Common Server Setups For Your Web Application - Part I

Introduction

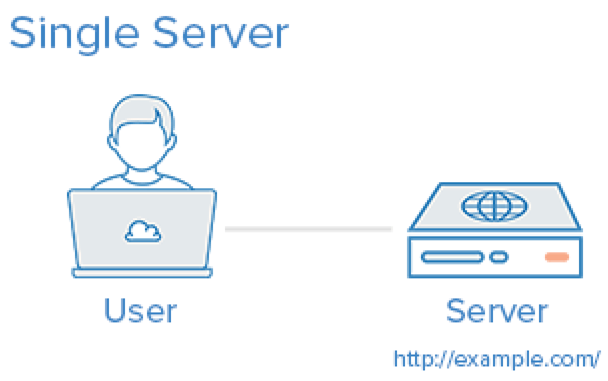
When deciding which server architecture to use for your environment, there are many factors to consider, such as performance, scalability, availability, reliability, cost, and ease of management.

Here is a list of commonly used server setups, with a short description of each, including pros and cons. Keep in mind that all of the concepts covered here can be used in various combinations with one another, and that every environment has different requirements, so there is no single, correct configuration.

1. Everything On One Server

The entire environment resides on a single server. For a typical web application, that would include the web server, application server, and database server. A common variation of this setup is a LAMP stack, which stands for Linux, Apache, MySQL, and PHP, on a single server.

Use Case: Good for setting up an application quickly, as it is the simplest setup possible, but it offers little in the way of scalability and component isolation.



Pros:

- Simple

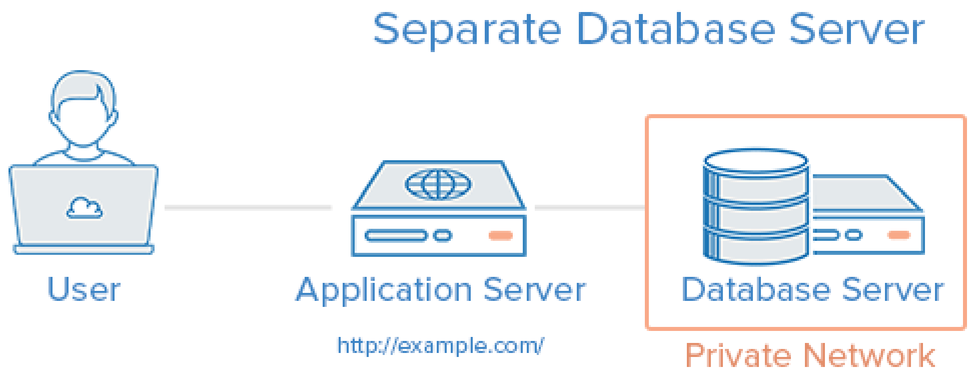
Cons:

- Application and database contend for the same server resources (CPU, Memory, I/O, etc.) which, aside from possible poor performance, can make it difficult to determine the source (application or database) of poor performance
- Not readily horizontally scalable

2. Separate Database Server

The database management system (DBMS) can be separated from the rest of the environment to eliminate the resource contention between the application and the database, and to increase security by removing the database from the DMZ, or public internet.

Use Case: Good for setting up an application quickly, but keeps application and database from fighting over the same system resources.



Pros:

- Application and database tiers do not contend for the same server resources (CPU, Memory, I/O, etc.)

- You may vertically scale each tier separately, by adding more resources to whichever server needs increased capacity
- Depending on your setup, it may increase security by removing your database from the DMZ

Cons:

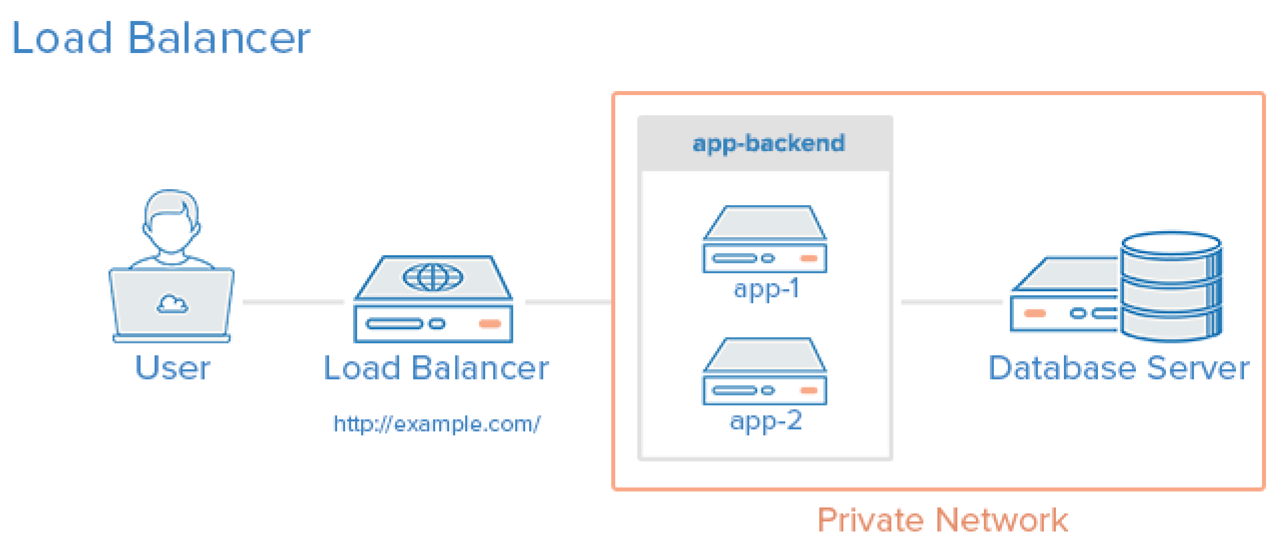
- Slightly more complex setup than single server
- Performance issues can arise if the network connection between the two servers is high-latency (i.e. the servers are geographically distant from each other), or the bandwidth is too low for the amount of data being transferred

Load Balancer (Reverse Proxy)

Load balancers can be added to a server environment to improve performance and reliability by distributing the workload across multiple servers. If one of the servers that is load balanced fails, the other servers will handle the incoming traffic until the failed server becomes healthy again. It can also be used to serve multiple applications through the same domain and port, by using a layer 7 (application layer) reverse proxy.

Examples of software capable of reverse proxy load balancing: HAProxy, Nginx, and Varnish.

Use Case: Useful in an environment that requires scaling by adding more servers, also known as horizontal scaling.



Pros:

- Enables horizontal scaling, i.e. environment capacity can be scaled by adding more servers to it
- Can protect against DDOS attacks by limiting client connections to a sensible amount and frequency

Cons:

- The load balancer can become a performance bottleneck if it does not have enough resources, or if it is configured poorly
- Can introduce complexities that require additional consideration, such as where to perform SSL termination and how to handle applications that require sticky sessions

Source : https://www.digitalocean.com/community/tutorials/5-common-server-setups-for-your-web-application