Final Documentation

Project 4.0

Camerlynck Michael ACS r0831153
Claes Mats CCS02 r0842906
Liekens Jentel CCS01 r0835618
Van Genechten Luca APP01 r0849476
Van Reusel Senne APP01 r0903904

CAMPUS
Geel

Project 4.0

Academic year 2022-2023

Table of content

Content

Table of content 3

1 The Project 6
1.1 Project Description 6
1.2 Project Goals 6

2 Technologies Infrastructure 7
2.1 Infrastructure 7
2.2 AWS Beanstalk 7
2.2.1 Why do we use AWS Beanstalk? 7
2.2.2 What is AWS in general? 7
2.2.3 Why AWS Elastic Beanstalk? 8
2.2.4 What is AWS Elastic Beanstalk? 8
2.2.5 Advantages 8
2.2.6 Components of AWS Elastic Beanstalk 9
2.2.7 How does EB in AWS work? 10
2.2.8 Architecture of AWS Elastic Beanstalk 11
2.2.9 Design of EB 12
2.3 AWS VPC 13
2.3.1 Why do we use a VPC? 13
2.3.2 What is a VPC? 13
2.3.3 Why do we need a VPC? 13
2.3.4 Key Features of VPC 14
2.3.5 Network Configuration 14
2.4 AWS Load Balancing 14
2.4.1 Why do we use a load balancer? 14
2.4.2 What is Load Balancing? 15
2.4.3 Types of Load Balancers 15
2.4.4 Summary 15
2.5 AWS Auto Scaling 15
2.5.1 Why do we use auto scaling? 15
2.5.2 What is auto Scaling? 15
2.5.3 How it works 16
2.5.4 Summary 16
2.6 AWS EC2 16
2.6.1 Why do we use AWS EC2? 16
2.6.2 Why Amazon EC2? 16
2.6.3 Key Features of EC2 17
2.7 AWS RDC 17
2.7.1 Why do we use AWS ECR? 17

2.8 AWS RDS 17
2.8.1 Why do we use AWS RDS? 17
2.8.2 What has RDS to offer? 17
2.8.3 Key Features of RDS 17
2.9 AWS Backup 18
2.9.1 Why do we use AWS Backup? 18
2.9.2 What has AWS Backup to offer? 18
2.9.3 Key Features of AWS Backup 18
2.10 Jenkins 18
2.10.1 Why do we use Jenkins? 18
2.10.1 What is Jenkins? 19
2.11 Combell 19
2.11.1 Why do we use Combell? 19
2.11.2 What has Combell to offer? 20
2.12 SonarQube 21
2.12.1 Why do we use SonarQube? 21
2.12.1 What has SonarQube to offer? 21

3 How to set up the pipeline 22
3.1 Step 1: Cloning the project from bitbucket 22
3.2 Step 2: Building the backend .env file 22
3.3 Step 3: Building the backend 23
3.4 Step 4: Analysis by SonarQube 24
3.5 Step 5: Deploy Backend 24
3.6 Step 6: Logging into AWS ECR 24
3.7 Step 7: Building the frontend 25
3.8 Step 8: Pushing the fronted to the ECR 25
3.9 Step 9: Docker image pull from ECR 25
3.10 Step 10: Cleaning up the Jenkins server 26

4 Technologies Application 26
4.1 Spring Boot 26
4.2 Next.JS and React 27
4.3 Adyen 28
4.4 Contentful 28
4.4.1 Add data to contentful 29
4.4.2 Add a new page in Contentful 29
4.5 Mock Data 30
4.5.1 Getting the Data 30
4.5.2 Uploading Data to Contentful 31
4.5.3 Uploading Data to Algolia 31
4.6 Algolia 32
4.6.1 Search Configuration and Optimization 32
4.6.2 Events 32
4.6.3 Personalization 33

4.6.4 Machine Learning Models for Recommendations 33
4.6.5 Business Analytics 33
4.6.6 WebHook 34

5 Usage of the app 34
5.1 Landing Page 34
5.2 Navigation 34
5.3 Browsing Products 35
5.4 Product Detail Pages 35
5.5 Shopping Cart 35
5.6 Checkout 35
5.7 Leaving a Review 35
5.8 Rating System 35
5.9 Viewing Reviews 36
5.10 Conclusion 36

6 Promotional Video 36

7 Credentials 36

8 Conclusion 36

1 The Project

1.1 Project Description

Elision aims to create a state-of-the-art, composable e-commerce platform that is scalable, connected, and provides the best experience to customers. The platform will be built using Next.js, with React as the JavaScript library for the user interface. The backend will be Java-based, using the Spring (Boot) framework, with REST APIs facilitating communication between the database, backend, and frontend. The platform will be hosted in the cloud, and a relational database will be used.

External services such as search capabilities via Algolia, CMS via Contentful and recommendation engines will be chosen based on research, consulting with coaches, and evaluating their value proposition.

The project will be divided into two main specializations: App/AI development and Infrastructure. App/AI development will focus on developing the web application using Java, Spring Boot, and PI4J, setting up APIs and interfaces, integrating external services, and creating search capabilities, and recommendation engines. Infrastructure will focus on configuring and managing the cloud-based infrastructure, database, and addressing issues such as privacy, security, communication, backup, and database selection.

1.2 Project Goals

The primary goal of the project is to create a state-of-the-art, composable e-commerce platform that is scalable, connected, and provides the best experience to customers. This will be achieved by:

- Developing a web application using a progressive web app (PWA) framework, Next.js, and React as the JavaScript library for the user interface.
- Building a scalable and connected backend using Java, Spring (Boot) framework, and REST APIs that facilitate communication between the database, backend, and frontend.
- Selecting and integrating external services such as search capabilities via Algolia, CMS via Contentful, and recommendation engines that add value to the platform.
- Hosting the platform in the cloud and selecting a relational database to store and manage data.
- Configuring and managing the cloud-based infrastructure and addressing issues such as privacy, security, communication, backup, and database selection.

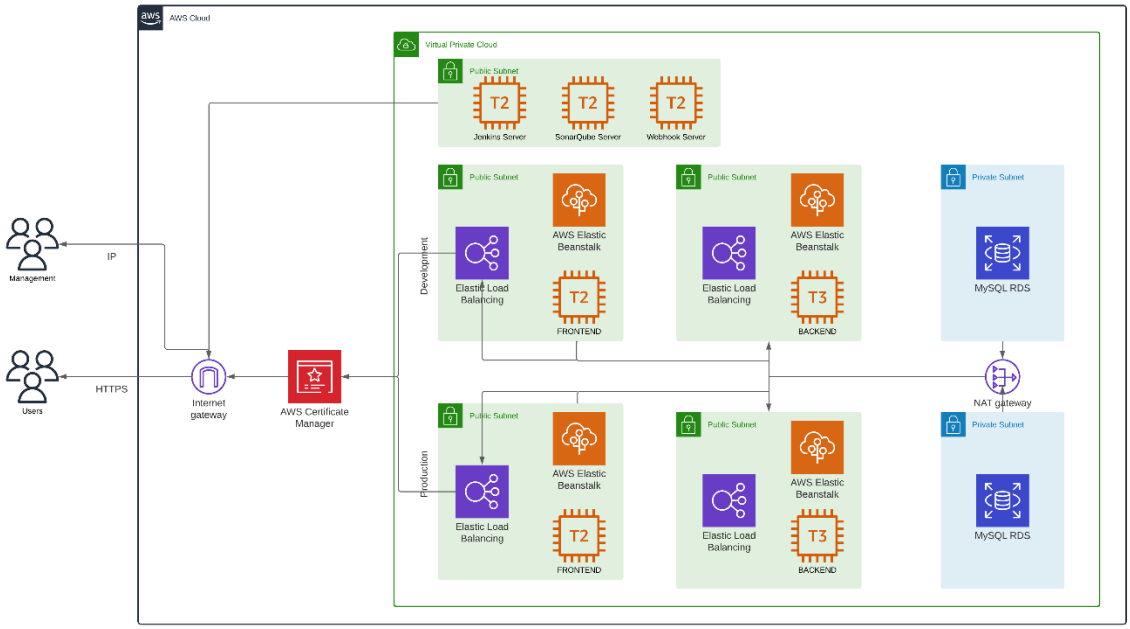
The successful completion of the project will result in a scalable, connected, and composable e-commerce platform that provides customers with an exceptional shopping experience. The platform will be

built using modern web technologies, hosted in the cloud, and integrated with external services that add value to the platform. The project will also provide an opportunity for students to develop their skills in web application development and cloud infrastructure management.

