



# DESIGN DOCUMENT MULTI

*ICT & INFRASTRUCTURE SEMESTER 3*

Alec Laclé (426195)  
Angel Molina (4230396)  
Javier Duran (3567885)

7 – Jan – 2022

## Revision Table

Description	Date	Authors	Approved by
Created design document	7 - 01 - 2022	Alec Lacle, Angel Molina, Javier Duran	Mikaeil Shaghelani
Improve design document	13 - 01 - 2022	Alec Lacle, Angel Molina, Javier Duran	Mikaeil Shaghelani

## Contents

Revision Table.....	2
Agreements – made with the tutor .....	4
1. Requirements .....	5
2. System Setup .....	6
3. Network configuration.....	7
3.1 Network diagram and description .....	7
3.2 Description of the services .....	8
4. Conclusion .....	9

## Agreements – made with the tutor

At the beginning of the semester we made agreements with the tutor about how we are going to meet up. This includes at what time, which day, and exactly what we will be discussing in the meeting. All of this was discussed at the first meeting on week 2. There will be 2 types of meetings depending on the week. The first type is *Tutor Meetings*. These are normal meetings where we can discuss our work with our tutor AS a tutor. Agreements that we settled on are:

- Day and time:
  - o We will meet up on Wednesday on normal circumstances (sometimes the meeting has to be rescheduled).
  - o The time of the meeting will be between 13:00 – 13:30
- A chairman and a secretary will be assigned before every meeting:
  - o A chairman should be the leading member of the meeting, guiding the other people involved through the meeting at all times.
  - o The secretary has to note every important detail and feedback that happens during the meeting within an agenda.

Furthermore there will be *Client Meetings* as well. These are meetings where our tutor will act as a client of an independent business that needs our help to deploy their application. We have to extract as much information as possible within these meetings, since we do not get these types of meeting that often. They function just like *Tutor Meetings* but we do not get feedback from our tutor, since he technically is a client a that giving point.

In addition to the meetings we also have a group Gitlab repository with all the work that we have created uploaded so the teachers can follow our work and evaluate us. This includes all our code files, documentation files and all our screenshots.

By the end of the project we need to make sure that we have these tasks completed:

- o We are expected to run a fully functional infrastructure that connect from the frontend to the backend.
- o We also need to create a database that can gather data from the client's application.

## 1. Requirements

To begin working on the project we have to be aware of the requirements first. There is a list of tasks given to us at the start. They are linked to documentation and the technical requirements:

- For the documentation there are: Project Plan, User Requirements Specification, Design Document (this document), Process Report, and User and Technical Manual.
- For the technical side we have to help programming students deploy their app online, this is done by making a web server to support different types of programming languages, while also making sure data is being stored in a database. We also have to create a mail server to notify changes or bugs within our infrastructure to the other students, and lastly make a file storage to store their application, we chose to use S3 bucket.

**Important note:** Some plans were changed since most programming students did not pass their part of the project, so we had to work along with our tutor only on the project.

Another thing that is also important to keep in mind was the costs. Every feature that you use in AWS comes with a cost, and overtime those costs can build up to quite a large amount. That is why there is tool to help you estimate how much you are going to use. For the Multidisciplinary project we made sure to the costs will not be too heavy while also maintaining decent performance with all necessary features required to run the application. Below you can the estimation for this project:

My Estimate <a href="#">Edit</a> <a href="#">Info</a>		
Estimate summary <a href="#">Info</a>		
Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	68.30 USD	819.60 USD

## 2. System Setup

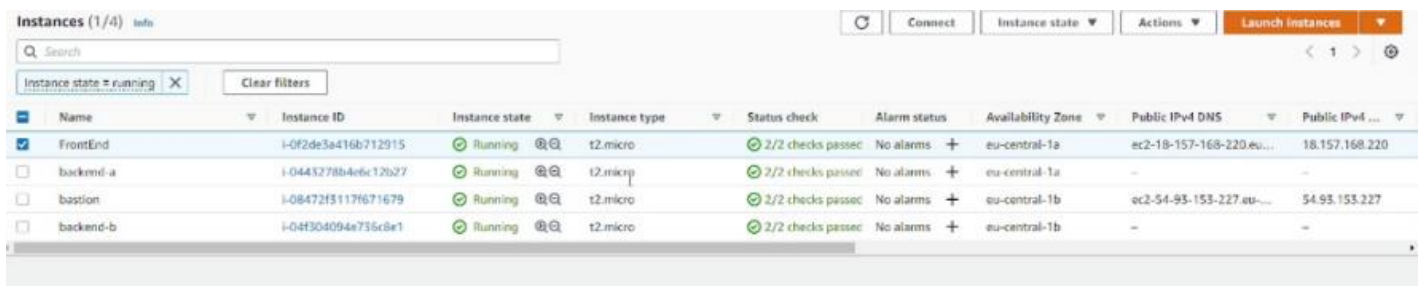
We will create our infrastructure within AWS, we will create our virtual machines here along with other necessary configurations, such as the Security Groups, elastic IP's, VPC, RDS, Load balancer, Availability zones, Route53, etc. Within said virtual machines we will also run Terraform, Ansible and PowerShell scripts.

