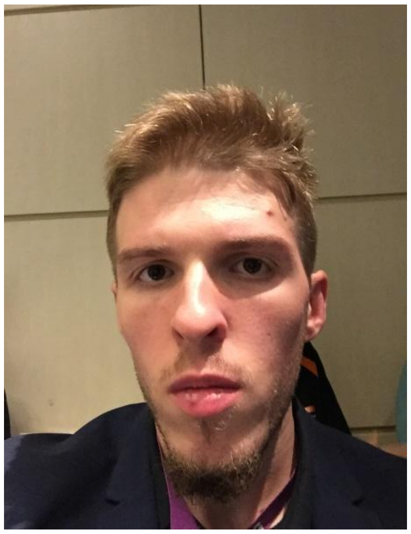
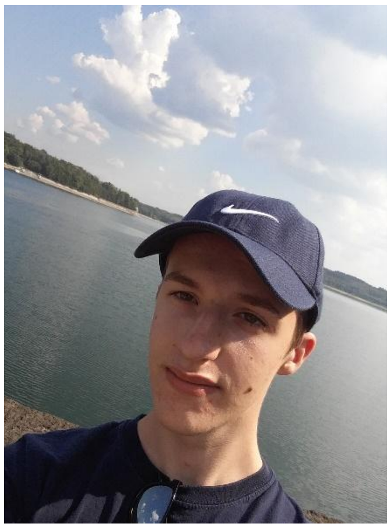
Handoff team B3





elision

Contents

Project Description

Elision is a company that specializes in e-commerce. They mainly use SAP in their project to create those webshops, but the problem with SAP is that it can get expensive. As a result, we were tasked with creating a webshop for elision, with their guidance, without using SAP. To do this we need to use a headless approach. The backend had to be written in java using spring boot and the frontend had to be made using NextJS. Besides these two requirements we also decided to use algolia which is used for searches and recommendations and contentful as our content management system.

Everything from our product info to the privacy policy is stored in contentful so that they can be easily edited by the owner of the webshop.

Webshop

Credentials

e-mail: elision@mail.com

Password: elision

How to prepare to run the development server/build the application:

To set everything up, the user needs to first open up a terminal in the root of the project folder (this can be done via an IDE's terminal or using the command prompt). The user needs to run the command *npm install* or *npm i* in order to install all of the packages as instructed in the *package.json* file. An installation will start. If that is done the user can proceed to run *npm run dev* to spin up the development server, or *npm run build* to run a production build of the application. If *npm run dev* is the case, the user can now browse to *localhost:3000* (if port 3000 is not available then another one will take its place which should be the port in *localhost<port-number>*).

How to use the webshop:

The top of each page contains a navigation bar to reach frequently used pages. The bottom of each page has a footer which has less frequently used links.

NavBar:

The (elision) logo at the left of the navbar redirects to the homepage.

The search bar: click and fill in search terms, a list of items get displayed and shortens with more precise search terms. Clicking one of these items redirects to the product page for this item.

The about link redirects to the about page with information about the webshop as well as some contact details.

The categories link redirects to a page with a list of all categories. These categories can be clicked to show all products in that category, afterwards these products can be clicked to navigate to the product page of the clicked product.

The log-in button redirects to the log-in page where you can log in. When logging in was successful you get redirected to the home page and the log in button gets changed to a log out button. The log-in page also contains a link that redirects to the sign-up page in case the user still needs an account.

When signing up for an account you need to fill in a name, e-mail address and password. When logging in you use your e-mail address and your chosen password.

Footer

Under the title “Follow us” a couple of social media icons are displayed (Facebook, Instagram and Twitter) clicking these icons opens the clicked social media account of elision in a new tab.

Under the title “legal” are two links, one for the privacy policy and one for the Terms & conditions. Clicking one of the links redirects to the respective page with the information for the privacy policy or the terms and conditions.

At the bottom of the footer there is a bit of copyright information.

