/project-s3:db".
  + Allocates 0 CPU units.
  + Maps container port 3306 to host port 3306 for TCP traffic.
  + Essential flag is set to true.
* Volume Configuration
  + Defines volume named project-database.
  + Configures EFS volume with a specific file system ID and root directory, /.
* Mount point
  + Mounts the project-database volume to the container "/var/lib/mysql" path, to persist data.
* Logging configuration
  + Specifies AWS CloudWatch logs as the log driver.
  + Sets up log options including log group creation, group name and stream.
* Task role and execution role
  + Assigns IAM role “ecsTaskECStoEFS” for both tasks and execution roles.
* Network mode
  + Network mode is specified as “awsvpc”.
* Resource configuration
  + Requires compatibility with Fargate.
  + Allocate 512 CPU units and 1024 MiB of memory.
* Runtime platform
  + Specifies CPU architecture as "X86\_64" and operating system family as "LINUX."

**Web-instance-project-image**

* Container definition (MYSQL)
  + Defines the docker image "insently/project-s3:latest".
  + Allocates 0 CPU units.
  + Maps container port 80 and 443 to host ports 80 and 443 for TCP traffic.
  + HTTP as the application protocol for both port mappings
  + Essential flag is set to true.
* Environment variables
  + Sets an variables MYSQL\_DATABSE with the value project-database.project-database.
* Logging configuration
  + Specifies AWS CloudWatch logs as the log driver.
  + Sets up log options including log group creation, group name and stream.
* Task role and execution role
  + Assigns IAM role “ecsTaskExecutionRole” for both tasks and execution roles.
* Network mode
  + Network mode is specified as “awsvpc”.
* Resource configuration
  + Requires compatibility with Fargate.
  + Allocate 512 CPU units and 1024 MiB of memory.
* Runtime platform
  + Specifies CPU architecture as "X86\_64" and operating system family as "LINUX."

**CR-(project name)-private-task**

* Container definition (MYSQL)
  + Defines the docker image "mysql:latest" docker image.
  + Allocates 512 CPU units and 1024 MiB of memory to the container.
  + Maps container port 3306 to host ports 3306 for TCP traffic.
  + Essential flag is set to true.
* Environment variables
  + Sets an variables MYSQL\_DATABSE with the value “(project-name)”.
  + Sets an variables MYSQL\_PASSWORD with the value “demo123!”.
  + Sets an variables MYSQL\_ROOT\_PASSWORD with the value “root\_password”.
  + Sets an variables MYSQL\_USER with the value “demo”.
* Logging configuration
  + Specifies AWS CloudWatch logs as the log driver.
  + Sets up log options including log group creation, group name and stream.
* Task role and execution role
  + Assigns IAM role “ecsTaskECStoEFSgeneral” for both tasks and execution roles.
* Network mode
  + Network mode is specified as “awsvpc”.
* Resource configuration
  + Allocate 512 CPU units and 1024 MiB of memory.

**CR-(project name)-public-task**

* Container definition (HTTPD)
  + Defines the docker image “"httpd:latest”.
  + Allocates 0 CPU units.
  + Maps container port 80 to host port 80 for TCP traffic.
  + Essential flag is set to true.
  + Mounts the EFS volume at "/usr/local/apache2/htdocs".
* Container definition (sFTP)
  + Defines the docker image " fauria/vsftpd:latest".
  + Allocates 0 CPU units.
  + Maps container port 21 to host port 21 for TCP traffic.
  + Essential flag is set to true.
  + Configures FTP-related environment variables.
  + Mounts the EFS volume at "/home/myuser/upload".
* Volume Configuration
  + Defines new volume, efs-mount.
    - Root directory: “/”
    - Transit encryption: enabled.
* Logging configuration
  + Specifies AWS CloudWatch logs as the log driver.
  + Sets up log options including log group creation, group name and stream.
* Task role and execution role
  + Assigns IAM role “ecsTaskECStoEFSgeneral” for both tasks and execution roles.
* Network mode
  + Network mode is specified as “awsvpc”.
* Resource configuration
  + Allocate 512 CPU units and 1024 MiB of memory.