We made a handful of virtual machines through EC2. These machines are mainly to run our hosted site, making sure the workload is manageable through load balancer, to run the database, and to make a link from the frontend and the backend. All of the instances can run the same or different instance types: t2.nano, t2.micro, t2, medium, etc.

We will have 2 public subnets and 2 private subnets that both run DHCP. So with every deployment the IP changes for every instance. We left it like this as a security measurement.

The instances that we created for the project are:

- FrontEnd:
  - **CPU:** 1 virtual CPU that runs with 1 core
  - **RAM:** 1GB of RAM
  - **OS:** Architecture of i386 or x86\_64
  - **Network:** 10.10.1.X
  - **Storage:** 10GB of Storage (HDD)
- backend a:
  - **CPU:** 1 virtual CPU that runs with 1 core
  - **RAM:** 1GB of RAM
  - **OS:** Architecture of i386 or x86\_64
  - **Network:** 10.10.3.X
  - **Storage:** 10GB of Storage (HDD)
- backend b:
  - **CPU:** 1 virtual CPU that runs with 1 core
  - **RAM:** 1GB of RAM
  - **OS:** Architecture of i386 or x86\_64
  - **Network:** 10.10.4.X
  - **Storage:** 10GB of Storage (HDD)
- bastion:
  - **CPU:** 1 virtual CPU that runs with 1 core
  - **RAM:** 1GB of RAM
  - **OS:** Architecture of i386 or x86\_64
  - **Network:** 10.10.2.X
  - **Storage:** 10GB of Storage (HDD)

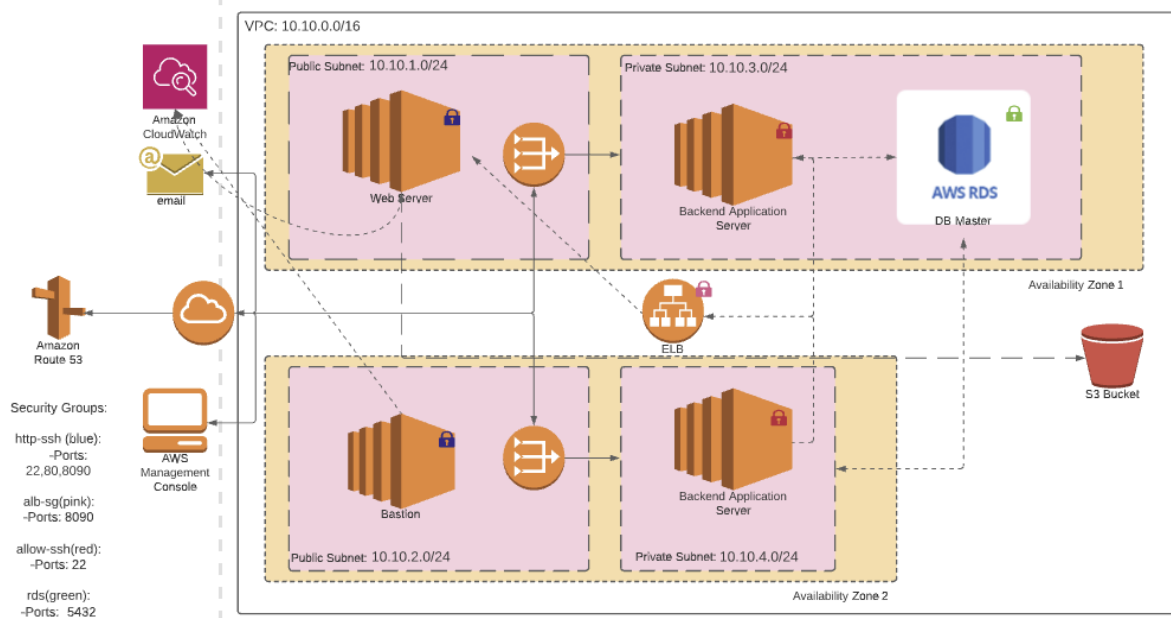


Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
FrontEnd	i-0f2de3a416b712915	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1a	ec2-18-157-168-220.eu...	18.157.168.220
backend-a	i-0445278b4e6c12b27	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1a	-	-
bastion	i-08472f5117671679	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1b	ec2-54-93-153-227.eu...	54.93.153.227
backend-b	i-04f304094e735c8e1	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1b	-	-