2 Technologies Infrastructure

2.1 Infrastructure

On the picture below you can see the schema of how our environment is build up. You can find every point we used in our environment in the points below.



2.2 AWS Beanstalk

2.2.1 Why do we use AWS Beanstalk?

When we are looking into our stack maybe one of the most important components is our AWS Elastic Beanstalk. This is the middle of everything because it will host our frontend and our backend. It will also spin up some automated functions to support the hosting. So will it setup our EC2 that host the real code, an auto scaling group and a load balancer. We will go deeper into this in the points on the next pages.

2.2.2 What is AWS in general?

Amazon Web services is a cloud provider that offers a variety of services such as compute power, database storage, content delivery, and other resources to help businesses scale and grow.

- Largest Cloud provider in the market

2.2.3 Why AWS Elastic Beanstalk?

Elastic Beanstalk makes it easy for developers to share their applications across different devices with less amount of time.

2.2.4 What is AWS Elastic Beanstalk?

EB is a service used to deploy and scale web applications by developers. Not only web applications but any application that is being developed by the developers.

EB supports various programming languages:

- Java
- .NET
- PHP
- Node.js
- Python
- Ruby
- Go
- Docker

For any other programming languages we can just make a request with AWS for arranging that specific language. (Not in our case)

2.2.5 Advantages

- **Highly scalable service**
  → whenever we require the resources in demand we can **scale up/down** the resources (flexibility we get in term of changing the type of resources whenever we need it)
- **Fast and simple to begin**
  → Have to focus on the development of an application, building an application and then we can just deploy the application directly using the Beanstalk
    - Every network aspect is been taken care of by the Beanstalk
    - Deploys application in the backend on the servers
    - Directly access your application using the URL or through the IP address
- **Quick Deployment**
  → Don't have to bother about the networking concepts
    - Focus on the application development
    - Upload your application & deploy → good to go!
- **Supports multi-tenant architecture**
  → We can have a virtual environments for separate organizations or the divisions within the organizations that will be virtually isolated
- **Simplifies operations**

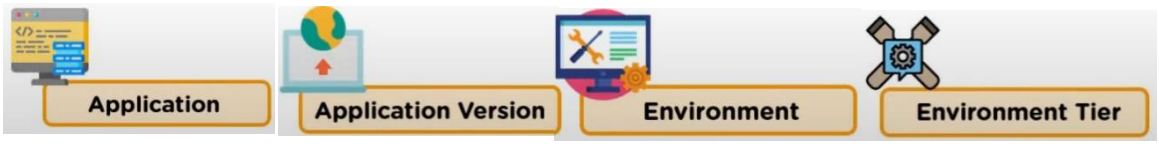
➔ When the applications has been deployed it becomes easy to maintain and support that application using the Beanstalk services itself

- **Cost-efficient service**
  ➔ The cost optimization can be better managed using the AWS beanstalk as compared if we are developing or if we are deploying any kind of an application or a solution on the on-prem servers

2.2.6 Components of AWS Elastic Beanstalk

There are some components associated with the AWS Beanstalk and it has to be created in the form of a sequence manner.

Important components which are required while developing an application:



1. Application

Refers to an unique label which is used as deployable code for a web application

➔ We deploy our web application = unique label

2. Application Version

Resembles a folder which stores a collection of components such as environments, versions & environment configurations

➔ Components are being stored using the application version

3. Environments

Only the current versions of the applications runs.

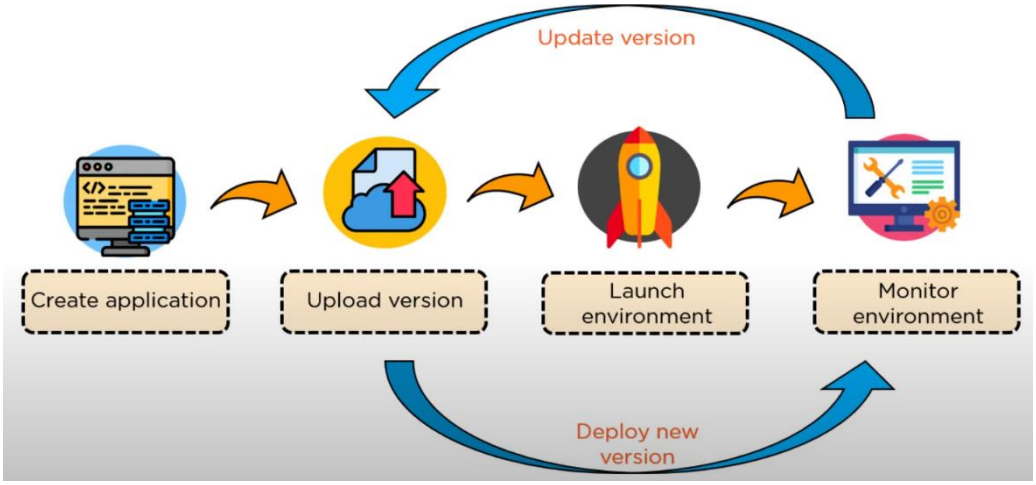
➔ If we want to run another version of an application we need to create another environment

4. Environment Tier

It basically designates the type of application that the environments runs on now

➔ Normally 2 types of environment:
  - Web node
  - Worker node

2.2.7 How does EB in AWS work?



1. Create application: Would be done by the developers. (Choose programming language)
2. Upload version: One the application is created, we have to upload the version of an application on the AWS.
3. Launch environment: Once the version is uploaded, we have to launch our environment (click on the buttons, nothing more)
4. Monitor environment: One the environment is launched, we can view that environment using a web URL or using the IP address.

2.2.7.1 What happens when we launch the environment in the backend?

EB automatically runs an EC2 instance (server) and using metadata the Beanstalk deploys our application within that EC2 instance.

- Don’t have to take care of the security groups
- Don’t have to take care of the IP addressing
- Don’t have to login into the instance
- Deploy will be done automatically by the Beanstalk

We just have to monitor the environment and the statistics will be available in the Beanstalk dashboard (Also in the CloudWatch logs).

2.2.7.2 Automatically launched with a load balancer

With a load balancer you can access the applications using the **Load Balancer DNS.**

2.2.7.3 Other Features?

Include auto scaling. If we wanted to create our EC2 instance where the application will be deployed within the virtual private cloud or in a particular subnet within the VPC. All those features are available and we can select them using the Beanstalk itself. You don’t have to move out to

the VPC, don't have to go to the EC2 dashboard and select all those separately. Everything will be available in the EB dashboard.

2.2.7.4 Create custom URL

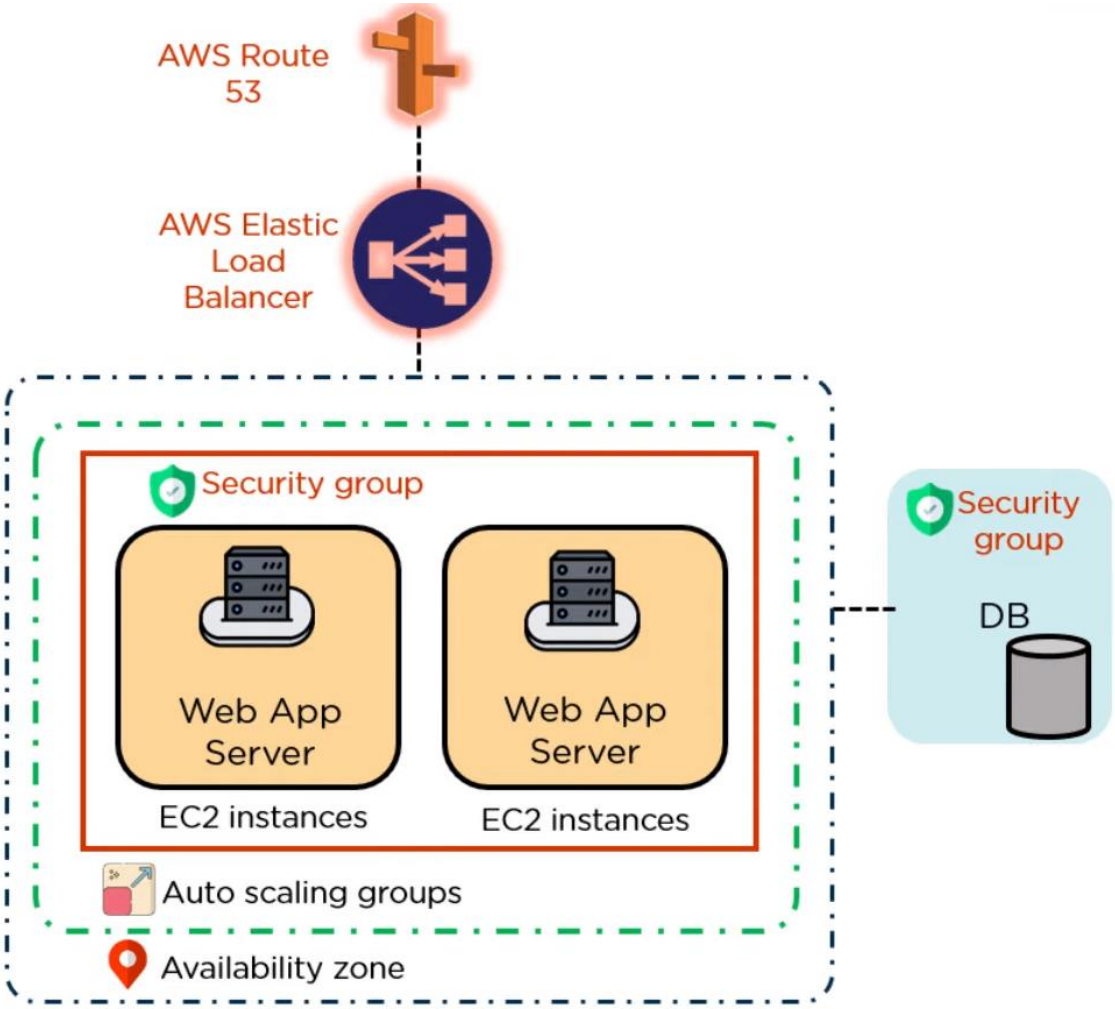
Check for the availability for that URL and then we can use that URL to access our application.