Product page

When looking at the product page of a specific product you see an image of the product on the left followed by the price and description to the right of the picture. The name of the product is displayed on top of the other information. Below the image you either see “log in to buy this product!” when not logged in or an amount which you can increase or decrease above a buy now button which can be clicked and then you get redirected to the payment page.

Payment page

After pressing the buy now button you end up on this page. On this page you fill in your payment details and can see the total price to be paid in the confirmation button. After successfully paying you get redirected to a success screen, if something goes wrong with the payment you get redirected to a refused screen.

How to run Unit Tests

In order to run Unit Tests, the user first needs to make sure that, in the root of the project, *npm i* or *npm install* has been run in a terminal to install all the packages as instructed by the *package.json* file. The tests are run in Jest which is a JavaScript testing library. If that was done, a test file is already present which means the user can proceed to, in the same terminal, run the command *npm run test*. This command will start the test script and run through all the present unit tests. Once it’s done, a table of coverage will be made available along with the verdict of each test that was run. Coverage is presented in a wide range of variability with *% Lines covered* being the most important here.

Contentful

Credentials

Username: r0752792@student.thomasmore.be

Password: F5&KxFYVzMzy

How to use contentful

After logging in on contentful and navigating to the content tab at the top you can sort everything with the dropdown menu next to content type. After selecting the desired content type, you can select the entry you want to edit and make the needed changes. You can also use the search at the top to look for one specific entry to make changes that way.

When you want to create a new item, you click add entry in the top right corner. After clicking the button you're prompted which type of content you want to create, pick the desired content type and fill in the required fields.

Some pieces of content have a page and an item content type for example, privacy policy page and privacy policy item. An item is one part of the privacy policy, a title and a body. A privacy policy page consists of multiple privacy policy items.

Algolia

Credentials

Username: r0831309@student.thomasmore.be

Password: Password4TeamB3!

How to use algolia

After logging in to Algolia, in the left-most part of the screen the user can select between search and recommendation. Clicking on search will open up any index if available. An application and index can both be selected at the top of the screen to make sure the adequate index for the application is selected for proper view.

This index contains data that is used in the Front-end by an application search component as a client in order to sift through the search items available as the user types a search query (mind you, this is data that was transformed when retrieved from contentful, but the data of the search client comes from this Algolia index).

Adyen

Credentials

Account: Company.Elision

Username: TM-Team-B3

Pass: Password4TeamB3!

Api-key: AQEphmfxK4vOahREw0m/n3Q5qf3VbYdEHppFVig6Nm4dH/xx82FfjRWnSZMQwV1bDb7kfNy1WIxIIkxgBw==-IcqsXI/tOuiDct0tli6Zp42HT6rbHMsYNHTttaA70U4=-S5E,%PG#pQWUrDUT

How to use Adyen

Login to Adyen.com into the test environment. In the left sidebar you can find the title "developers" and under that "API credentials". Within the api credentials tab you can select the credential of team B3 (called ws_40116). Here you can adjust the settings for the credential. ATTENTION: if you regenerate the API key you have to replace every .env file with the right key in it, and you cannot see the full key again after generation. Below we can find the title "Authentication", here all the allowed origin URLs are listed to get through the CORS errors.

In settings and then payment methods, here are the methods listed that are to be used. The credit card only works but if we disable the others, other teams will also have these changes applied to them. In the backend under adyenController and then under the paymentMethods mapping, there is a block with "setBlockedPAymentMethods" that disables the payment methods in code locally.

After a transaction is completed, the transaction will show in the transactions tab under payments, the payments stat with "payment:" the mail of the paid user and a unique UUID.

