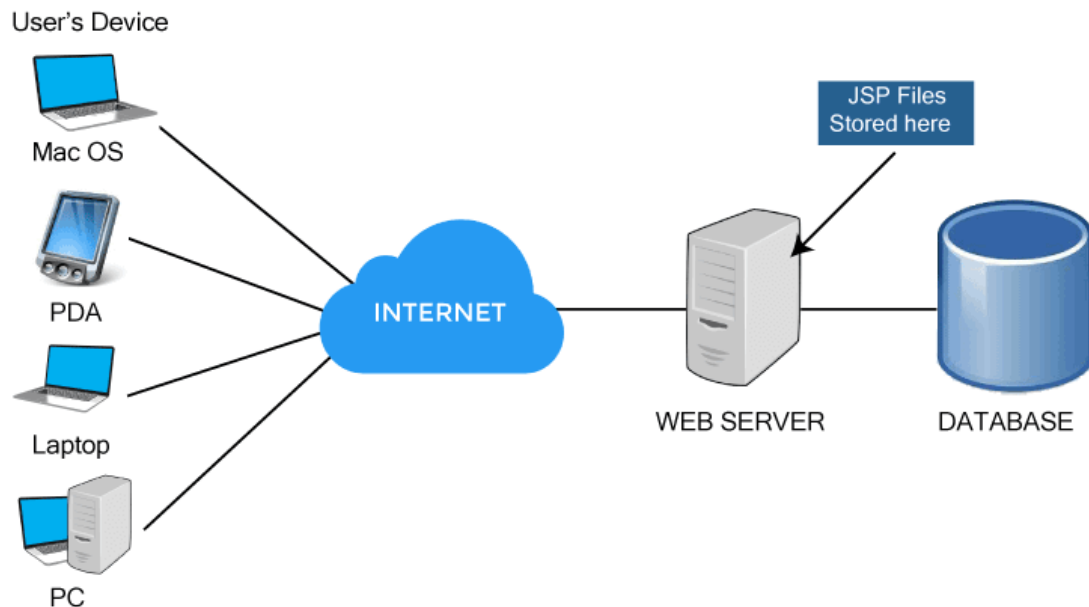
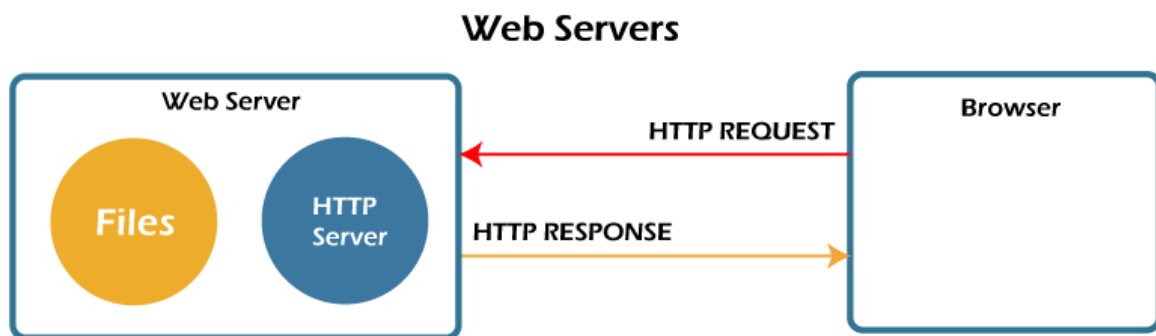


WEB SERVERS

A web server is a software application or hardware device that stores, processes, and serves web content to users over the internet. It plays a critical role in the client-server model of the World Wide Web, where clients (typically web browsers) request web pages and resources, and servers respond to these requests by delivering the requested content.

Web servers operate on the Hypertext Transfer Protocol (HTTP), which is the foundation of data communication on the World Wide Web. When you enter a website's URL into your browser, it sends an HTTP request to the web server hosting that website, which then sends back the web page you requested, allowing you to view it in your browser.