### 3.2.9.2 Services

**Public dev cluster**

* **Project-frontend**
  + Basic configuration
    - Family: Project-frontend
    - Revision: latest
    - Desired tasks: 1
    - Min running tasks: 100%
    - Max running tasks: 300%
  + Networking
    - VPC: Customer VPC public.
    - Subnets:
      * Cheesehead-website-subnet-1b
      * Cheesehead-website-subnet-1c
    - Cheesehead-web-SG
    - Public IP: On
  + Load balancing: CH-Loadbalancer
  + Service autoscaling: CH-Frontend-AutoScaling
* **Project-database**
  + Basic configuration
    - Family: Project-Backend
    - Revision: latest
    - Desired tasks: 1
    - Min running tasks: 100%
    - Max running tasks: 300%
  + Networking
    - VPC: Cheesehead hosting VPC private.
    - Subnets:
      * Cheesehead-website-subnet-1b
      * Cheesehead-website-subnet-1c
    - Cheesehead-web-SG
    - Public IP: On
  + Service connect.
    - Namespace: project-database
    - DNS record
      * DNS record type: A
      * TTL: 15

**CR-(project name)-cluster**

* **CR-(project name)-public-service**
  + **Project-frontend**
    - Basic configuration
      * Family: web-instance-project-image
      * Revision: latest
      * Desired tasks: 1
      * Min running tasks: 100%
      * Max running tasks: 300%
    - Networking
      * VPC: Cheesehead hosting VPC public.
      * Subnets:
        + CR-(Project name) public subnet AZ-1-2-1 (eu-west-1a)
        + CR-(Project name) public subnet AZ-1-2-2 (eu-west-1b)
      * Cheesehead-web-SG
      * Public IP: On
    - Load balancing: CR-(project name)-Loadbalancer
    - Service autoscaling: CR-(project name)-Autoscaler
* **CR-(project name)-private-service**
  + Basic configuration
    - Family: web-cluster-database-image
    - Revision: latest
    - Desired tasks: 1
    - Min running tasks: 100%
    - Max running tasks: 300%
  + Networking
    - VPC: Cheesehead hosting VPC private.
    - Subnets:
      * CR-(Project name) public subnet AZ-1-2-1 (eu-west-1a)
      * CR-(Project name) public subnet AZ-1-2-2 (eu-west-1b)
    - Cheesehead-web-SG
    - Public IP: On
  + Service connect.
    - Namespace: project-database
    - DNS record
      * DNS record type: A
      * TTL: 15

## 3.2.10 API gateway

**API gateway: Deployment**

* API type: REST API
* API endpoint type: Regional
  + Resource: Website
    - Integration type: AWS Service
    - AWS Region: eu-west-1
    - AWS Service: Step Functions
    - HTTP Method: POST
    - Action: StartExecution
    - Execution role: AO-week8
    - Content Handling: Passthrough
    - Use default timeout: 299
    - CORS: Enabled
    - URL: https://89lcaztwz5.execute-api.eu-west-1.amazonaws.com/website/Website

## 3.2.11 Other peripherals

### 3.2.11.1 EFS

* **Project-database**
  + Performance mode: general purpose
  + Throughput mode: Elastic
  + Lifecycle management
    - Transition into infrequent Access (IA): 30 day(s) since last access
    - Transition into archive: None
    - Transition into standard: None
  + Availability zone: Regional
  + Automatic backups: enabled.
  + Networking
    - VPC: CheeseHead hosting VPC private.
    - Mount targets
      * Availability zone: eu-west-1c
      * Subnet ID: subnet-08dda45e9acdb60cc
      * IP address: 172.17.1.4
      * Security group: EFS-SG
* **CR-(project name)-efs-Mount**
  + Performance mode: general purpose
  + Throughput mode: Elastic
  + Lifecycle management
    - Transition into infrequent Access (IA): 30 day(s) since last access
    - Transition into archive: None
    - Transition into standard: None
  + Availability zone: Regional
  + Automatic backups: enabled.
  + Networking
    - VPC: Customer VPC private.
    - Mount targets
      * Availability zone: eu-west-1a-c (varies)
      * Subnet ID: Dynamic
      * IP address: Dynamic
      * Security group: EFS-SG

# 4. Naming connection

## 4.1.1 Username convention

The following format will be used for usernames:

User’s usernames will be structured in the following way:

* [first two letters first name] [first two letters last name]

The system will also have operator users. Operator users will follow the following naming convention:

* CH-Operator-xx

## 4.1.2 System naming convention

Systems will follow the following naming convention:

An example can be CH-NAT01

For tagging the customer infrastructure we have used the project name of the client.

An example where the project name is Documentation.