Backend

Credentials

The backend uses the same .env file as the frontend for ease of use. Not all fields are needed but it should have at least following fields:

ADYEN_APIKEY=AQEphmfxK4vOahREw0m/n3Q5qf3VbYdEHppFVig6Nm4dH/xx82FfjRWnSZMQwV1bDb7kfNy1WIxIIkxgBw==-IcqsXI/tOuiDct0tli6Zp42HT6rbHMsYNHTttaA70U4=-S5E,%PG#pQWUrDUT

ADYEN_MERCHANT_ACCOUNT=Elision_Stage2022_TEST

If you want to run the backend locally you have to add the file “vockey4.pem” in the backend root, with the content:

```
-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEAm7FXtnpS2D4FALD7QJjG+c92o5VzpIRepp11B3DVmLGQ59Hp

vYEH7KKKHYCrsfvXI3Ftu2ZvLI2HAmf0PypcM68pFinE2JnCX2JzE4ZFY7TDyidJ

QSL+FMvcXKylgCLnsKMPGtWcA2hdGPPONOyLDDCtXNQWSJ3gA7KEdJ3I9D4i/Pww

gamsT8cSQYbWo13Mq5UODOA0xLBb/y4OSzMg7acMmPz62xh9eAQl+InSipdyaI/3

mE7OFJXwthK1vrKwf0SGFrQyjQnMlPPDzdyqiBz7SCPqHnadHYcUTOHK3nAsNbME

Du5MLvhlkPNw44sOzCD3+xWMh5mtdUrdyWaCkQIDAQABAoIBAA3tEPFrg5mAAqaL

Lg4bHEEgA3aiWZOvm1+/2cXum8xqpZF2vzeKIRFTgJMe/TyQUDAFZ1FJ2kvb5OWc

MsarnU+qLTykQ800A6TQg4aNjgtbEeA1H2JKJPvrOCy57JvHg33TN/iofqn3Xcxr

klg9iDpw6DDTb6LTLERoZVyg+GaQPEIFRdbLr1WTwSt/xqKYoVBo3WmLdRRTxYTi

FCmA4hfQ5ZkAD1Xg29YQFqIMShPD3l/rFhIelvhAcL0XcbjbMN8CPGBA4v8XmtAd

lTjnJawwU+sWig6QLDVD88zReBs9BR2vpRourmZ0JMcQFmbRM+gYiIOu4GUjDEO/

nInZAKECgYEA4xOdCUO0Wrnbn1WC4OWFjB/Y43T4ufhU36CiqSKgfzVI21bgrDfO

nFhgYrXSEo9errBZTgIiaLEBjbg9MMdVMzR0/dRZLaD1t1qBE55DsVR0q10WkKuO

7kdRXp+oSExZwB9A3JB3MCWk8o8h94443coAx0sSakxSMLD+IIOrsF0CgYEAr4YV

h6oyMMb1X5CG4UmJeS9+bFgbPxDL5uXxRyGKLVmO9Hj59490FAD8YnrATdKcJ2Ln

WzAg/naPA/vRND+7ZOo/yzqznfJXQNMEfcSssAm7so2TdQA29LwswRW3xDiZmECC

+hVm7P8rp58rzyayletPamlKqd+YIV9AD+v2R8UCgYEAoo3Q1ef8vRHdkYNCbMKB

xpypw7Jht6d5AplYuCFMqaO4YyQfM8nDgxKU9TTZjGXLztv5IinO/gjwZsZkL34S

lNRRh7+yk0Jtg/MrAaNDLDecSWwbEjdt2098hXNfoVneyfTVls9oMzRPj0A9fZz3

QnYv9nrcVPILyEV7tkTr7BUCgYAKFhRqpVcZAnXbNfzqOUWTFy2WoEGAkECHVjNV

ORItpPXYCpOirDWpaJ0YM6GVV0Lt6HsO/GcI+FsjiXbuH7NvCx77WLY1n1VwPjAF

4iViQEzu57/Pe2GtEpsnxU24EuCQpTRhlpVBjTA/A6CK2NhuUQVRQheuR1EAqe6O

e6q9WQKBgBVqTKJBR5XQaB2tZkm377xq4Pc0P5ge6A5vs8qvQHqXrjwjWFFnIvtI

DbCqU9ZJWOCjBetf8lnB/FEVz4w0Icse7lMzWNPu+JdbG8WucGLJcAwcou48JKQw

gvpon9WomI4mAaOpMCNfpqSxwivtgfBkhESTnabcSfy9U5Ak7uxI

-----END RSA PRIVATE KEY-----
```