2.2.8 Architecture of AWS Elastic Beanstalk

2 different environment tiers:

1. Web Server Environment
   ➔ Front end facing
   - Client should be accessing this environment directly using a URL
   - Handles the HTTP(s) request from the client
2. Worker Environment
   ➔ Backend applications
   - The micro apps, which are basically required to support the running of the web applications
   - Processes background tasks and minimizes the consumption of resources

2.2.9 Design of EB



- If the application receives client requests, Amazon Route53 sends these requests to the AWS Elastic Loud Balancer
- Later, AWS Load Balancer shares the requests among EC2 instances
- Every EC2 instance have a security group
- The Load Balancer is connected to Amazon EC2 instances, which are part of an Auto Scaling group
- Auto Scaling Group automatically starts additional Amazon EC2 instances to accommodate increasing load on your application
- Auto Scaling Group automatically monitors and scales AWS EC2 instances based on the workload
- When to load on the application decreases, EC2 instances will also be decreased
- EB has a default security group. Here, the security group acts as a firewall for the instances
- With these groups, allows establishing security groups to the database server as well

2.3 AWS VPC

2.3.1 Why do we use a VPC?

An AWS VPC (Virtual Private Cloud) is a virtual network environment that allows users to launch Amazon Web Services (AWS) resources in a private and isolated section of the cloud. The biggest reason why that we used this is the security part. It is verry important that every part of the web shop has some good security. It is also verry important that there is some network control so that was an other reason to choose for a VPC. The more detailed description of what a VPC is and why to use it is tolled in the point underneath.

2.3.2 What is a VPC?

Thousands, even millions of customers using AWS every day. They're using ec2 instances, s3 buckets, databases, … We don't want to just throw our resources out there with all the other millions of resources.

We want our own private slice of the overall AWS pie. = VPC

Our own private cloud/network within the cloud.

→ isolates our resources from everyone else's.

When we create our AWS account and check out the VPC section, we see that we have one default VPC. But, we can create our own with specific IP's so that the we know which areas the user can access and which not.

2.3.3 Why do we need a VPC?

1. Security: Amazon VPC allows us to create a virtual network that is isolated from the rest of the internet. In our case, this can be useful for protecting sensitive data (credit card information) and applications from external threats.
2. Network Control: Amazon gives us complete control over our own virtual networking environment, including IP address range, subnets and route tables. This allows us to design and implement our network in a way that meets our specific business needs.
3. Performance: It enables us to launch our resources, such as Amazon EC2 instances, into a virtual network that is optimized for low latency and high performance.
4. Cost Savings: By using Amazon VPC, we can avoid the cost and complexity of using leased lines or VPN connections to connect to the AWS cloud. Amazon VPC also enables us to only pay for the resources we consume.

2.3.4 Key Features of VPC

1. Ip address range: We can select our own IP address range for our Amazon VPC, or let Amazon VPC select one for us.
2. Subnets: We can divide our Amazon VPC into one or more subnets, each representing a range of IP addresses in our Amazon VPC. We can select the IP address range for each subnet and associate security groups and network access control list with each subnet.
3. Security groups: We can use security groups to control inbound and outbound traffic to our Amazon EC2 instances. We can create security groups for our Amazon VPC, and specify the protocols, ports and source IP ranges that are allowed to reach our instances.
4. Network Access Control lists: We van use network access control lists to control inbound and outbound traffic to our subnets. We can specify the protocols, ports and source IP ranges that are allowed to reach our instances.
5. Elastic IP addresses: We can allocate static IPv4 address to our Amazon VPC and associate them with our instances, even if the instance is stopped and restarted.
6. VPN connections: We can use VPN connections to connect to our on-premises data centers to our Amazon VPC and our on-premises data center, and then use the VPN connection to access our VPC from our on-premise datacenter (not in our case).
7. AWS Direct Connect: We can use AWS Direct Connect to establish a dedicated network connection between our on-premises data center and our Amazon VPC. This allows us to bypass the public internet and use a private network connection to our Amazon VPC (not in our case).

2.3.5 Network Configuration

We can customize our network configuration for our Amazon VPC. For example, we can create a public-facing subnet for our web servers that has access to the internet. On the other hand, we can place our backend systems such as databases or applications servers in a private-facing subnet with not internet access. We can leverage multiple layers of security, including security groups and network access control lists to help control access to Amazon EC2 instances in each subnet. We can both use IPv4 & IPv6 in our VPC for secure and easy access to resources and applications.

2.4 AWS Load Balancing

2.4.1 Why do we use a load balancer?

We use the load balancer to increase the availability and fault tolerance of our application by automatically distributing incoming traffic across multiple targets, such as Amazon EC2 instances, containers, and IP

addresses, in one or more Availability Zones. You can find a more detailed description of what a load balancer is underneath.

2.4.2 What is Load Balancing?

Amazon Web Services (AWS) Elastic Load Balancing (ELB) is a load balancing service that automatically distributes incoming application traffic across multiple Amazon EC2 instances, containers, and IP addresses in one or more Availability Zones.

2.4.3 Types of Load Balancers

1. **Application Load Balancer:** This is a layer 7 load balancer that routes traffic to targets within Amazon Virtual Private Cloud (Amazon VPC) based on the content of the request. It is designed to handle incoming traffic from the internet and operates at the request level (layer 7).
2. **Network Load Balancer:** This is a layer 4 load balancer that routes traffic to targets within Amazon VPC based on IP protocol data. It is designed to handle millions of requests per second while maintaining ultra-low latencies.
3. **Classic Load Balancer:** This is a legacy load balancer that routes traffic at the connection level (layer 4). It supports either TCP or SSL protocols and is useful for applications that rely on the performance of a raw TCP connection or the security of an SSL/TLS connection.

2.4.4 Summary

AWS ELB helps you increase the availability and fault tolerance of your application by automatically distributing incoming traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, in one or more Availability Zones. It also enables you to scale your application up or down in response to incoming traffic and ensure that traffic is evenly distributed across healthy targets.

2.5 AWS Auto Scaling

2.5.1 Why do we use auto scaling?

We use AWS Auto Scaling group to automatically scale Amazon EC2 instances in response to changing demand, or to maintain a steady number of instances running even when demand is unpredictable. This means that a new EC2 will be spined up automatically when the current one is failing or when there is a lot of traffic going to this instance.

2.5.2 What is auto Scaling?

Amazon Web Services (AWS) Auto Scaling is a service that automatically adjusts the number of Amazon Elastic Compute Cloud (Amazon EC2)

instances in your application's environment based on conditions you define. It helps ensure that you have the appropriate number of instances available to handle the load on your application.

