# Design document

## Contents

# 1. Introduction

This document will provide detailed information about the project and its planned features. The project is about designing and creating a fully working architecture of WordPress web servers in the AWS cloud that is secured, scalable and reliable. This will include the implementation of some services that AWS offers like VPC, EC2 or EFS. This will give me personally all the experience with working with AWS and Terraform and some real-world implementation of simple architectures.

# 2. Requirements

The goal of the project is to deliver to a potential client a fully working WordPress architecture that will be:

- **Ready to use "out of the box"** - no additional advanced configuration needed by the client.
- **Cost efficient** - we don't want the client to pay a lot for a not-that-very-complex architecture.
- **Reliable and scalable** - we want to ensure 99.99% availability of the project, so there is almost no downtime of the website.
- **Secured** - we don't want to servers get hijacked and users' data stolen.

# 3. Design

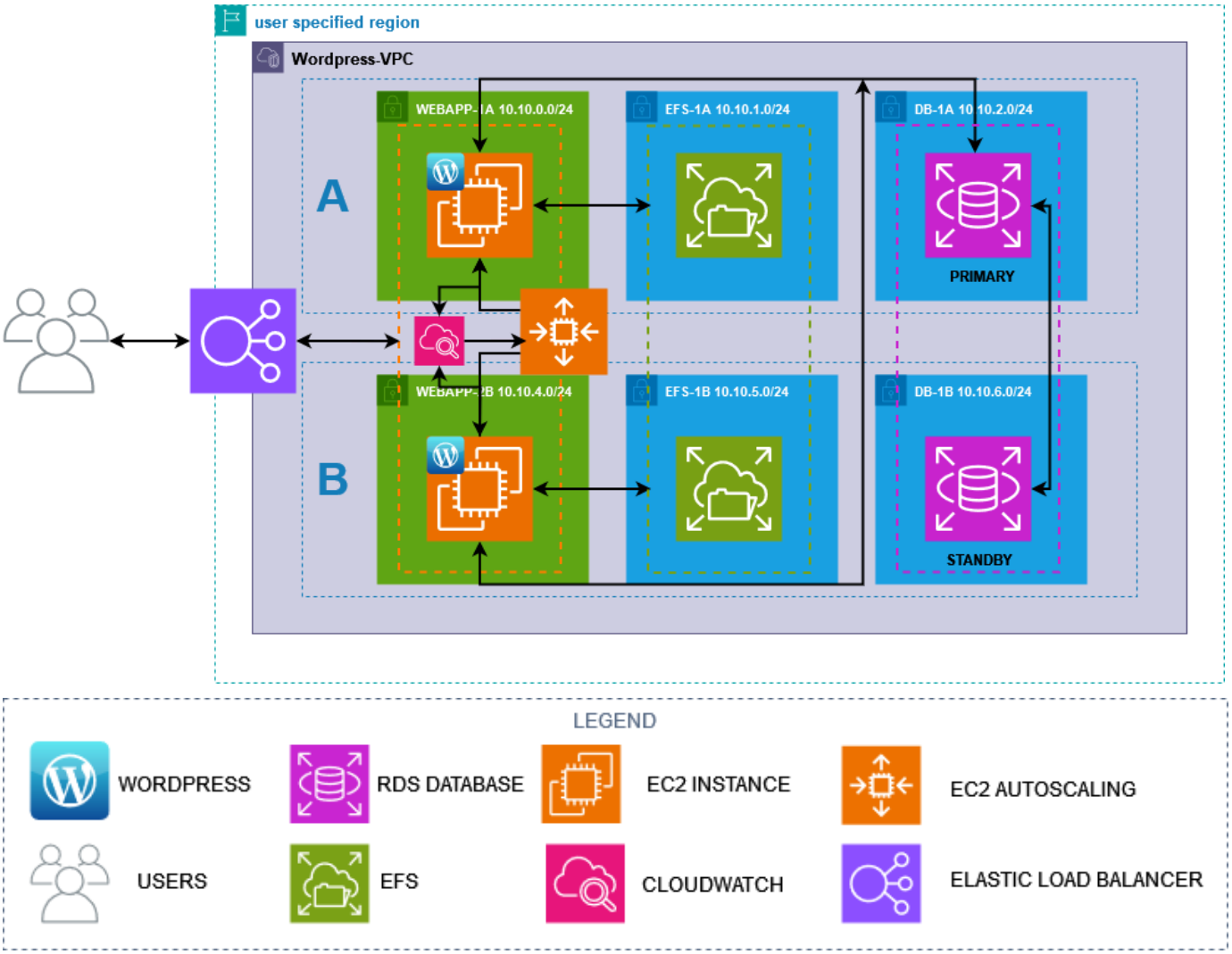
- The whole project will be in a **user-specified region.**
- Custom VPC with CIDR of **10.10.0.0/21** - this gives us **2046** addresses to allocate - for smaller projects is enough.
- VPC will be allocated in **two first availability zones of a region** - to provide some failover and load balancing between different locations.
- There will be three tiers of segmentation - **web app, efs and db.**
- This gives us 3 x 2 **= 6 subnets required**. We need to change the **mask to 24** to allocate 6 subnets. We will have **254** usable IP addresses per subnet. There will be two additional subnets reserved for future scalability.
- There will be **1 web application ec2 instance** containing our web application in a randomly selected AZ that will be deployed by an **auto-scaling group** using a **lunch template with user data**. If the **average load of the CPU is higher than 75%**, it will **deploy a second one** in the other AZ. If the **average load of the CPU is lower than 25%**, it will **remove the second ec2 instance to ensure cost optimisation.**
- The instance type used for the project will be **t3.micro** - to provide enough resources and to be cost-optimized
- **User-data** will contain a **bash script** that will automatically install WordPress, configure it and ensure connectivity with RDS and EFS.