You also have to run a ssh connection in the terminal, this connection is ever running so you might want to use a new one.

```
ssh -f ubuntu@52.21.0.223 -L
127.0.0.1:3309:awseb-e-javeqbppnv-stack-awsebrdsdatabase-1c5pvpzfmpuq.cb3xkqkbupg1.us-east-1.rds
.amazonaws.com:3306 -N -i " vockey4.pem "
```

this creates a tunnel to the hosted sql via a connection string. You need to add the sql credentials to your environment files:

```
SQL_DATABASE_NAME=Elision
SQL_USERNAME=root
SQL_PASSWORD=Xcs6C68C!4*FDhCT
```

and you need to comment the bottom block of “application.properties” and uncomment the top 3 lines.

The backend works via API calls, it uses the controller-service-repository pattern. URL calls come in to the controller, the logic is performed in the service and the queries are handled in the repository.
the controller advisor catches custom and defined errors that happen in the annotated controllers.

Users are mapped to a secure user DTO if they are to be sent to the front end. They have no password connected to them, any password check gets done in the backend.

Adyen flow: first the payment methods get generated, this is made with a new client and the merchant account. As it returns a response entity filled with the session.
after the methods it will initialize a payment session with the amount to pay and a reference with the logged in user their mail.

Infrastructure

Infrastructure plan

In this part we will discuss the overall assignment, what AWS cloud has to do with the assignment, who is the customer? What are the requirements and what are the expectations.

The infrastructure

The AWS cloud environment needs to provide the infrastructure to host the web application. It also needs to have a relational database to store the data that is generated by the web application. A new version of the website needs to be deployed through a CI/CD pipeline which checks if there are any problems in the code. The web application must be available online for all the users. All the pages need to have an SSL certificate.

Requirements

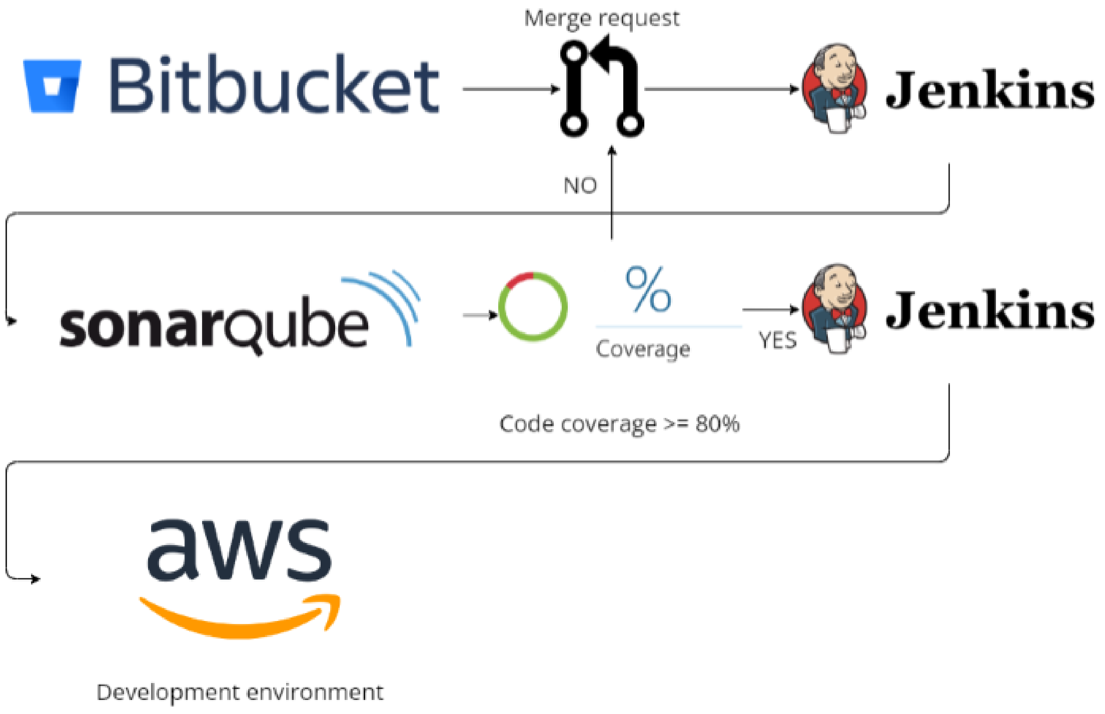
We need to build a cloud environment in AWS. We need to use a relational database like MySQL. Furthermore, we need to use Elastic Beanstalk to manage the EC2 instances and S3 buckets. The budget for the entire environment is 200 dollar per month of AWS credits. The website needs to be publicly available by using Combell. All the different pages need to have an SSL certificate. All the resources in the cloud need to be monitored.