Documentation-NAT01

|  |  |  |  |
| --- | --- | --- | --- |
| Site | | Function | |
| CheeseHead  Customer | CH-  CR- | Webserver | WEBxx |
| Database | DBxx |
| NAT instance | NATxx |
| Operator instance | OPxx |
|  | |
|

# 5. Authorization plan

## 5.1 IAM policies

### 5.1.1 ECStoEFS

* **Effect**
  + Allow
* **Action**
  + elasticfilesystem:ClientMount
  + elasticfilesystem:ClientWrite
* **Resource**
  + arn:aws:elasticfilesystem:eu-west-1:349962368193:file-system/fs-06b02f4e9c57bbdbd
* **Entities attached**
  + ecsTaskECStoEFS

### 5.1.2 ECStoEFSgeneral

* **Effect**
  + Allow
* **Action**
  + elasticfilesystem:ClientMount
  + elasticfilesystem:ClientWrite
* **Resource**
  + arn:aws:elasticfilesystem:eu-west-1:349962368193:file-system/\*
* **Entities attached**
  + ecsTaskECStoEFSgeneral

### 5.1.3 LambdaSSM

* **Effect**
  + Allow
* **Action**
  + Statement 1:
    - Effect: Allow
    - Action:
    - ssm:SendCommand
    - ssm:GetCommandInvocation
    - Resource: \*
    - Sid: AllowSSMExecution
  + Statement 2:
    - Effect: Allow
    - Action: ec2:DescribeInstances
    - Resource: \*
    - Sid: AllowEC2DescribeInstances
* **Resource**
  + \*
* **Entities attached**
  + LambdaSSM

## 5.2 IAM roles

### 5.2.1 ecsTaskECStoEFSgeneral

* **Permissions policies**
  + AmazonECSTaskExecutionRolePolicy
  + ECStoEFSgeneral
    - This role allows connections to newly created EFS to store the customers database on.
* **Trust relationships**
  + Effect: allow
  + Principal: Service, ecs-tasks.amazon.com
* **Connected to**
  + Customer public task definitions.

### 5.2.2 ecsTaskECStoEFS

* **Permissions policies**
  + AmazonECSTaskExecutionRolePolicy
  + ECStoEFS
    - This role allows connection to a specific EFS where we store our database.
* **Trust relationships**
  + Effect: allow
  + Principal: Service, ecs-tasks.amazon.com
* **Connected to**
  + Web-cluster-database-image

### 5.2.3 LambdaSSMrole

* **Permissions policies**
  + AmazonSESFullAccess
  + AmazonSSMFullAccess
  + AmazonSSMManagedInstanceCore
  + AWSLambdaBasicExecutionRole
  + LambdaSSM
    - This role allows Lambda to send ssm:SendCommand & ssm:GetCommandInvocation commands to EC2 instances.
* **Trust relationships**
  + Effect: allow
  + Principal: Service, lambda.amazonaws.com
* **Connected to**
  + LambdaTriggerTerraformSSM

# 6. Security

## 6.1 Security groups

|  |  |  |  |
| --- | --- | --- | --- |
| CheeseHead-db-SG | | | |
| Inbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| MYSQL/Aurora | TCP | 3306 | 172.17.1.0/28 |
| MYSQL/Aurora | TCP | 3306 | 172.16.1.0/28 |
| MYSQL/Aurora | TCP | 3306 | 172.16.254.28 |
| SSH | TCP | 22 | 172.17.254.0/28 |
| ALL ICMP -IPv6 | ICMP | All | 172.17.254.0/28 |
|  |  |  |  |
| Outbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| IPv4 | All | All | all |

|  |  |  |  |
| --- | --- | --- | --- |
| CheeseHead-NAT-SG | | | |
| Inbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| All | TCP | All | 172.17.1.0/28 |
| All | TCP | all | 172.17.2.0/28 |
| SSH | TCP | 22 | 172.17.254.0/28 |
| ALL ICMP -IPv6 | ICMP | All | 172.17.254.0/28 |
|  |  |  |  |
| Outbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| IPv4 | All | All | all |

|  |  |  |  |
| --- | --- | --- | --- |
| CheeseHead-op-SG | | | |
| Inbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| All | TCP | All | 0.0.0.0/0 |
| SSH | TCP | 22 | 172.17.2.0/28 |
| SSH | TCP | 22 | 172.17.254.0/28 |
|  |  |  |  |
| Outbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| IPv4 | All | All | all |

|  |  |  |  |
| --- | --- | --- | --- |
| CheeseHead-web-SG | | | |
| Inbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| MYSQL/Aurora | TCP | 3306 | 172.17.1.0/28 |
| Custom TCP | TCP | 21-22 | 0.0.0.0/0 |
| SSH | TCP | 22 | 172.17.2.0/28 |
| HTTPS | TCP | 443 | 0.0.0.0/0 |
| HTTP | TCP | 80 | 0.0.0.0/0 |
|  |  |  |  |
| Outbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| IPv4 | All | All | all |

|  |  |  |  |
| --- | --- | --- | --- |
| EFS-SG | | | |
| Inbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| All | TCP | All | 0.0.0.0/0 |
|  |  |  |  |
| Outbound | | | |
| Type | **Protocol** | **Port range** | **Source** |
| IPv4 | All | All | all |

