WEB PROJECT

# First deliverable || **Deployment schema approach**

Marc Perez Arnaiz
David Jovellar Fantova
Aitor Zaragoza Galiana
Edgar Moreno Molina

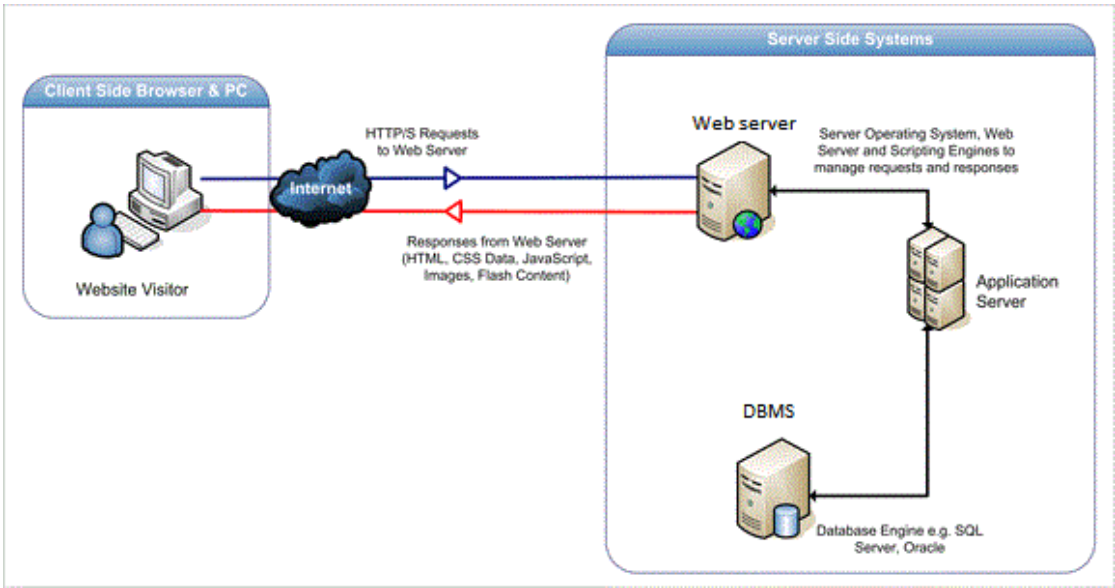
# 1. Introduction

Notice the following proposal solution has been reached by **taking into account: security and efficiency**, leaning towards security. Another key factor is **scalability**, we should never take for granted the scalability factor. We should point that building a deploying approach is hard and, in fact, just an approach as the administrators do not know beforehand which will be the traffic of the network. That traffic will only depend on the users interactions, that is why the approach is probably going to need to be modified over time. The administrator's job is to consider and analyze the previous factors: security, efficiency and scalability, so the number of changes of the initial approach are as low as possible.

Let's take a minute to consider what the following situation's consequences: imagine we have one server that holds both the database and the web server itself, the answers for queries on the database would reach the web server really fast as the web server and database, are in fact in the same machine, so it would be indeed really fast. However, security wise, in case that one server is compromised then the whole web server and database would be compromised. Another approach could be to have the web server and database in different servers, on this approach efficiency would depend on the speed of the connection between servers but would tackle the security issue mentioned above.

## 2. Approach

The idea is to keep the **web, application and database in different servers**, which all will **depend on each other**. However the client will only interact with the web server itself via petitions. The image shown below illustrates the whole point:

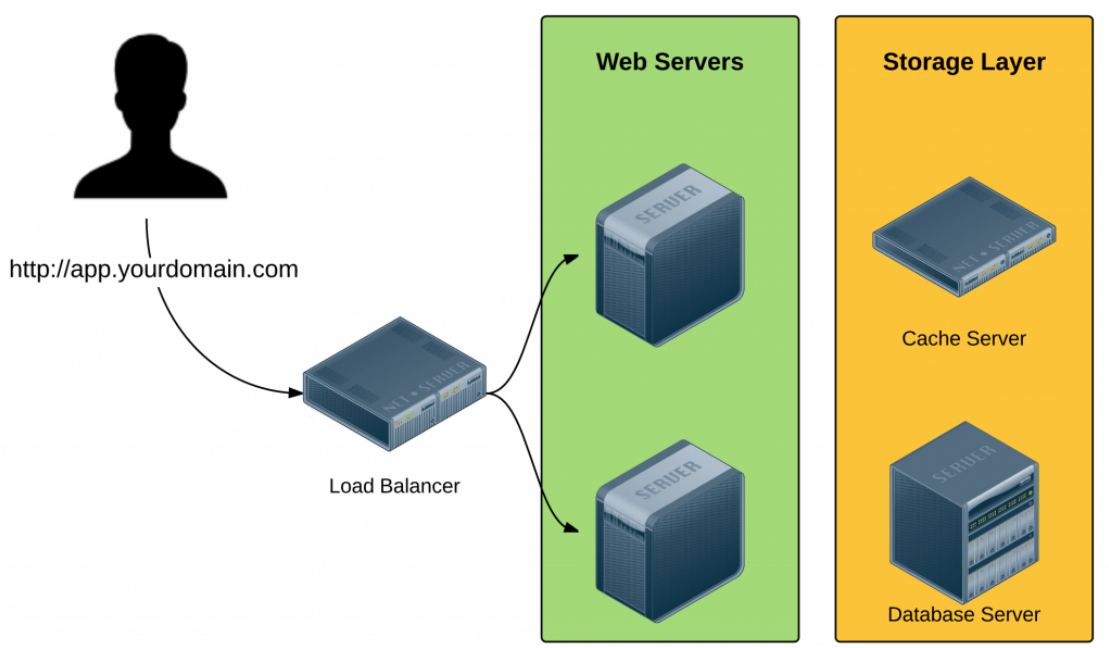



The functions of each server are:

- **Web server**: handles web requests sent by visitors when visiting the website, e.g. gunicorn.
- **Application server**: works between the web server and database server and processes the data.
- **Database server**: handles de database queries, e.g. mysql, postgresql.

A really simple approach to begin with could be the following: having a **load balancer** between the user and web server, that would allow the application to grow over time as well as handling temporary surges on traffic. We would then have **2 web servers** which would get requests from the load balancer.

Now, let's see what approach we take for the **database servers**: In every web application there is usually a few common queries that are repeated over time and take the most resources, the idea is to run an analysis on for instance what are the queries that take the longest time to complete and simply save them on a **cache server**. So we would have one cache server and one database server. This cache server would not necessary be needed but is **optional**, however it could make an improvement efficiency wise.




We would then have two servers for the database for each web server (database cache server and another whole database server). So if we have two web servers we would have **four databases servers** in total (two database caches servers and two whole database server). That means we would also have **four application servers**.

Keep in mind the approach is just an initial approach and could change over time as stated in the beginning of the document.