Deployment workflow

In this part we will explain the deployment workflow of our project.

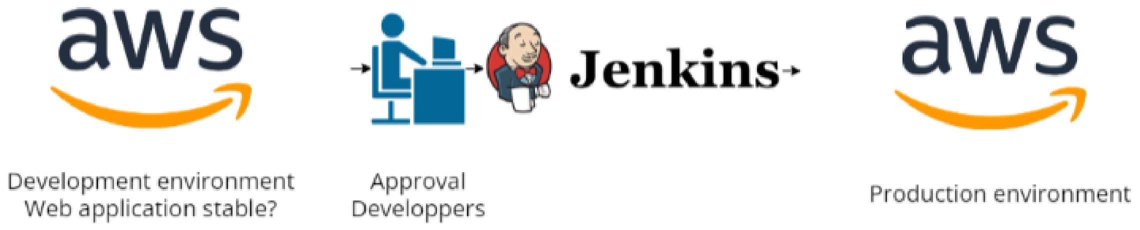
New features

When there is a new feature merged into our development branch a Jenkins pipeline will start. This pipeline will copy the code from our development branch and send it to SonarQube. When there is less than 80% code coverage by unit tests the pipeline will stop. If there is more than 80% code coverage the pipeline will deploy the frontend and the backend to the AWS development environment.



Stable development environment

When our development environment is stable, we can trigger a pipeline which will send everything to our production AWS environment.



Infrastructure realisations

In this section, we will explain the different parts of the infrastructure.

Bitbucket

All our code will be stored in a Bitbucket repository. When there are new changes merged into the development branch a Jenkins pipeline will start. This pipeline will first send the code to SonarQube. When there is more than 80% code coverage the code will be deployed to our AWS development environment.

Manually build environment
We wanted to use Terraform scripts to automate the setup of our environment. This seemed to be a project on its own, so we set up the AWS infrastructure via the graphical user interface.

AWS Elastic Beanstalk
This is an AWS feature to manage EC2 instances (VM's) and S3 buckets (Storage). This feature was obligated to be used in our project.

EC2 instances
Our frontend application runs on a EC2 and our backend EC2 will run on a different EC2 instance.

S3 bucket
Our Docker configuration and Java configuration will be stored inside a S3 bucket.

Database
For the database in production, we used a MySQL database on a RDS instance. We use a different MySQL database on a different RDS instance for our development data.

Autoscaling group
For our production frontend we will spin up 1 EC2 instance, we will also spin up a different EC2 for the production backend, 1 EC2 for the development frontend and 1 EC2 for the development backend.

All these EC2's will be placed in their own autoscaling group. These groups make sure that there is always an instance running. If the instance goes down, the group will spin up a new instance. We want this because we want our web application to be redundant and scalable.