- EC2 will be secured by a **security group (“webappSG”) and instance profile ("wordpressInstanceProfile")**.
- The security group will have **only open inbound port 80** to the entire internet to ensure connectivity through HTTP and will be **open entirely outbound** to ensure connectivity to the Internet.
- The instance profile will contain **two permission policies**:
    - **AmazonSSMManagedInstanceCore** - to make instances connectable through session manager (for security purposes)
    - **AmazonElasticFileSystemClientFullAccess** - to make sure instances have all permission to be able to do write and read operations on EFS service.
- **ALB** will be implemented to load balance between ec2 instances when the load is high/one of the servers is unhealthy. Also, this will give a solution to not connect directly to an ec2 instance but to connect through a load balancer, so this gives more security to the servers.
- There will be a **1 RDS database with a multi-az function.** This means we will have **one primary RDS database in one AZ and one standby RDS database in the other AZ**. The main RDS will be used for everything, and the standby will be used only for data replication and backup.
- RDS will use **rdsSG** with **port 3306 open** for **MySQL connections**. Password for RDS will be **randomized** by terraform and the instance type used by RDS will be **db.t3.micro**
- **EFS** will be implemented to hold WordPress files for the ec2 instances. This will guarantee persistent data between servers and make them almost identical.
- EFS mount targets **will be deployed in two EFS subnets to ensure high availability**, also it will use **efsSG** with **port open 2049 for NFS connections**.
- Everything in the project will be written in **Terraform** files so there will be automation included in the project. The client won’t have to click everything and set up everything by himself.
- Terraform will request **the user to choose a particular AWS region** and it will get all necessary region-based resources **automatically** (for example AMI, availability zones)
- After the creation of infrastructure, Terraform will display outputs like **ec2 AMI used, first availability zone, second availability zone, load balancer DNS name** (for accessing WordPress in the browser), **RDS username and password, region name and VPC CIDR.**

- IP addressing of the subnets:

| CIDR: 10.10.0.0/21 | 1A | 1B |
|---|---|---|
| WEBAPP (public) | 10.10.0.0/24 | 10.10.4.0/24 |
| EFS (private) | 10.10.1.0/24 | 10.10.5.0/24 |
| DB (private) | 10.10.2.0/24 | 10.10.6.0/24 |
| RESERVED (private) | 10.10.3.0/24 | 10.10.7.0/24 |

- Diagram of the whole infrastructure:



# 4. Conclusion

In this document, you have seen the design of the entire architecture for deploying WordPress in the AWS Cloud. We have detailed why specific AWS services were chosen to maximize the efficiency and performance of the WordPress architecture. With services like EC2, ASG, ALB, EFS, and RDS, we can ensure cost efficiency, high availability, and robust security within the architecture, providing additional benefits such as persistent storage and failovers. This document enables us to preplan effectively, ensuring that all stakeholder requirements are met and the product is delivered with high quality.