Web Server Architecture

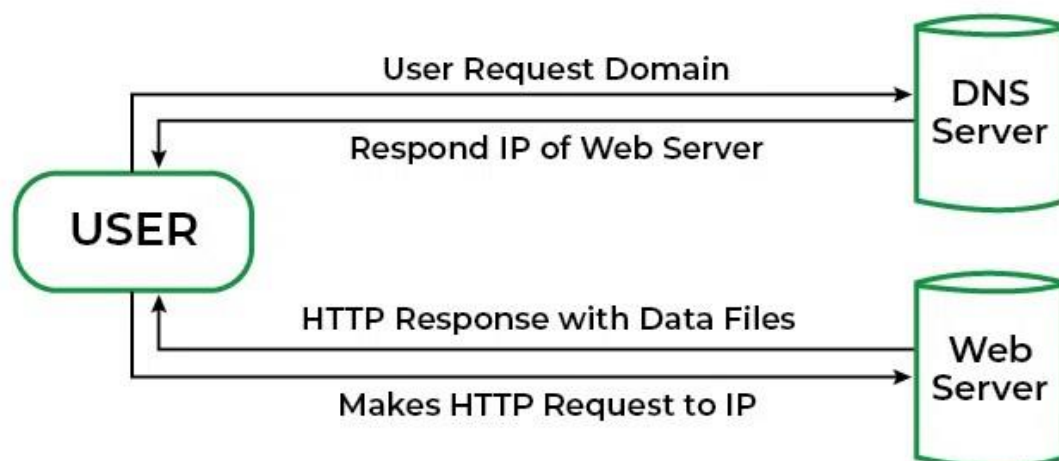
Web server architecture refers to the structure and design of web servers, outlining how they handle incoming requests and deliver web content. There are two main approaches to web server architecture:

Single-Tier (Single Server) Architecture:

In a single-tier architecture, a single server is responsible for both processing requests and serving web content. This is suitable for small websites or applications with low traffic. However, it has limitations in terms of scalability and fault tolerance. If the server goes down, the entire service becomes unavailable



Single Server Architecture of Web Server

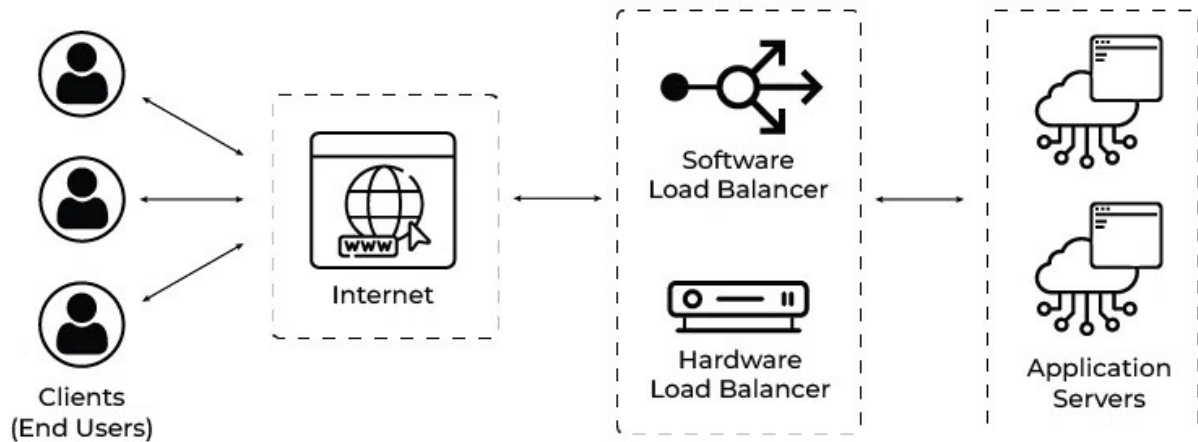


Multi-Tier (Load-Balanced) Architecture:

In a multi-tier architecture, multiple servers are used to distribute the workload and ensure high availability. This approach often involves load balancers that evenly distribute incoming requests across a cluster of web servers. Each server can serve web content independently, and if one server fails, the load balancer redirects traffic to healthy servers, ensuring uninterrupted service.