## 6.2 CloudWatch

CloudWatch Dashboard

* + CH Frontend
    - Memory Utilization
    - CPU Utilization
  + CH Load Balancer
    - Target Response Time
  + CH Backend
    - Memory Utilization
    - CPU Utilization
  + CH-EFS-Database
    - Storage in Bytes
  + CHNAT02
    - Network in and out
  + CHOP2
    - Network Packets in and out
  + CH Instances
    - CPU Utilization
  + Lambda Functions Data Tables (Errors, Invocations, Throttles Execution Duration Concurrent Executions).
    - Lambda Trigger SSM
    - Step Function
    - Lambda Function step function
  + SSM Commands (Succeeded, Failed, Timed Out)
  + CH Bucket Number of Objects & Storage Used
    - [www.cheeseheadhosting.nl](http://www.cheeseheadhosting.nl)
    - Ch-bt01
  + Certificate – Days to expiry
    - [www.cheeseheadhosting.nl](http://www.cheeseheadhosting.nl)

A screenshot of a computer

Description automatically generated

VPC Flow Logs

* Cheesehead Hosting VPC Public
  + Web-vpc-flow-reject
    - Destination Type: cloud-watch-logs
  + Web-vpc-flow-accept
    - Destination Type: cloud-watch-logs
  + Web-vpc-flow-all
    - Destination Type: S3

## 6.3 CloudTrail

## Trails

SNOWeek16 (Used for case study from assignment)

* + Collects Management, Data and Insight events.
  + Saved to S3 Bucket (snoweek8/CloudTrail) for further querying with Athena or for sending logs to a CloudWatch log group.
  + SNS Notification Delivery Enabled
    - Alarms
      * DDoS attaclk detection alarm.
      * Over budget monthly alarm.
  + Athena Table
    - Cloudtrail\_logs\_snoweek8\_cloudtrail
      * Eventversion (string)
      * Eventtime (string)
      * Eventsource (string)
      * Eventname (string)
      * Awsregion (string)
      * Sourceipaddress (string)
      * Useragent (string)
      * Errorcode (string)
      * Errormessage (string)
      * Requestparameters (string)
      * Responseelements (string)
      * Additionaleventsdata (string)
      * Requestid (string)
      * Eventid (string)
      * Resources (string)
      * Eventtype (string)
      * Apiversion (string)
      * Readonly (string)
      * Recepientaccountid (string)
      * Serviceeventdetails (string)
      * Sharedeventid (string)
      * Vpcendpointid (string)
      * Tlsdetails (string)

