Course Project

Nicolas El Khoury

Introduction

Web application development is a technology field that is rapidly evolving and growing. Traditionally, applications were designed and implemented using a Monolithic architectural style, with which the application was developed and deployed as a single component, divided into multiple modules. Monolithic applications are very easy to develop and deploy. However, such an architectural pattern becomes a burden once the application becomes too large.

With the emergence of Cloud Computing, and the concept of the on-demand provisioning of resources, a more suitable architectural pattern was required. Microservices rapidly gained popularity, and became a widely used architectural pattern, especially for applications deployed on the cloud.

The purpose of this project is to introduce the basics of web development from an architectural and deployment perspective. As a matter of fact, software engineers today are required to collaborate together across departments (e.g., Developers, QA, System Administrators, etc), rather than work in silos.

Project Description

Design and Deployment of Static Websites.

Requirements

A client has developed two basic static websites, and requires them to be deployed and served. Below are the client requirements:

- **Deployment Mode**: one Virtual Machine (Single Instance).
- **Operating System:** Ubuntu 20.04.
- **Website 1:**
  - **host**: website-one.mywebsite.com
  - **port**: 82
  - **code:**

```
<!DOCTYPE html>
<html>
    <head>
        <title>First Website</title>
    </head>
    <body>
        <p>This is the first website.</p>
    </body>
</html>
```

- **Website 2:**
  - **host**: website-two.mywebsite.com
  - **port**: 83
  - **code:**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Second Website</title>
    </head>
    <body>
        <p>This is the second website.</p>
    </body>
</html>
```

Expected Output

- Website 1 deployed and configured to be served through the following address: ***http://website-one.mywebsite.com:82***

- Website 2 deployed and configured to be served through the following address: ***http://website-two.mywebsite.com:83***
- A document detailing and explaining the steps and decisions taken.
- A demo for the solution

Hints and steps

To successfully complete the project, take into consideration the following:

- Deploy Apache2.
- Configure the Virtual host for website 1.
- Deploy and test website 1.
- Configure the Virtual host for website 2.
- Deploy and test website 2.
- The hostnames will only work if you modify the hosts file on your machine.

Design and Deployment of Web Applications

Requirements

The NK Backend Service is an open source project that serves as a basic backend service. The service is developed using NodeJS, and SailsJS (a framework on top of NodeJS to build Node applications). The service uses ArangoDB to store and manage its data.

Deploy the NK Backend Service with the following requirements:

- **Deployment Mode**: one Virtual Machine (Single Instance).
- **Operating System:** Ubuntu 20.04.

Expected Output

- The application must be fully deployed and running on port 85 using the address: `http://<machine-ip>:85`
- All APIs to be reachable and respond appropriately
- The Application must be deployed either on the VM directly or as Docker Containers.

Hints and Steps

To successfully complete the project, take into consideration the following:

- Deploy and configure ArangoDB.
- Deploy all the requirements for the backend application.
- Download the application code.
- Install the dependencies.
- Run the application.

Bonus

The backend application is deployed and served using the address `http://<machine-ip>:85` . In real life scenarios, backend applications are not exposed to the internet directly. One better alternative is to place them behind a web server, which acts as a reverse proxy, accepting the requests, and then routing them to the backend application. To do so in this project, configure Apache2 with another virtual host:

- **Server name:** nk-backend.app.com
- **Port:** 80

Note that the backend application, that is already deployed and is running on port 85 must remain listening on the same port. To complete this step, you may need to make use of the `proxy_pass` directive.