Cloud Infrastructure Management Lab

LAB-2: Configure Dynamic Web Server

In Last Lab, we installed apache2 and configured static webserver. In this Lab, we will configure to run it as dynamic web server serving dynamic web pages such as php , python or jsp.
As it supports modular architecture with wide range of modules, you can configure it to serve dynamic web pages which are best suited for interactive, personalized, or database-driven applications where content changes based on user actions.
When configured with modules like mod_php, mod_python, or proxy settings for application servers (e.g., Node.js or Django), Apache2 serves dynamic content by executing server-side code and generating content on the fly.

Steps to serve PHP:

1. Start apache2
   sudo systemctl start apache2

2. Update the local package index to reflect the latest upstream changes
   sudo apt update

3. Access your server's IP address in the browser:
   http://127.0.0.1/index.php

4. To enable PHP, install PHP and its Apache module:
   php -v
   sudo apt install php libapache2-mod-php -y
   php -v

5. Navigate to one of virtual root directory: *cd /var/www/example1.com/public_html*

6. Create a test PHP file:
   sudo nano index.php

7. Add the following content to the file
   ```
   <?php
      phpinfo();
   ?>
   ```

8. Save and exit (Ctrl+O, Enter, Ctrl+X).

9. Restart Apache:
   sudo systemctl restart apache2

10. Access your server's IP address in the browser:

http://example1.com/index.php

11. Create another test PHP file:
    sudo nano fetchparams.php

12. Add the following content to the file
    ```
    <?php
    // Set the content type to JSON for better readability
    header('Content-Type: application/json');

    // Capture GET and POST parameters
    $getParams = $_GET;
    $postParams = $_POST;

    // Combine GET and POST parameters into one response
    $response = [
    "GET Parameters" => $getParams,
    "POST Parameters" => $postParams,
    ];

    // Print the response in JSON format
    echo json_encode($response, JSON_PRETTY_PRINT);
    ?>
    ```

13. Save and exit (Ctrl+O, Enter, Ctrl+X).

14. Restart Apache:
    sudo systemctl restart apache2

15. Access your server's IP address in the browser:
    http://example1.com/fetchparams.php?name=MIT&place=Manipal

16. Open terminal, enter following
    Curl –X POST -d "fname=FirstName&lname=LastName"
    http://example1.com/fetchparams.php

Steps to serve JSP:

You can configure Apache2 to serve JSP (Java Server Pages) or Servlets by integrating it with a Java application server like Apache Tomcat.

1. Install a Java Development Kit (JDK) required for running JSP and Servlets:
   sudo apt update
   sudo apt install openjdk-17-jdk -y

2. Verify the installation:
   java -version

3. Download Apache Tomcat:
   wget https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.34/bin/apache-tomcat-10.1.34.tar.gz

4. Extract the Archive:
   sudo tar xvf apache-tomcat-10.1.34.tar.gz -C /opt

5. Rename the Directory:
   sudo mv /opt/apache-tomcat-10.1.34 /opt/tomcat

6. Set Permissions:
   sudo chmod +x /opt/tomcat/bin/.sh*

7. Start Tomcat:
   /opt/tomcat/bin/startup.sh

8. Verify Tomcat: Open your browser and visit
   http://example1.com:8080

9. Integrate Apache2 with Tomcat (Using mod_jk or mod_proxy). To enable Apache2 to forward JSP and Servlet requests to Tomcat, you can use mod_proxy (simpler) or mod_jk (advanced). Let us configure it with mod_proxy:

10. Enable mod_proxy Modules. Run the following commands to enable the required modules:
    sudo a2enmod proxy
    sudo a2enmod proxy_http
    sudo a2enmod proxy_ajp

11. Enable the default site if disabled:
    sudo a2ensite 000-default.conf
    systemctl reload apache2

12. Configure Apache to Forward Requests to Tomcat. Edit the default Apache configuration. Add the following lines inside the <VirtualHost *:80> block at the end. In next steps, we are going to install a sample application war file.
    sudo nano /etc/apache2/sites-available/000-default.conf
    ProxyPass /sample http://localhost:8080/sample
    ProxyPassReverse /sample http://localhost:8080/sample

13. Test the Apache configuration for syntax errors:
    sudo apache2ctl configtest

14. Save and exit, then restart Apache:
    sudo systemctl restart apache2

15. deploy your application. Place the .war file or your project folder in Tomcat's webapps directory. Tomcat will automatically deploy the application.:
    sudo cp sample.war /opt/tomcat/webapps/

Note: If sample.war file is not found in Lab Desktop, request for help

16. Access your application via Tomcat:
    http://localhost:8080/sample

17. Access your application via Apache:
    http://localhost/sample

18. Stop Tomcat:
    /opt/tomcat/bin/shutdown.sh

19. Stop Apache2:
    sudo systemctl stop apache2