Load Balance web server architecture



Working of Web Servers

A web server works in the following ways:

Obtain the IP address from domain name: IP address is obtained in two ways either by searching it in the cache or requesting DNS Servers

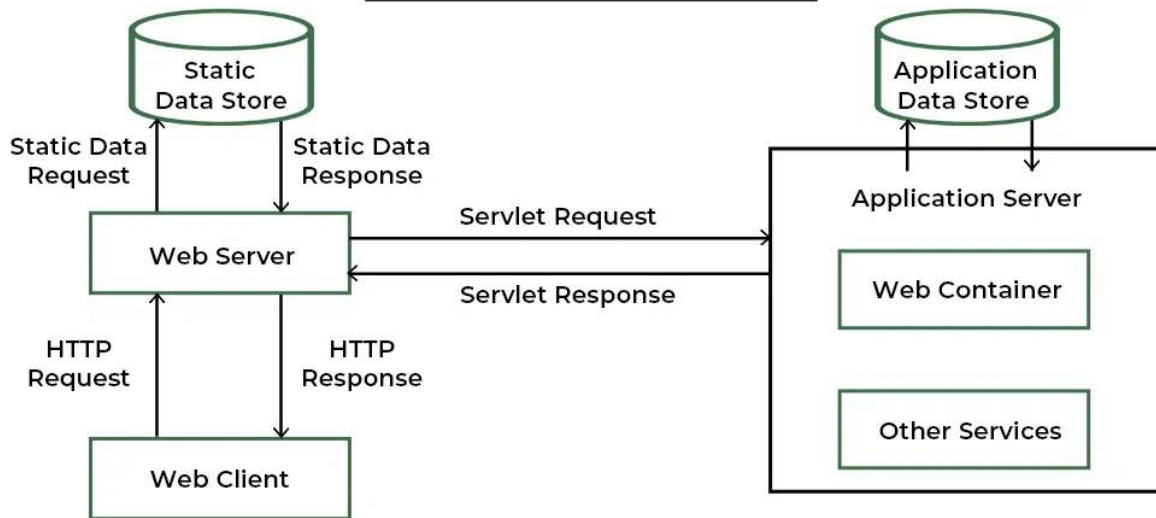
Requests full URL from Browsers: After fetching IP address a full URL is demanded from the web server

Web Server Responds to the request: In accordance with the request a response is sent by the server in case of successful request otherwise appropriate error message is sent

The Web Page is displayed on the browser: After getting the response from the server, the web browser displays the result



Working of Web Server



Types of Web Servers Softwares:

There are several types of web servers, each designed for specific purposes:

- Apache HTTP Server** : Apache is one of the most popular open-source web servers globally, known for its flexibility and robustness. It's highly customizable and supports a wide range of modules and extensions.
- Nginx** : Nginx is another widely used web server known for its speed and efficiency in handling concurrent connections.

Features of Web Servers

Web servers offer a range of features, including:

Content Hosting : They store and serve web content, including HTML pages, images, videos, and other multimedia files.

Security : Web servers implement various security mechanisms to protect against unauthorized access and cyberattacks.

Load Balancing : Some web servers can distribute incoming traffic across multiple server instances to ensure optimal performance and availability.

Logging and Monitoring : They provide tools to track and analyze server performance, user access, and error logs.

Caching : Web servers can cache frequently accessed content to reduce server load and improve response times.

Benefits of Web Servers

Using web servers offers several advantages, including:

Scalability : Web servers can handle a large number of simultaneous connections, making them suitable for high-traffic websites.

Reliability : They are designed for continuous operation and can recover from failures gracefully.

Security : Web servers include security features to protect against common web threats like DDoS attacks and SQL injection.

Customization : Web server configurations can be tailored to specific application requirements.

Uses of Web Server:

Hosting Websites: The most common use of web servers is to host websites, making them accessible on the internet.