2.5.3 How it works

1. You define one or more scaling policies that specify the conditions under which you want to scale up or down, and the number of instances to add or remove when those conditions are met.
2. You create a group of Amazon EC2 instances, called an Auto Scaling group, that you want to manage.
3. You specify the minimum and maximum number of instances you want in the group.
4. You set up one or more Amazon CloudWatch alarms to monitor conditions that you care about, such as the average CPU utilization or the number of failed requests.
5. If an alarm triggers, Auto Scaling responds by adding or removing instances based on the scaling policies you defined. This helps ensure that you have the right number of instances to handle the load on your application.

2.5.4 Summary

Auto Scaling can be used to scale Amazon EC2 instances in response to changing demand, or to maintain a steady number of instances running even when demand is unpredictable. It can also be used to reduce costs by automatically scaling down instances when they are not needed, and scaling them back up when demand increases.

2.6 AWS EC2

2.6.1 Why do we use AWS EC2?

We use some standalone Amazon Elastic Compute Cloud instance to host Jenkins, SonarQube and a webhook. We will come back to those parts in some further points. We have chosen to use an EC2 to run those technologies because its verry easy to add those in our network and it is also possible to turn those off when the day is over to save some money. In the point underneath you can find some more detailed information about the Amazon EC2.

2.6.2 Why Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 500 instances and choices of the latest processor, storage, network, operating system and purchasing model to help you best fit your workload needs. Intel, AMD, and Arm processors are supported, there's on-demand EC2 Mac instances, and there's 400 Gbps of Ethernet networking. More SAP, high performance

computing (HPC), ML, and Windows workloads run on AWS than any other cloud.

2.6.3 Key Features of EC2

1. Run cloud-native and enterprise applications: Amazon EC2 provides secure, reliable, high-performance, and cost-effective compute infrastructure to meet specific business requirements.
2. Scale for HPC applications: Access the on-demand infrastructure and capacity you need to run HPC applications faster and cost-effectively.

2.7 AWS RDC

2.7.1 Why do we use AWS ECR?

We have used an Amazon ECR as an easy midpoint to deploy our site. We did this because it's very easy to send data to an ECR. This is also why we first send the date to the ECR and after that we will copy this data to right Beanstalk instance.

2.8 AWS RDS

2.8.1 Why do we use AWS RDS?

We also use an database in each stack (production and development). Our database is only connected with our backend. The database stands alone in a private subnet. We connected to this database with an SSL tunnel. The data on this database is some personal information about our clients so this really needed to be secure. We will save information like orders, addresses, users, products that are coupled to the order and some other personal information. You can find some more information about de RDS underneath.

2.8.2 What has RDS to offer?

- Deploy and scale the relational database engines
- Achieve high availability with Amazon RDS Multi-AZ deployments
- Benefit from extensive operational expertise, security best practice and innovation in databases in the cloud.

2.8.3 Key Features of RDS

- Build web and mobile applications: Support growing apps with high availability, throughput, and storage scalability. Take advantage of flexible pay-per-use pricing to fit different application usage patterns.

- Move everything to a managed database: Innovate and build new apps with Amazon RDS instead of worrying about managing your databases yourself, which can be time consuming, complex and expensive.
- Escape Standard Databases: Free yourself from expensive, penalizing, commercial databases by migrating to Amazon Aurora. When you migrate to Aurora, you get the scalability, performance, and availability of commercial databases at 1/10th the cost.

2.9 AWS Backup

2.9.1 Why do we use AWS Backup?

We use just the AWS backup implemented version of the Beanstalk. This will make a backup off our database (RDS).

2.9.2 What has AWS Backup to offer?

- Facilitates backup and recovery with a fully managed, policy-based service.
- Make as many backups as you want to protect yourself against accidents and malicious incidents.
- Monitor and confirm data protection thanks to auditor-ready reports.

2.9.3 Key Features of AWS Backup

- Cloud-native backups: Back up important data stores, such as buckets, volumes, databases, and file systems, through AWS services.
- Hybrid data protection: Centralize data protection management for your applications running in hybrid environments, such as VMware workloads and AWS Storage Gateway volumes.
- Centralized data protection policies: Configure, manage, and control your backup activity for your company's AWS accounts, resources, and AWS regions.
- Data protection compliance: Examine your resources against data protection policies to ensure compliance with organizational or legal requirements.

2.10 Jenkins

2.10.1 Why do we use Jenkins?

Maybe one of the most important technologies that’s left is Jenkins. We used this technology to set up everything automatically. We wrote a pipeline that will get our data and send it to the right destination. This will be discussed in a further point. Like we already told is Jenkins installed

on an EC2 instance. You can find some information about what Jenkins is in the points underneath.

2.10.1 What is Jenkins?

Jenkins is an open-source automation server that is used to automate different stages of the software development process, such as building, testing, and deploying code changes. It is written in Java and provides hundreds of plugins that enable users to customize and extend its functionality to fit their specific needs.

Jenkins allows developers to automate repetitive tasks, such as compiling code, running tests, and deploying applications, which can save significant time and effort. It can also be integrated with other software tools, such as Git, GitHub, and Docker, to provide a complete DevOps solution.

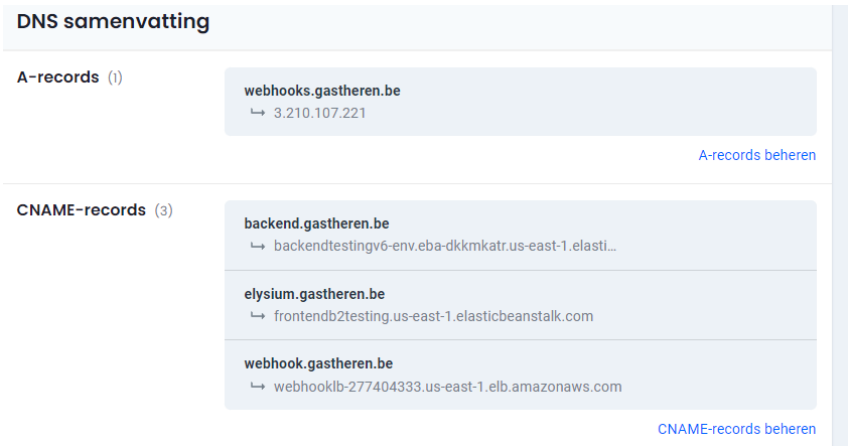
Jenkins is designed to be highly flexible and scalable, making it suitable for projects of any size or complexity. Its user interface is web-based and provides an intuitive dashboard that allows users to manage their jobs and view build logs and reports.

Overall, Jenkins is a powerful automation tool that can streamline the software development process, improve efficiency, and reduce errors. Its extensive plugin ecosystem and flexibility make it a popular choice for developers and teams seeking to automate their build, test, and deployment processes.

2.11 Combell

2.11.1 Why do we use Combell?

Because we are using the ‘High Availability’ feature on our EC2s of the elastic beanstalk, our beanstalk will launch an elastic load balancer, with this load balancer we can configure that our server can use HTTPS. Now in order to use HTTPS we need to obtain a certificate. Because we already had a domain on Combell we used that and got our certificate from there and put it on our beanstalk using the service: Certificate Manager.



We put the generated URL we get from the beanstalk server and put that URL in Combell. Now our site is available at https://elysium.gastheren.be

2.11.2 What has Combell to offer?

Combell is a web hosting and cloud services provider that offers a range of solutions for businesses and individuals seeking reliable and scalable hosting services. Here are some of the key services and features that Combell offers:

- Web hosting: Combell provides a range of web hosting solutions, including shared hosting, WordPress hosting, and VPS hosting, to meet the needs of businesses of all sizes.
- Domain registration: Combell offers domain registration services, allowing users to register and manage their domain names with ease.
- Cloud hosting: Combell provides a range of cloud hosting solutions, including cloud servers, cloud storage, and cloud backup, to provide scalable and flexible cloud hosting solutions.
- Email hosting: Combell offers email hosting solutions, including Microsoft Exchange, to provide reliable and secure email services for businesses.

- SSL certificates: Combell offers SSL certificates, allowing users to secure their websites and protect their customers' data.
- 24/7 support: Combell provides 24/7 customer support via phone, email, and live chat, ensuring that users can get help when they need it.

Overall, Combell offers a range of reliable and scalable hosting solutions, backed by excellent customer support. Its services are designed to meet the needs of businesses of all sizes, from small startups to large enterprises, and its flexible pricing options make it an affordable choice for businesses seeking quality hosting services.

2.12 SonarQube

2.12.1 Why do we use SonarQube?

We use a simple EC2 server to run SonarQube. We use SonarQube for some different purposes. A first reason why we use SonarQube is that will check if there is an 80% code coverage. This is something where we didn't succeed in so we made the chose to not use this in the further pipeline. Another reason is to check on the quality of the code and to see if there are any errors. In the part below you can read some other features of SonarQube and what that they do.