A screenshot of a computer

Description automatically generated

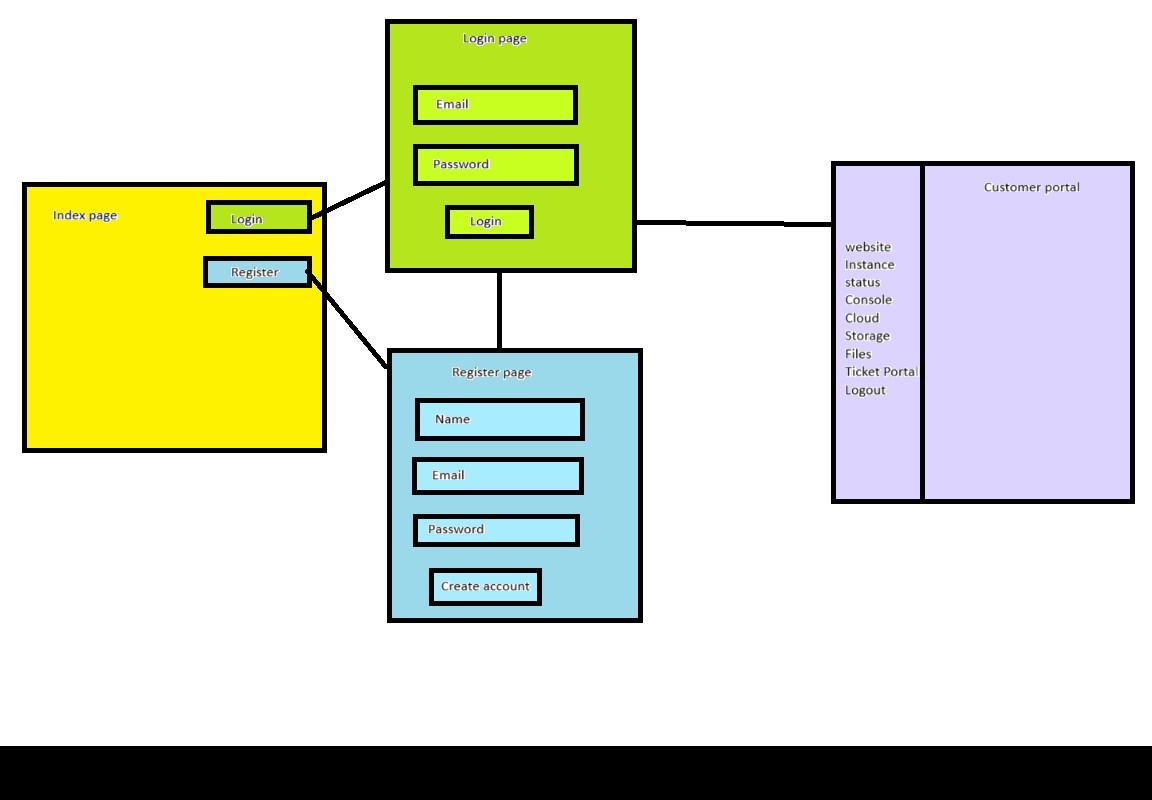
A screenshot of a computer

Description automatically generated

# 7. Website

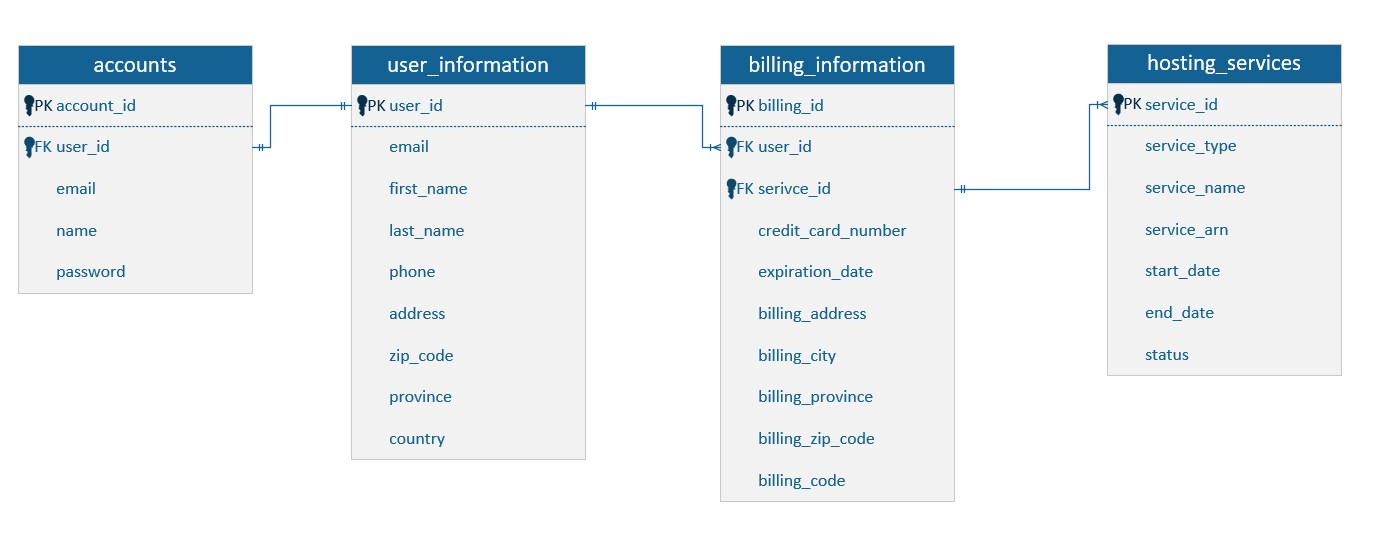
## 7.1 Website wireframe

In the image below you can see a basic wireframe explaning the layout of the website. The index page has a login button that is connected to the login page. It also has a register button that is linked to the register page. When creating an account you get forwarded to the login page. After logging in the customer can see the customer portal.



## 7.2 Database design

In the image below you can find our database design for the customer information for our own website.



# 9. References