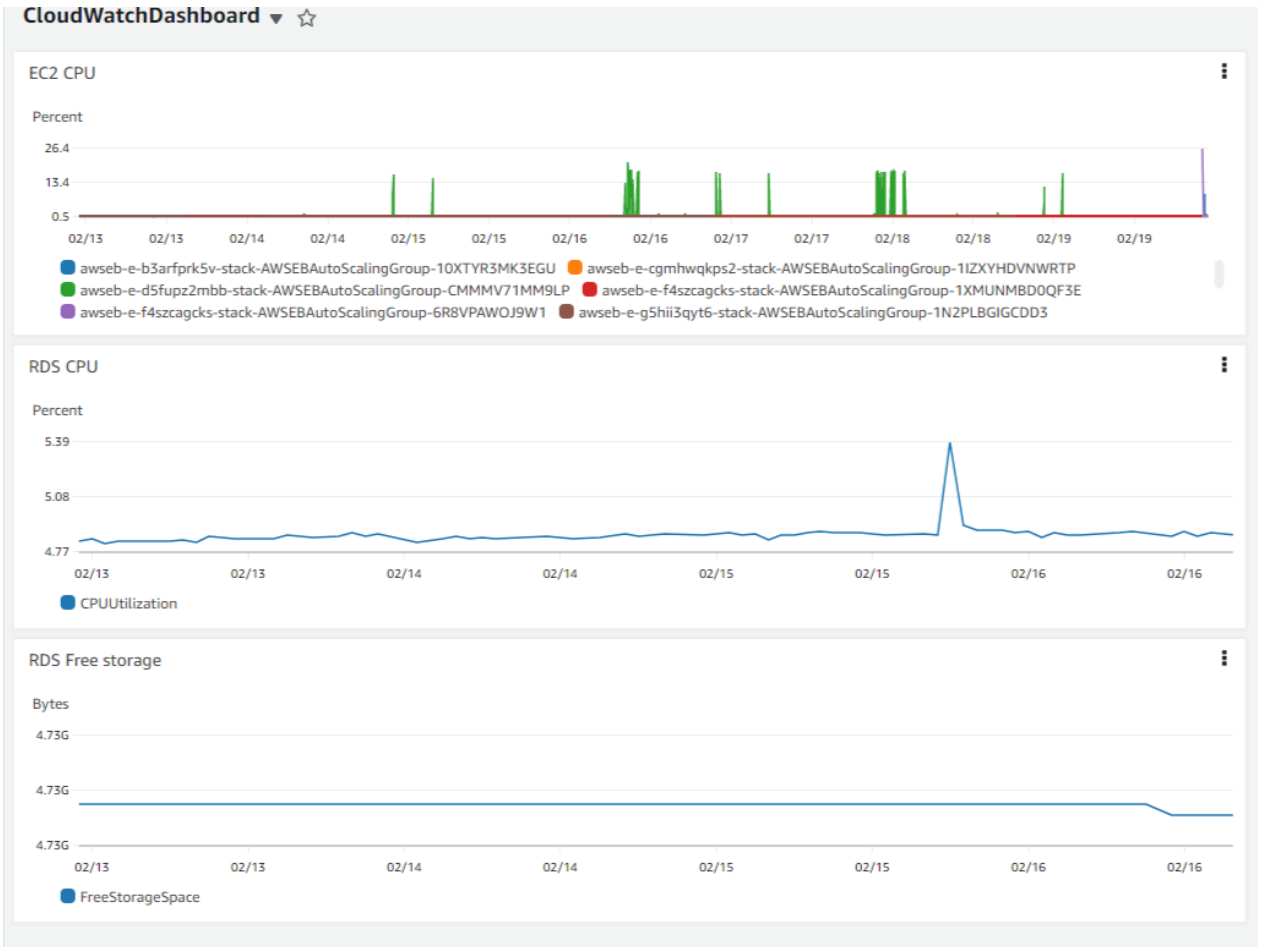
When you need more than one instances, you can configure the autoscaling group in a way that there are always two running instances. We are building our environment for small to medium companies, so we want to provide a way to scale up or down very easily.

Load balancer
If you want to spin up more than one EC2 in the frontend or the backend we will provide a load balancer in each auto scaling group. So, you can easily scale the production or development, frontend and backend.