Web Applications: Web servers provide the infrastructure for hosting web applications, enabling users to interact with software through a web interface.

File Sharing: Some web servers are used for file sharing and collaboration, allowing users to upload and download files securely.

Content Delivery: Content delivery networks (CDNs) use web servers to distribute content like images and videos to users worldwide, reducing load times.

API Hosting: Web servers are used to host APIs (Application Programming Interfaces) that allow applications to communicate and exchange data over the internet.

Conclusion

Web servers are the backbone of the internet, enabling us to access and interact with websites and web applications every day. Understanding their working and architecture is essential for web developers, administrators, and anyone interested in the digital world. Whether you're running a personal blog or managing a complex e-commerce platform, web servers play a pivotal role in delivering online content efficiently and securely.

Configuring and installing server

1.Ngnix server

Step 1: Install NGINX

```
amazon-linux-extras install nginx1.12
```

Step 2: Start the NGINX service

```
systemctl start nginx
```

Step 3: Verify NGINX service status

```
systemctl status nginx
```

Step 4: Reload NGINX configuration

```
systemctl reload nginx
```

Step 5: Check running processes for NGINX

```
ps -aef | grep nginx
```

Step 6: Check NGINX access and error logs

Change to the NGINX log directory:`cd /var/log/nginx/`

View the access log:`cat access.log`

View the error log:`cat error.log`

Step 7: Edit NGINX's default HTML page or open the chrome with the ip_address

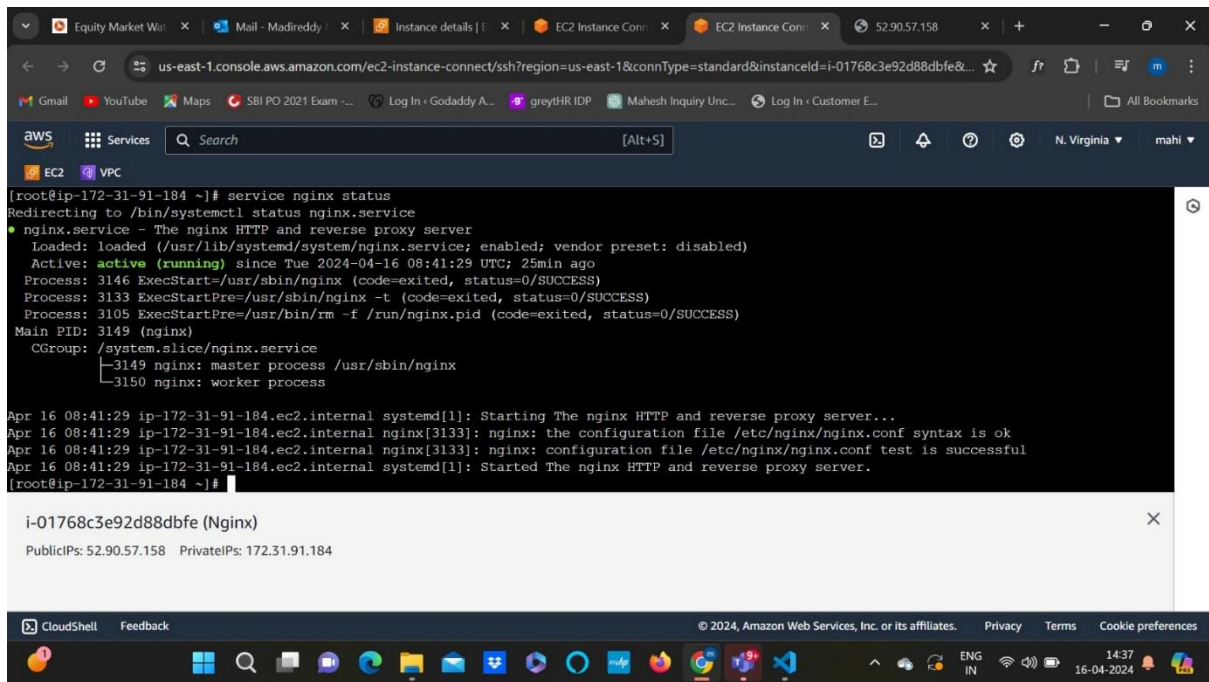
Change to the NGINX HTML directory:`cd /usr/share/nginx/html/`