2.12.1 What has SonarQube to offer?

SonarQube is a popular open-source platform for continuous code quality inspection, which is used to analyse and measure code quality throughout the software development process. Here are some of the key features that SonarQube has to offer:

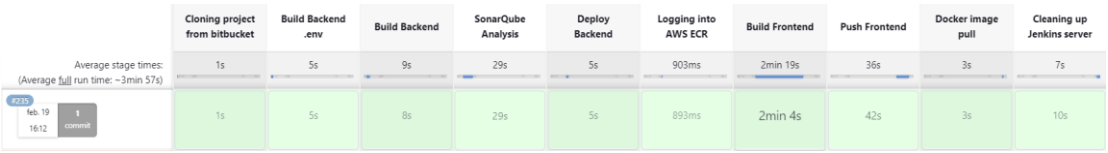
- Code analysis: SonarQube analyses code quality in a variety of programming languages, including Java, C++, Python, and many others. It provides detailed reports on issues such as code smells, bugs, and vulnerabilities.
- Continuous inspection: SonarQube integrates with popular build tools such as Maven, Gradle, and Jenkins, allowing developers to run code analysis as part of their continuous integration and delivery pipelines.
- Quality gates: SonarQube allows users to define quality gates, which are a set of criteria that must be met before code can be promoted to production. This ensures that code meets predefined quality standards and reduces the risk of introducing defects into production environments.
- Dashboards and reports: SonarQube provides interactive dashboards and reports that allow users to visualize and track code quality trends over time, making it easier to identify areas for improvement.

- Code coverage: SonarQube provides code coverage metrics, allowing users to measure how much of their code is covered by unit tests.

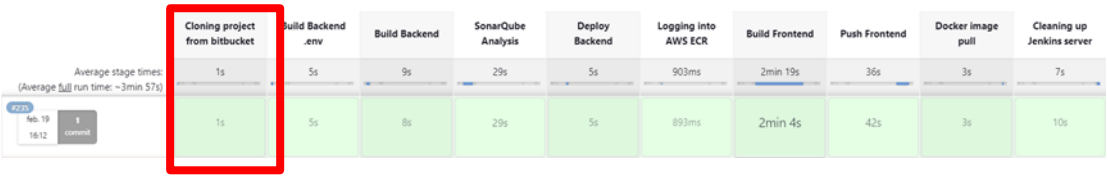
Overall, SonarQube is a powerful platform for continuous code quality inspection, providing detailed insights into code quality issues and helping teams to improve their software development practices. Its integration with popular build tools and the ability to define quality gates make it a valuable tool for teams seeking to ensure that their code meets predefined quality standards.

3 How to set up the pipeline

To deploy our front and backend we use a Jenkins pipeline. This will make it possible to deploy it automatically. In the following parts we will go deeper into every part in the following steps. On the picture below you can see how our pipeline looks.



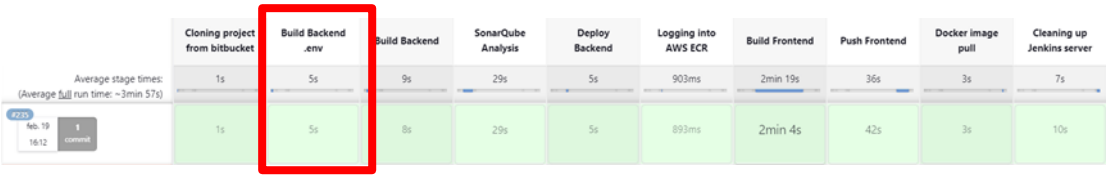
3.1 Step 1: Cloning the project from bitbucket



In our first step of the pipeline we will clone the project from bitbucket. The pipeline will start every time that we make a change to the code in the development branch on bitbucket. In the pipeline for our production environment it will use the master branch to clone the project. If Jenkins notice a change it will clone the project from the right branch and it will move on to the next step. On the picture below you can see the code that we use in this step.

```
stage('Cloning project from bitbucket') {
    git credentialsId: '238d9fd0-3163-4d0e-bd95-515338573471',
    url: 'https://bitbucket.org/xploregroup/thomas-more-2-2023.git',
    branch: 'deployment'
}
```

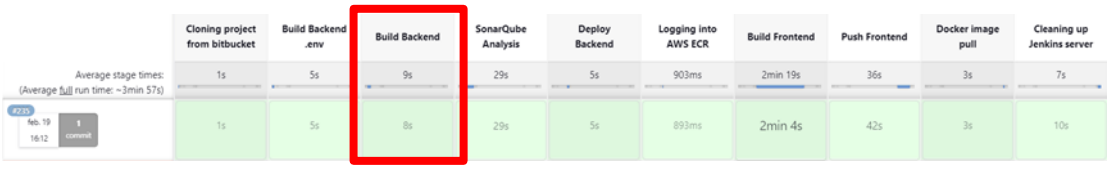
3.2 Step 2: Building the backend .env file



In the next step we are going to rewrite some variables. This is for security reasons so that those vulnerable variables don't need to stand in the code and that these can be saved and add in a secure way. You can see on the picture below you can see the code that we use in this step.

```
stage('Build Backend .env') {
    script {
        dir('./Elision-BackEnd/src/main/resources/'){
            sh 'pwd'
            sh 'cp .env.example .env'
            sh 'sed -i "s+example15+$MAIL_PASS_JENKINS+g" .env'
            sh 'sed -i "s+example14+$MAIL_USER_JENKINS+g" .env'
            sh 'sed -i "s+example13+$CMA_TOKEN_JENKINS+g" .env'
            sh 'sed -i "s+example12+$ALGOLIA_INDEX_JENKINS+g" .env'
            sh 'sed -i "s+example11+$ALGOLIA_APP_KEY_JENKINS+g" .env'
            sh 'sed -i "s+example10+$ALGOLIA_APP_ID_JENKINS+g" .env'
            sh 'sed -i "s+example1+$CONTENTFUL_SPACE_JENKINS+g" .env'
            sh 'sed -i "s+example2+$CONTENTFUL_TOKEN_JENKINS+g" .env'
            sh 'sed -i "s+example3+$DB_URL_JENKINS+g" .env'
            sh 'sed -i "s+example7+$JWT_COOKIE_JENKINS+g" .env'
            sh 'sed -i "s+example8+$JWT_EXPIRY_JENKINS+g" .env'
            sh 'sed -i "s+example9+$JWT_SECRET_JENKINS+g" .env'
            sh 'sed -i "s+example4+$DB_EC2_URL_JENKINS+g" .env'
            sh 'sed -i "s+example6+$DB_USERNAME_JENKINS+g" .env'
            sh 'sed -i "s+example5+$DB_PASS_JENKINS+g" .env'
            sh 'cat .env'
        }
    }
}
```

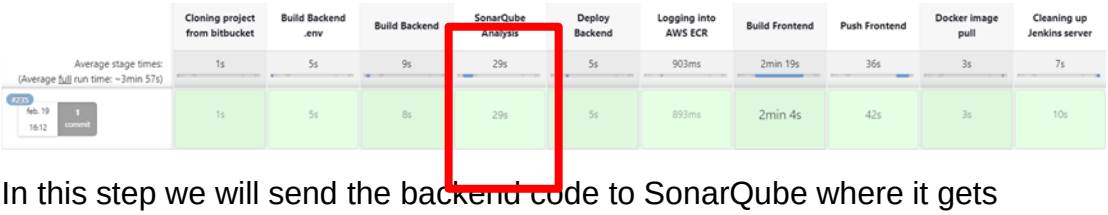
3.3 Step 3: Building the backend



The third step we are going to do is the build of the actual backend. We use this build to send the code to SonarQube and to our ECR. This will be explained in the next steps. In the code below you can see the maven command that we used to build the backend.

```
stage('Build Backend'){
    script{
        dir('./Elision-BackEnd/'){
            sh 'mvn clean package -D maven.test.skip'
        }
    }
}
```

3.4 Step 4: Analysis by SonarQube



In this step we will send the backend code to SonarQube where it gets an analyse. In the production it lets the pipeline fail if there isn't an 80% coverage but because we don't have that percentage we cant let it stop. In the picture of the code bellow you can see how we did this.

```
stage('SonarQube Analysis'){
    script {
        dir('./Elision-BackEnd/'){
            withSonarQubeEnv('SonarQube') {
                sh 'mvn clean verify sonar:sonar -D sonar.projectKey=Backend-Testing -D sonar.host.url=http://23.21.45.139:9000 -Dsonar.login=sqp_d483d346371e1fa1b9c529c189355c990c60465c'
            }
        }
    }
}
```