AWS Backups
AWS automatically takes a snapshot every day of our database. We will use these snapshots if there goes something wrong with the data in our database.

AWS CloudWatch
When something goes wrong, we want to know it. We are going to use AWS CloudWatch to monitor the state of our instances and our RDS instances. We also want to monitor the CPU utilisation and storage of our EC2 instances. Likewise, we also want to monitor the storage of our RDS instances.

- EC2
    - CPU
- RDS
    - CPU
    - Storage



AWS Simple Notification Service (SNS)

This service will send an email when one of the above metrics triggers and alarm. By example EC2 CPU is over 80%, an email will be sent to us saying the CPU of the EC2's is too high. This service is free is you use maximum 1000 requests and maximum 1000 emails per month, we do not expect to need this much.

VPC

We don't want our development environment to be public, so to make our development environment private we will use private and public subnets.

Private subnet

In this subnet we will place our development environment, database and file storage. This subnet is not accessible from the internet, but our resources can reach the internet when they need to update something. This traffic goes through the NAT gateway.

Public subnet

We want our website to be public so this will be published inside a public subnet. Traffic will flow in and out this subnet using the internet gateway.

AWS Lambda

We will use this function to shut down the EC2 and RDS instances every evening and turn them back on every morning. In production the website needs to be available 24/7. But in our development phase this only needs to be available during the day. So, in the evening we will turn EC2 and RDS off to save costs.

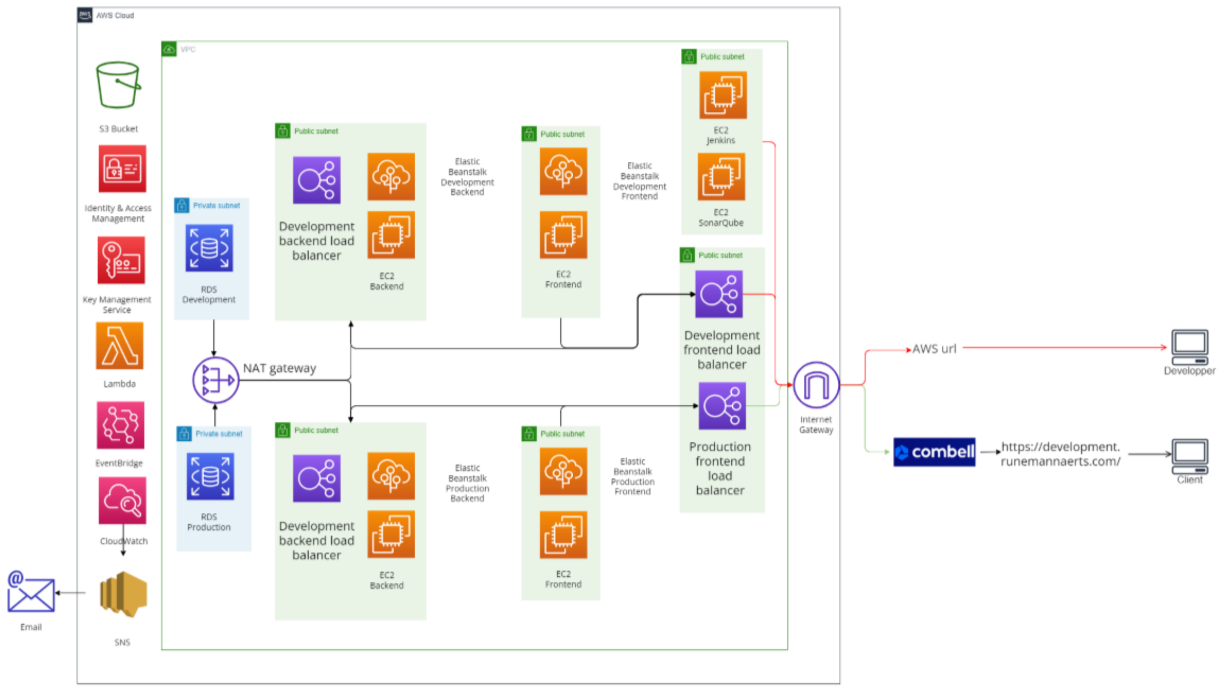
AWS EventBridge

This service will trigger the event to shut down and start up the resources.