Open index.html file (`vi index.html`)

Step 8: Reload NGINX after changes

```
systemctl reload nginx
```

step 9:open the chrome with ip_address



```
[root@ip-172-31-91-184 ~]# service nginx status
Redirecting to /bin/systemctl status nginx.service
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-04-16 08:41:29 UTC; 25min ago
     Process: 3146 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
     Process: 3133 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 3105 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Main PID: 3149 (nginx)
   CGroup: /system.slice/nginx.service
           └─3149 nginx: master process /usr/sbin/nginx
             └─3150 nginx: worker process

Apr 16 08:41:29 ip-172-31-91-184.ec2.internal systemd[1]: Starting The nginx HTTP and reverse proxy server...
Apr 16 08:41:29 ip-172-31-91-184.ec2.internal nginx[3133]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Apr 16 08:41:29 ip-172-31-91-184.ec2.internal nginx[3133]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Apr 16 08:41:29 ip-172-31-91-184.ec2.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
[root@ip-172-31-91-184 ~]#
```

i-01768c3e92d88dbfe (Nginx)

PublicIPs: 52.90.57.158 PrivateIPs: 172.31.91.184

2. Apache Tomcat:

1. Install Java Development Kit:

```
yum install java-17 -y
```

```
java -version
```

2. Download and Install Apache Tomcat:

```
wget https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.20/bin/apache-tomcat-10.1.20.tar.gz
```

3. Unzip and install :

```
tar -xvf apache-tomcat-10.1.20.tar.gz
```

4. To check the version (there is no built in command to check the version)

a. First you change the name of the folder (apache-tomcat-10.1.20.tar.gz) for this use this command

```
mv apache-tomcat-10.1.20.tar.gz/ tomcat
```

b. change the directory to 'tomcat'

```
cd tomcat/
```

c. Now check the version:

```
cd bin/
```



```
sh version.sh
```

5. Check the port is in LISTENING state or not

```
netstat -tuln | grep :80
```

6. Start the server:

--> you are in bin folder

```
cd tomcat/bin/
```

```
sh startup.sh
```

7. Copy the public Ip address of your instance and copy in the new browser:

54.175.244.23:8081--->(port number)

8. Check the permissions for manager app:

a. first open all the context.xml in your tomcat

```
find / -name context.xml
```

b. all related context.xml will shown like here"

```
/opt/tomcat/conf/context.xml
```

```
/opt/tomcat/webapps/docs/META-INF/context.xml
```

```
/opt/tomcat/webapps/examples/META-INF/context.xml
```

```
/opt/tomcat/webapps/host-manager/META-INF/context.xml
```

```
/opt/tomcat/webapps/manager/META-INF/context.xml
```

c. open each and every file and check this line and comment it out:

```
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
```

```
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

d. now open another file called tomcat-users.xml:

```
cd conf/
```

```
vi tomcat-users.xml
```

e. add the roles and permissions given below (add all these lines at last in between tomcat-users tag:

```
<role rolename = "manager-gui"/>
```

```
<role rolename = "manger-script"/>
```

```
<role rolename = "manager-jmx"/>
```

```
<role rolename = "manager-status"/>
```

```
<user username = "admin" password= "admin" roles = "manager-  
gui,manager-script,manager-jmx,manager-status"/>
```

```
<user username = "deployer" password = "deployer" roles = "manager-  
script"/>
```