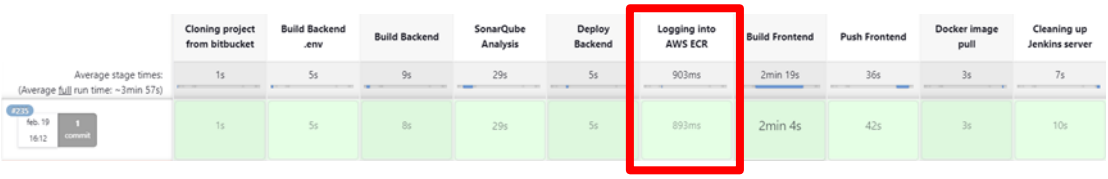
3.5 Step 5: Deploy Backend

The fifth step is the last one for our backend. This step will deploy our backend to the correct beanstalk environment. We will first pull the code from the ECR we pushed it earlier to. After that we will create and update this version. On the images below you can see the code we used for this. We used two images because this code is to long to show in one time.

```
stage("Deploy Backend"){
    script{
        sh 'aws s3 cp ./Elision-BackEnd/target/Elision-BackEnd-0.0.1-SNAPSHOT.jar s3://elasticbeanstalk-us-east-1-550988791405/Elision-BackEnd-0.0.1-S
        sh 'aws elasticbeanstalk create-application-version --application-name "BackendTestingV6" --version-label "Application Backend $BUILD_NUMBER"
        sh 'aws elasticbeanstalk update-environment --application-name "BackendTestingV6" --version-label "Application Backend $BUILD_NUMBER" --enviro
    }
}
```

```
1-BackEnd-0.0.1-SNAPSHOT.jar'
d $BUILD_NUMBER" --source-bundle S3Bucket=elasticbeanstalk-us-east-1-550988791405,S3Key=Elision-BackEnd-0.0.1-SNAPSHOT.jar'
_NUMBER" --environment-name "Backendtestingv6-env"'
```

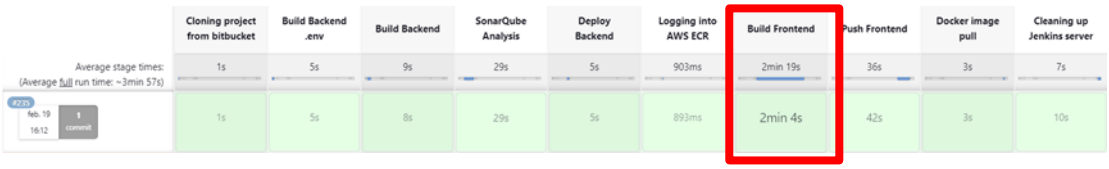
3.6 Step 6: Logging into AWS ECR



In the next step we will log into our AWS ECR for the frontend. You can see how this is done with the code from the next picture.

```
stage('Logging into AWS ECR') {
    script {
        sh "aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/u8e2q3m6"
    }
}
```

3.7 Step 7: Building the frontend

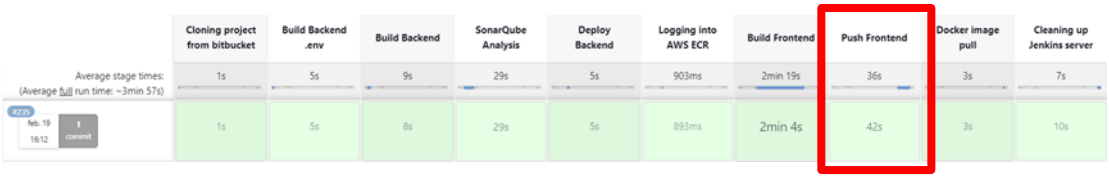


The seventh step is the building of the frontend. We will also add some secret variables to our code. When that's done we will use Docker to build an image of the frontend. We will use this image in the next step of our pipeline. You can see the code we use in this step on the image below.

```
stage('Build Frontend') {
    script {
        dir('./elision-frontend/'){
            sh 'pwd'
            sh 'cp .env.example .env'
            sh 'sed -i "s+example15+$MAIL_PASS_JENKINS+g" .env'
            sh 'sed -i "s+example14+$MAIL_USER_JENKINS+g" .env'
            sh 'sed -i "s+example13+$CMA_TOKEN_JENKINS+g" .env'
            sh 'sed -i "s+example12+$ALGOLIA_INDEX_JENKINS+g" .env'
            sh 'sed -i "s+example11+$ALGOLIA_APP_KEY_JENKINS+g" .env'
            sh 'sed -i "s+example10+$ALGOLIA_APP_ID_JENKINS+g" .env'
            sh 'sed -i "s+example1+$CONTENTFUL_SPACE_JENKINS+g" .env'
            sh 'sed -i "s+example2+$CONTENTFUL_TOKEN_JENKINS+g" .env'
            sh 'sed -i "s+example3+$DB_URL_JENKINS+g" .env'
            sh 'sed -i "s+example7+$JWT_COOKIE_JENKINS+g" .env'
            sh 'sed -i "s+example8+$JWT_EXPIRY_JENKINS+g" .env'
            sh 'sed -i "s+example9+$JWT_SECRET_JENKINS+g" .env'
            sh 'sed -i "s+example4+$DB_EC2_URL_JENKINS+g" .env'
            sh 'sed -i "s+example6+$DB_USERNAME_JENKINS+g" .env'
            sh 'sed -i "s+example5+$DB_PASS_JENKINS+g" .env'
            sh 'cat .env'

        }
        dockerImage = docker.build("public.ecr.aws/u8e2q3m6/jenkins-pipeline-public","-f ./elision-frontend/Dockerfile .")
    }
}
```

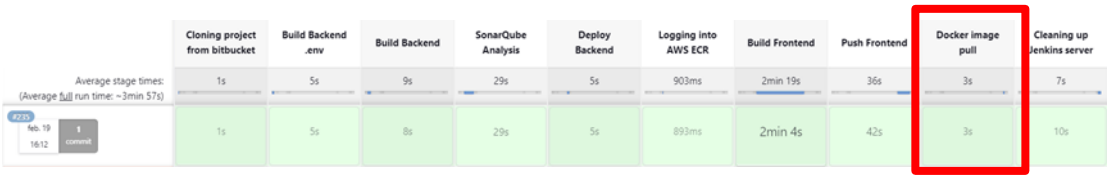
3.8 Step 8: Pushing the fronted to the ECR



In step eight we will push our code to the ECR. This will make it possible to put this code on the right beanstalk. You can see the code that we use in this step on the picture below.

```
stage('Push Frontend') {
  script {
      sh "docker push public.ecr.aws/u8e2q3m6/jenkins-pipeline-public:latest"
    }
}
```

3.9 Step 9: Docker image pull from ECR



In our penultimate step is also the last step for our frontend. This step will deploy our frontend to the correct beanstalk environment. We will first pull the code from the ECR we pushed it earlier to. After that we will create and update this version. On the images below you can see the code we used for this. We used two images because this code is to long to show in one time.

```
stage('Docker image pull') {
  script {
    sh 'aws s3 cp ./elision-frontend/Dockerrun.aws.json s3://elasticbeanstalk-us-east-1-550988791405/Dockerrun.aws.json'
    sh 'aws elasticbeanstalk create-application-version --application-name "testing_FrontendB2" --version-label "Application $BUILD_NUMBER"
    sh 'aws elasticbeanstalk update-environment --application-name "testing_FrontendB2" --version-label "Application $BUILD_NUMBER" --enviro
    }
}
```

```
ion'
.ication $BUILD_NUMBER" --source-bundle S3Bucket=elasticbeanstalk-us-east-1-550988791405,S3Key=Dockerrun.aws.json'
$BUILD_NUMBER" --environment-name "Testingfrontendb2-env"'
```

3.10 Step 10: Cleaning up the Jenkins server

In our last stap of the pipeline we will clean up our Jenkins server. This will prevent that its memory will run out at some time what will result in a fail in the pipeline. You can see the code we used in this last stap on the picture below.

```
stage('Cleaning up Jenkins server') {
    script {
        sh 'cd'
        sh 'docker system prune -af'
    }
}
```

4 Technologies Application

4.1 Spring Boot

Spring Boot is a Java-based framework that provides a convenient way to create stand-alone, production-grade applications using the Spring framework. It simplifies the process of building and deploying Java applications by providing a set of pre-configured options and dependencies, allowing developers to focus on writing code rather than setting up infrastructure.