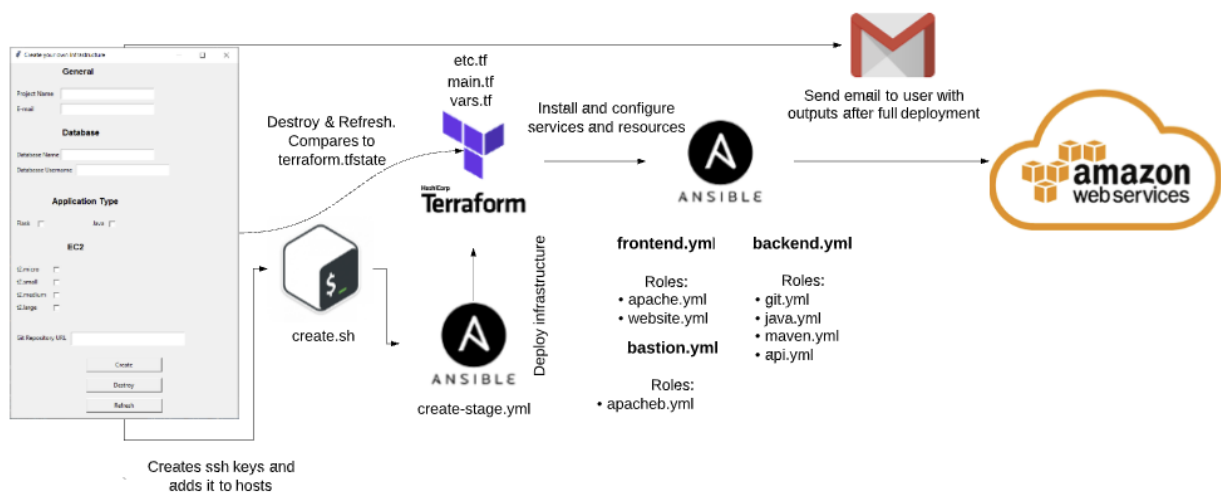
### 3. Network configuration

#### 3.1 Network diagram and description

Here we will explain our network diagram and how it is configured in detail. Below you can see our network infrastructure that we made for our network. This design was created with intention to run every required feature necessary while being as cost efficient as possible, since every feature on AWS chips away a bit on the total monthly costs.



Below you can see how the infrastructure would be created with the help of a GUI that we used for another project (to simplify things for the clients).



## 3.2 Description of the services

In this section we will provide the services to we are currently using in our infrastructure. The information of the services will be shown below in the table.

Service	Description
<b>EC2</b>	EC2 Instances: The EC2 instances will serve as virtual machines in the cloud. This is a main requirement because we will use the EC2 instances to serve all the features and services that the team is offering in the infrastructure such as web server, monitoring tools and security settings.
<b>Elastic IP</b>	With Elastic IP you make sure that every instance that has the permission to get an IP gets a unique public IP assigned to them.
<b>VPC</b>	The Virtual Private Cloud makes sure that everything is running in a private space. If you have an instance with a public IP you can connect to the internet, though you have to be careful since you are leaving a door open for hackers if you enter an unprotected site or download a sketchy file online.
<b>RDS</b>	RDS will be the database for the application. The RDS will be in a private subnet so it would not be open publicly. The RDS can be handful of databases such as PostgreSQL, MySQL, Amazon Aurora, and more.
<b>Load Balancer</b>	The Elastic Load Balancer will help balancing the traffic between the front-end to the back-end API. For example, if one out of the two back-end instances than the website will still work.
<b>CloudWatch Monitoring</b>	CloudWatch simple monitoring is enabled for all the instances. This will take simple metrics such as CPU and memory usage, as well as network traffic.
<b>Route53</b>	Route53 DNS is used to route the front end of the website to the DNS available. In this case, the DNS is "sixitnews.click". You can also configure Route53 failover: it will add resiliency to the infrastructure. If the front-end instance is down, the failover record will redirect the DNS to the bastion web server.
<b>S3 Bucket</b>	With S3 bucket you can save pretty much everything within it, kind of like a database, but much more simplified. You can save config files, pictures, documents, etc. In our case we chose to use them to save RDS snapshots, serving as a backup system.
<b>Availability Zones</b>	Availability Zones help you with redundancy and better connection to AWS. If you are planning to set up an infrastructure you should make sure that you are setting it up within your preferred AWS server. There servers located all over the world.
<b>Security Groups</b>	Security groups helps you set up rules for users joining in your AWS infrastructure. These rules can be anything from what they are allowed to open, what are allowed to see, what they are allowed to edit or modify.
<b>Apache Web Server</b>	The Apache web server was the web server of choice as it supports multiple programming languages.



<b>Flow Logs</b>	Used to display all the network traffic, including the rejected and accepted traffic happening in the VPC.
<b>GitLab CI/CD Pipelines</b>	Gitlab CI/CD pipelines will help on the continuous deployment and integration of the application. This will help us make changes and deploy the user's application in a more efficient way.
<b>Prometheus</b>	Prometheus will be used to extract data from the front-end and bastion instances. One Prometheus server will be hosted in the front-end.
<b>Node Exporter</b>	Will increase the amount of data that can be extracted to monitor the instances hosting a web server.
<b>Grafana</b>	We chose Grafana to display the data extracted from Prometheus and Node Exporter.

## 4. Conclusion

This concludes our design document for the Multidisciplinary Project. We did a lot of work throughout these 17 weeks of the project. We probably missed a few things here and there, but most of the important details are written down. We had a lot of headaches making this projects with all the errors we encountered, but at the end with manage to make everything work at the end. As a group we got along very well and will continue to encourage each other in the future.