```
<user username = "tomcat" password = "s3cret" roles = "manager-gui"/>
```

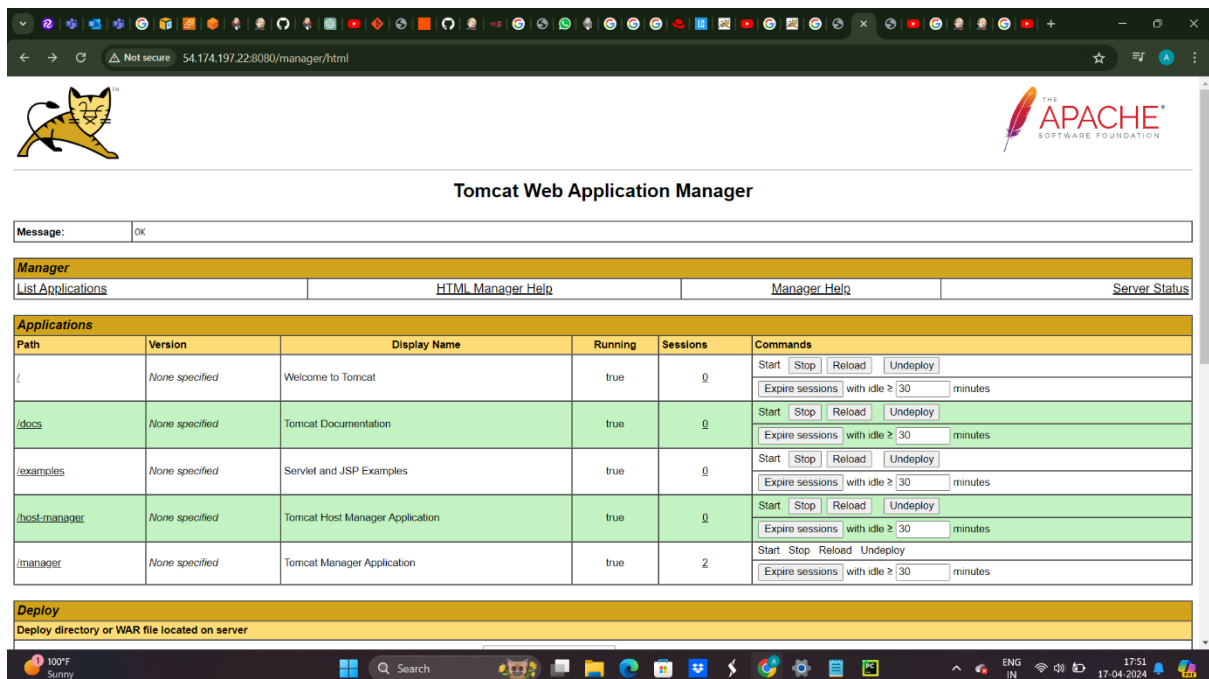
f. save the file

9. Now open the browser and reload it, it will ask username and password:

username:admin

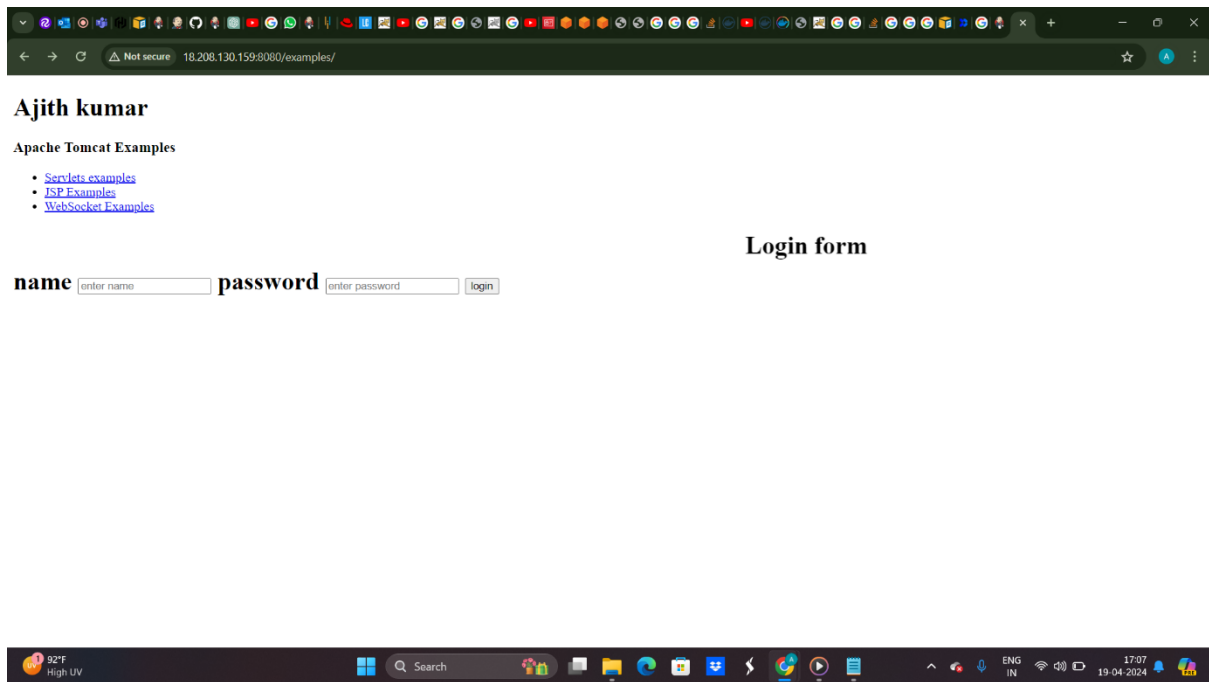
password:admin

10. Now the manager app will open and you want to deploy any file and made any changes for testing.



The screenshot shows the Tomcat Web Application Manager interface in a web browser. The browser address bar shows the URL `54.174.197.22:8080/manager/html`. The page features the Tomcat logo on the left and the Apache Software Foundation logo on the right. Below the logos, the title "Tomcat Web Application Manager" is displayed. A message bar at the top indicates a successful login. The main content area is divided into several sections: "Manager" with links for "List Applications", "HTML Manager Help", "Manager Help", and "Server Status"; "Applications" which contains a table of deployed applications; and "Deploy" which provides a field for deploying a directory or WAR file.

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes



Apache 2 Installation:

1. `sudo -i`: This command allows you to switch to the root user in the terminal session.
2. `apt update`: This command updates the local package index, which is a database of available packages

and their versions. It ensures that your system has the latest information about available packages.

3. `apt install apache2 -y`: This command installs the Apache HTTP server (apache2 package) on your system.

The `-y` flag automatically answers "yes" to any prompts during the installation process, so you don't have to manually confirm.

4. `service apache2 start`: This command starts the Apache HTTP server. It's misspelled as "apache2", so it