Spring Boot offers several benefits for Java developers:

- Simplified configuration: Spring Boot uses a convention-over-configuration approach, meaning that it comes with a set of default configurations that can be easily overridden by the developer. This

simplifies the process of setting up a new project and reduces the amount of boilerplate code required.
- Auto-configuration: Spring Boot automatically configures dependencies based on the contents of the class path, eliminating the need to manually specify dependencies in configuration files. This saves time and reduces the risk of errors.
- Embeddable web servers: Spring Boot includes support for multiple embedded web servers, including Tomcat, Jetty, and Undertow. This allows developers to easily run and test their applications locally, without the need to install a separate application server.
- Actuator: Spring Boot includes an Actuator feature that provides monitoring and management capabilities for applications. This includes endpoints for monitoring application health, metrics, and other operational information.

To include the appropriate dependencies for the project, developers can work with Maven or Gradle, as well as the variety of starter dependencies which Spring Boot provides. Once the dependencies are in place, developers can use the Spring Boot framework to create and configure their applications. This can be done using Java-based configuration or using annotations in the code.

Moreover, Spring Boot's seamless integration with databases and other backend systems allows data to be efficiently passed through the backend to the frontend, where it can be leveraged for various purposes. For example, the information gathered can be used to process payments using Adyen, a popular payment service provider, or to perform search operations using Algolia, a powerful search engine.

4.2 Next.JS and React

Next.js is a framework for building server-rendered or statically-exported React applications. It is built on top of React and provides a set of tools and features to make it easier to create and deploy web applications using React.

Next.js offers several benefits for React developers:

- Server rendering: Next.js allows developers to build server-rendered React applications, which can improve performance and SEO by pre-rendering content on the server.
- Static site generation: Next.js also allows developers to build statically-exported React applications, which can be deployed to a variety of hosting platforms and are easy to scale.
- Automatic code splitting: Next.js automatically splits code into chunks, allowing for faster loading times and better performance.
- Routing: Next.js includes built-in support for client-side routing, making it easy to create single-page applications with React.

To use Next.js, developers need to install the framework and include it in their project's dependencies. This can be done using npm or yarn. Next.js provides a variety of starter templates and examples to help developers get started quickly.

As soon as the dependencies are configured, developers can use Next.js to create their React applications. This can be done using JavaScript and React components, and Next.js provides a range of tools and features to help developers build and deploy their applications.

4.3 Adyen

Adyen is a payment platform that allows businesses to accept payments from customer. You can use Adyen for online, in-app and in-store payments. It has great fraud detection, account management and reporting tools.

The key benefits are:

- It supports a wide range of payment methods, it supports major credit and debit card, popular digital wallets and local payment options. There are over 250 payments methods that are supported.
- Adyen also supports over 150 currencies and the ability to do transactions in 187 countries. This makes it a good choice to use if u have customers all around the globe.
- The integrated fraud detection uses machine learning algorithms. This helps business to protect them and their customers from fraud.
- Scalability and reliability are key features of Adyen. The platform is able to process millions of transactions per hour. This makes it suitable for businesses of all sizes.

4.4 Contentful

Contentful is a content management system (CMS) that provides a lot of tools for managing and delivering content for application. It allows u to create, edit and manage content in a structure and consistent manner.

Contentful provides a flexible model that allows businesses to define their own custom content. Its easy to reuse and rearrange content in different contexts, this makes it easier to create and maintain a consistent content setup.

It is also built to handle large volumes of content and traffic. It offers multiple tools and features to help businesses to manage their content effectively, including search, localization of customers and versioning.

It also provides a variety of API and integrations that make it easier to integrate other system and tools. This allows us to build custom solution that fit their needs.

Fast and efficient are key components, it has multiple features like caching and delivery through CDN. This helps to ensure content is delivered quickly and reliably to users regardless of their device or location.

4.4.1 Add data to contentful

To add a new product or category to the website, follow these steps:

- Login into Contentful using your credentials.
- Navigate to the "Content" tab in the top navigation menu.
- Click the blue "Add Entry" button to start creating a new entry.
- Select the type of content you want to add, such as "Product" or "Category", … , from the dropdown menu. This will open a new entry form with fields specific to the selected content type.
- Fill in all the details for the new entry. This will include fields such as name, description, price, images, and other relevant information for the chosen content type.

Once you have filled in all the required details, click on the "Publish" button to save the new entry.

After you click "Publish", the new entry will be added to the website within approximately 4 seconds. The website will automatically fetch the new data from Contentful and display it in the relevant sections of the website, such as the product or category pages.

It is important to note that any changes made to the content in Contentful will be reflected on the website automatically. This means that if you need to update any product or category details, simply navigate to the corresponding entry in Contentful, make the necessary changes, and then publish the updated entry. The website will then be updated with the new information in real-time.

4.4.2 Add a new page in Contentful

The content types in Contentful define the structure and types of content that can be added to the website. To create a new content type, follow these steps:

- Login into Contentful using your credentials.
- Navigate to the "Content" tab in the top navigation menu. This will show you a list of existing entries and allow you to create a new one.
- Click the blue button "Add Content Type". This will open a new screen where you can define the structure of your new content type.
- Give the content type a name in the "Name" field. The API identifier is automatically filled in for you.
- Click the "Create" button to create the new content type.

- Click on the blue "Add Field" button to add fields to your content type. You can choose from various field types, such as text, number, location, and more.
- Choose the field type you want to add and configure it as needed. This includes setting a name for the field and defining any validation rules.
- Fill in the name for the field in the "Name" field. The field ID is automatically filled in for you.

Don't forget to publish your new content type to make it available for use.

You can add data to your page as mentioned above in *4.4.1 Add data to Contentful*

Once your content type is saved, you can copy the content type ID and use it in the frontend or backend to make API calls and retrieve all the content of this specific type.

```
const info =await ContentfulClient("607ViWugxz70AGhnkOUncQ")
```

4.5 Mock Data

4.5.1 Getting the Data

To simulate a plausible e-commerce site, we required a large amount of real product data. We accomplished this by scraping data from the website coolblue.be, which provided ample and relevant information for our site. The first step in this process was to scrape the links of all the products that were to be included in the project. We achieved this using the script scrapelinks.py, which identified predetermined types of products available on Coolblue, such as "https://www.coolblue.be/en/headphones" and "https://www.coolblue.be/en/soundbars". The script then visited each of these categories and collected the links of the listed products. The number of pages to scrape was determined by the desired number of products, with each page containing approximately 22 products. If, for example, 100 products per product type were required, the max_page was set to 5. The script generated a .txt file per product type, containing all of the links, which were then saved in a folder called links. We implemented multithreading to expedite the process, which can be disabled if the computer being used lacks sufficient power.

With the links gathered, the next step was to extract the data from the collected links. We accomplished this using the script scraper_data.py, which processed the .txt files with the links and extracted information such as product name, price, description, specifications, reviews, and images. The data was then saved to a JSON file, ready for upload to both Contentful and Algolia. Each product type had its own JSON file, which was saved in a folder called data. To speed up the process of uploading

the data, we also used multithreading, which could be disabled if necessary.

4.5.2 Uploading Data to Contentful

In order to transfer the product data from our scraped json files to the primary database for products, which is Contentful, we have developed a script called upload_contentful.py. This script iterates through each of the json files containing the products and uploads them to Contentful with their API.

To ensure the integrity and consistency of the data, we have implemented several checks before uploading it. Since the Web Layouts of the websites we scrape may not always be consistent, we validate the data of each product to ensure that every attribute is present. Additionally, we perform some clean-up of the data to ensure that the final presentation is as professional and polished as possible.

4.5.3 Uploading Data to Algolia

In order to enable search functionality on our website, we use Algolia as our search engine. Algolia requires that the items that we want to be searchable are stored in its own database. This means that the product data that was previously sent to Contentful must also be uploaded to Algolia.

To perform this task, we have developed a script called publish_to_algolia.py. This script retrieves all the products that are currently available on Contentful and uploads them to Algolia using its API. Since the data stored in Contentful is already clean and consistent, no additional checks or edits are necessary before uploading it to Algolia.

Once the website is in production, a webhook will be responsible for updating the Algolia database automatically whenever new products are added, or existing products are modified in Contentful. However, the initial upload of product data to Algolia must be performed manually using publish_to_algolia.py.