AWS IAM

IAM stands for Identity and Access Management. Our Lambda functions needs an IAM role to specify the access of the Lambda function.

Architecture

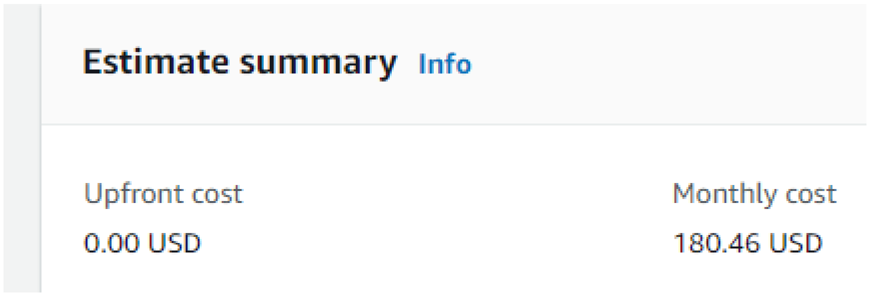


AWS Cloud
VPC
S3 Bucket
Identity & Access Management
Key Management Service
Lambda
EventBridge
CloudWatch
SNS
Email
Private subnet
RDS Development
NAT gateway
Private subnet
RDS Production
Public subnet
Development backend load balancer
EC2 Backend
Elastic Beanstalk Development Backend
Public subnet
EC2 Frontend
Elastic Beanstalk Development Frontend
Public subnet
EC2 Jenkins
EC2 SonarQube
Public subnet
Development frontend load balancer
Production frontend load balancer
Public subnet
Development backend load balancer
EC2 Backend
Elastic Beanstalk Production Backend
Public subnet
EC2 Frontend
Elastic Beanstalk Production Frontend
Internet Gateway
AWS url
Developper
combell
https://development. runemannaerts.com/
Client

Cost structure

In this part, you can find the link to our AWS cost estimation.

https://calculator.aws/#/estimate?id=bd3ff3b24909319ddb037c83223f866ce7513df8


Estimate summary

Upfront cost	Monthly cost
0.00 USD	180.46 USD

HTTPS

The reason why we use https for the website:

Overall, using HTTPS is a good practice that can help protect both your users and your website. It can help prevent attacks, boost your visibility, and give your users a sense of security when they interact with your site.

SonarQube

SonarQube is a tool for continuous inspection of code quality. It is used to analyze the quality of the code written in various programming languages.
The tool uses static code analysis to identify potential bugs, code smells, vulnerabilities, security issues, and other code quality issues.

Credentials:
ip: http://52.22.225.143:9000
username: admin
password: syK6l3%&ZUz9

Jenkins pipeline

We used Jenkins primarily for continuous integration and continuous delivery (CI/CD) of software applications, we also used it to add/change certain files in our pipeline. We also made use of Jenkins to deploy our code to SonarQube to get tested for the above mentioned vulnerabilities, security issues and other possible code issues

Credentials:
ip: http://3.210.64.159:8080
username: admin
password: y@yEK!E2H325v4Z9

Conclusion

This was a really challenging project with a lot of ups and downs. We tried to implement the basic features that were required. We had a lot more difficulties than we had expected in the beginning of the project. Looking back our first project plan was a bit ambitious, but we tried the best we could to make it work.

We have a result that we can show off, it's not what we had hoped for, but we are proud on what we have built and the work we have put into this project. We would like to thank Elision and Nfuse for this opportunity to create a project for a real company. We would also like to thank Brent Eerlingen, Jordy Verbeek and Arif Eredjeb for their support and coaching. At last, we want to thank Bram Heyns for coaching us and boosting our morale.