won't work correctly. The correct spelling is `apache2`.

5. `service apache2 status`: This command checks the status of the Apache HTTP server to see if it's running

or stopped.

6. `curl -I 3.106.187.125`: This command sends an HTTP HEAD request to the specified URL (3.106.187.125 in

this case) and displays the response headers. It's used to inspect the headers of a web page without downloading the entire page.

7. `cd /var`: This command changes the current directory to `/var`, which is a directory containing variable data files for system services.

8. `ls -al`: This command lists all files and directories in the current directory (`/var` in this case) in a detailed format, including hidden files.

9. `cd www`: This command changes the current directory to `/var/www`, which is the default directory for web content on an Apache server.

10. `ls -al`

11. `cd html`

12. `vim index.html`: This command opens the `index.html` file in the Vim text editor for editing. This file is

typically the main webpage file for a website.

open a new tab copy the public IP address of the server (instance) and check if the output is displayed

correctly or not. If its display correctly that means successful.

13. `cd`: used to change the current working directory

CHECK LOG FILES

`cd /var/log`

`cd opt`: This command changes the current directory to `/opt`, which is a directory for optional software

packages.

`cd log`: This command changes the current directory to `/var/log`, which contains log files for various system

services.

`cd syslog`: This command changes the current directory to `/var/log/syslog`, which is a log file containing

system messages.

`cd /etc/apache2`: This command changes the current directory to `/etc/apache2`, which contains

configuration files for the Apache HT