Overall, this process allowed us to gather real and comprehensive product data that was used to build and populate the e-commerce site. By scraping data from coolblue.be, we were able to obtain a vast amount of relevant information that was used to create a realistic and informative shopping experience for our users. The use of scripts such as scrapelinks.py and scraper_data.py allowed us to efficiently process and extract the necessary data, while multithreading helped to expedite the process. The resulting JSON files were easily uploaded to both Contentful and Algolia, allowing for seamless integration of the product data into our e-commerce platform.

4.6 Algolia

This section provides an overview of how Algolia is integrated in this project, including the available features and implementation details.

4.6.1 Search Configuration and Optimization

Algolia is a powerful search tool, but it needs to be properly configured to ensure the best user experience. This is achieved through relevance tuning. For a web shop, several configurations are performed. First, some of the product properties are added to the searchable attributes, such as brand, name, category, and description. The order of these attributes is determined by their relevance. However, this may vary depending on the type of shop or data being used. Next, a ranking and sorting method is configured. In our case, we have chosen to rank products by review score, followed by price as a tie-breaker. Other settings are also tuned, such as typo tolerance, language, and stop words. Lastly, facet items are configured, which are the filtering options available to users when using the search function on the website.

Overall, the goal of the search configuration and optimization process is to provide users with the most relevant and accurate search results possible, while also allowing them to easily filter and sort through the results to find the products they are looking for. By carefully configuring the search settings in Algolia, we can help ensure that our users have a seamless and efficient search experience..

4.6.2 Events

Algolia provides powerful tools such as recommendation systems and personalization to improve the performance of the webshop. However, to accomplish this, Algolia needs to gather the right data about what users are doing on the site. This is achieved by capturing and sending back certain events that users perform. These events help the system determine things such as items that are frequently bought together, items that are trending, and more. In this project, the following events are sent back to Algolia if the user consents:

- Product views and clicks

- Product searches

- Selected products after a search

- Products added to cart

- Multiple products bought together

These events are then stored and used to train a model. If a user is logged in, their userID is also sent back with the event, which enables personalization, as explained in the next section.

By collecting and analyzing these events, Algolia can provide valuable insights into user behaviour and preferences, which can be used to improve the search and recommendation systems, and ultimately enhance the overall shopping experience for our users.

4.6.3 Personalization

With the data gathered from events, the Algolia system can personalize search results for individual users. This allows for a better user experience as search results are tailored to what is most relevant to a particular user. However, the current Algolia account is limited to the free trial, which means that the full potential of personalization cannot be utilized. Nevertheless, all the necessary code for the implementation of personalization has been included in the front end. If enabled and with sufficient data, Algolia should start personalizing search results for users.

4.6.4 Machine Learning Models for Recommendations

With Algolia's ability to gather valuable data about user behaviour on the site, it is possible to leverage this information to enhance customer experience and boost sales. By implementing features such as trending items, trending categories, related products, and frequently bought together items, users can have a better shopping experience and be more likely to make a purchase.

On the home page, trending items and categories can be displayed to catch the user's attention and provide a glimpse into popular products. On the product detail page, related and frequently bought together items can be suggested to increase the likelihood of a sale and improve the overall user experience.

Machine learning models can be trained using the data gathered from user events to make these recommendations even more accurate and relevant. As more users interact with the site and more data is collected, the quality of these recommendations will improve, leading to even higher customer satisfaction and sales.

To ensure that the models remain up-to-date with the latest data, they are retrained every day to reflect the changing preferences and behaviours of users on the site. By leveraging Algolia's powerful capabilities, the site can provide a highly personalized and engaging shopping experience for all users.

4.6.5 Business Analytics

In addition to the insights mentioned above, Algolia's business analytics also provides information about top search queries, popular filters, and top-performing products. This information can be used to optimize the search and filter functionalities of the site, ensuring that users can quickly and easily find the products they are looking for. Furthermore, the data

can also be used to identify any potential issues with the site, such as high bounce rates or low conversion rates, and take corrective actions to improve the overall user experience. Overall, Algolia's business analytics provides a comprehensive overview of the site's performance and user behaviour, enabling the shop to make data-driven decisions and stay ahead of the competition.

4.6.6 WebHook

Because the web shop relies on both data from Contentful and Algolia, they need to be synced up at all times. To achieve this, we created a Python Flask application that receives the webhook events from Contentful. The Flask application has a route that can handle POST requests from Contentful containing the ID of the product that changes. Once a change is detected by Contentful of a product, the webhook triggers the flask application. The flask application then checks for the changes and updates Algolia using its API.

We also used a combination of Gunicorn and Nginx to serve the Flask application. Gunicorn is a Python web server that can handle multiple simultaneous requests, while Nginx can be used as a reverse proxy to route incoming requests to the appropriate application. Finally, we used Wheel to package our Flask application and any necessary dependencies into a distributable format, making it easier to deploy and maintain the application on our online server.

5 Usage of the app

The e-commerce application is a web-based platform that allows users to browse and purchase a variety of products online. The application is designed to be user-friendly and easy to navigate, with a range of features that make it easy to find and purchase products.

5.1 Landing Page

When you first visit the site, you will be greeted with a landing page that displays three trending categories with a range of top-selling products. You can click on any of these items to view the product detail pages, or you can click on a category to view all of the products in that category.

5.2 Navigation

At the top of the page, you will find a navigation bar that allows you to access various parts of the application. Here, you can find your shopping cart, a login button, the about page, and all the categories available for browsing. You can also use the Algolia instant search bar to search for specific products based on keywords.

5.3 Browsing Products

The e-commerce application offers a wide range of products from various categories. To find products, you can either browse through the categories or use the search bar to look for products based on specific keywords. Once you find a product that you are interested in, you can click on it to view more details about the product.

5.4 Product Detail Pages

The product detail pages provide all the information you need to make an informed purchasing decision. Here, you can find product descriptions, images, prices, and other relevant information. You can also add the product to your cart or purchase it directly from the page.

5.5 Shopping Cart

The shopping cart is where you can view all the products you have added for purchase. You can update the quantity of products, remove products, and view the total price of all the items in the cart. Once you are ready to purchase your items, you can proceed to the checkout page.

5.6 Checkout

The checkout page is where you can enter your payment and shipping information, review your order, and complete your purchase. The application offers a secure payment gateway that ensures all transactions are safe and secure.

5.7 Leaving a Review

To leave a review, users can simply scroll down to the review section on the product detail page and click on the "Leave a Review" button. This will open a form where users can rate the product and provide written feedback.

5.8 Rating System

The review system uses a five-star rating system, where users can rate the product from one to five stars. Users can also provide written feedback, such as what they liked or disliked about the product, how it performed, and any other relevant information.

5.9 Viewing Reviews

Users can view all reviews for a product by scrolling down to the review section on the product detail page. Reviews are displayed in chronological order, with the most recent reviews appearing first. Users can also sort reviews by rating or by date.

5.10 Conclusion

Overall, the e-commerce application is designed to provide a user-friendly and intuitive shopping experience. Whether you are browsing through categories, searching for specific products, or purchasing items, the application offers a range of features that make it easy to find and purchase products online. For more information see the promotion video in *6 Promotional Video*

6 Promotional Video

In the video underneath you can see the basic working of our web shop. We will take you on walk through every part of the shop and let you see how those parts work.

URL: https://youtu.be/ozv3I7oN9qw

7 Credentials

In the following table you can find the credentials, as asked, of the platforms that we used.

Service	Username	Password	Account
Database	Admin	TeamB2abc123!	
Jenkins	mats_jentel	TeamB2!M&J	
Algolia	r0903904@student.thomasmore.be	ElisionTeamB2!	
Adyen	TM-Team-B2	ElisionTeamB2!	Elision

8 Conclusion

In summary, our team has successfully achieved the objectives of the project by developing a composable e-commerce platform. We utilized Next.js and a Java-based backend, with REST APIs to connect the database, backend, and frontend. After thorough research, evaluation, and consultation with coaches, we integrated external services like Algolia, Contentful, and recommendation engines, which significantly enhanced the platform's functionality. Despite initial challenges, we were able to complete all must-haves and should-haves within the project

timeline, displaying great teamwork and communication skills throughout, despite one team member's departure.

Nevertheless, due to time constraints, we could not allocate more time to polish the platform. We acknowledge that the platform could have benefited from SonarQube/UnitTests, which could have improved the overall quality and functionality of the platform. However, given the time we had, we believe that we did our best.

Overall, our team's ability to choose and incorporate external services, create a scalable and well-connected infrastructure, and manage communication effectively resulted in the final product. We are proud to have developed a modern e-commerce platform that can be of immense value to businesses.