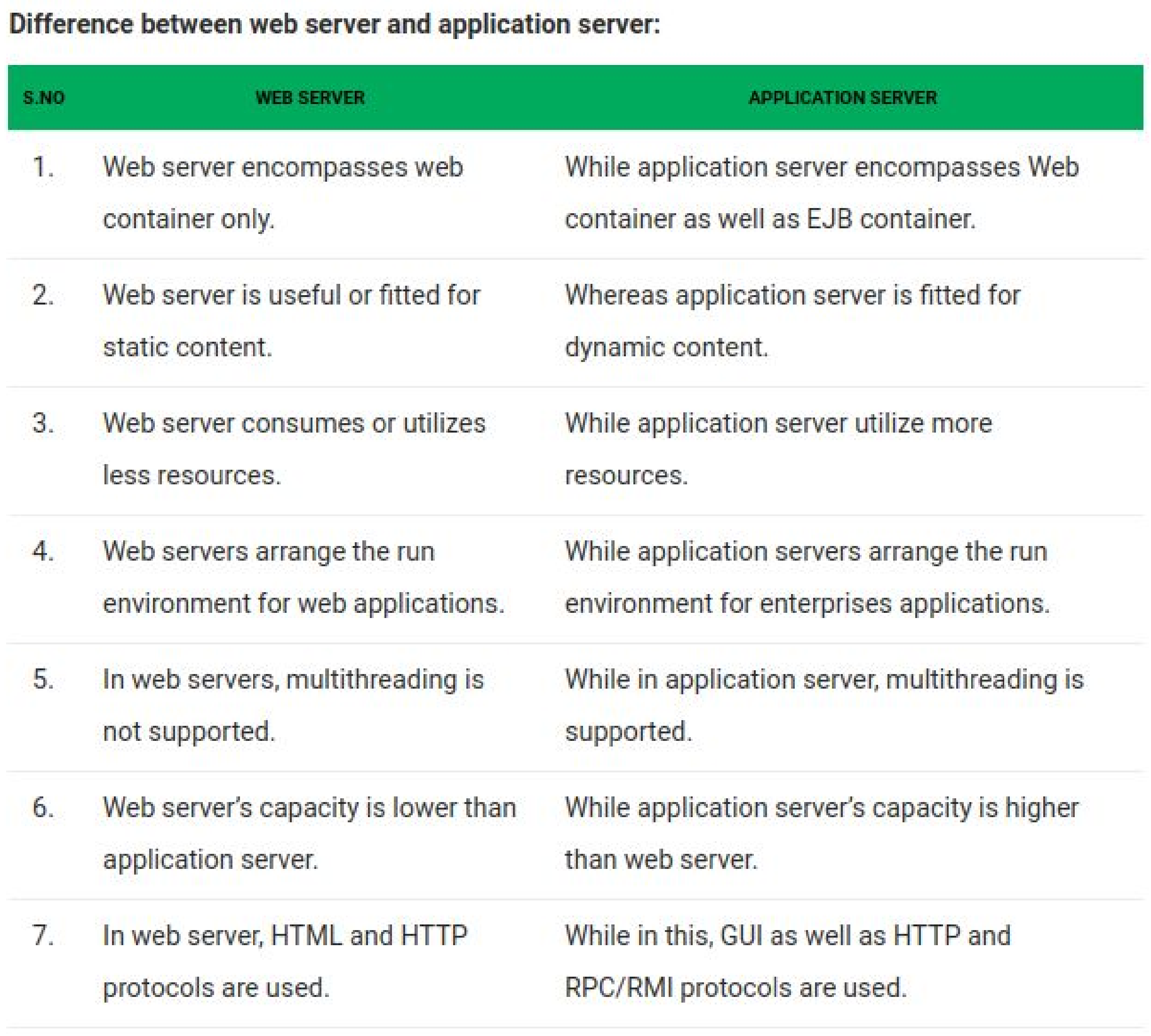
APPLICATION SERVER

Ques 1. What is the difference between an Application Server and a Web Server?

Ans 1.



Difference between web server and application server:

S.NO	WEB SERVER	APPLICATION SERVER
1.	Web server encompasses web container only.	While application server encompasses Web container as well as EJB container.
2.	Web server is useful or fitted for static content.	Whereas application server is fitted for dynamic content.
3.	Web server consumes or utilizes less resources.	While application server utilize more resources.
4.	Web servers arrange the run environment for web applications.	While application servers arrange the run environment for enterprises applications.
5.	In web servers, multithreading is not supported.	While in application server, multithreading is supported.
6.	Web server's capacity is lower than application server.	While application server's capacity is higher than web server.
7.	In web server, HTML and HTTP protocols are used.	While in this, GUI as well as HTTP and RPC/RMI protocols are used.

Ques 2. What is Catalina?

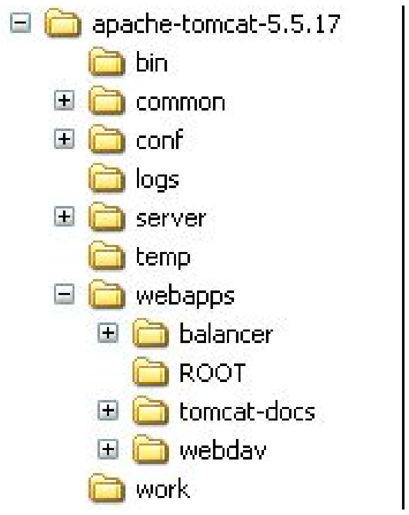
Ans 2.

Catalina is Tomcat's servlet container. Catalina implements Sun Microsystems' specifications for servlet and JavaServer Pages (JSP). In Tomcat, a Realm element represents a "database" of usernames, passwords, and roles (similar to Unix groups) assigned to those users. Different implementations of Realm allow Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification.

Ques 3. Describe tomcat directory structure.

Ans 3.

Once Tomcat has been installed, you will see a directory structure something like:



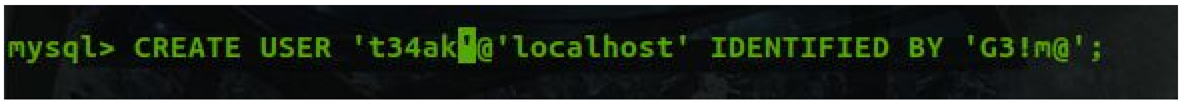
(This one was installed with my NetBeans installation.)

The typical directory hierarchy of a Tomcat installation consists of the following:

- **bin** - startup, shutdown and other scripts and executables
- **common** - common classes that Catalina and web applications can use
- **conf** - XML files and related DTDs to configure Tomcat
- **logs** - Catalina and application logs
- **server** - classes used only by Catalina
- **shared** - classes shared by all web applications
- **webapps** - directory containing the web applications
- **work** - temporary storage for files and directories

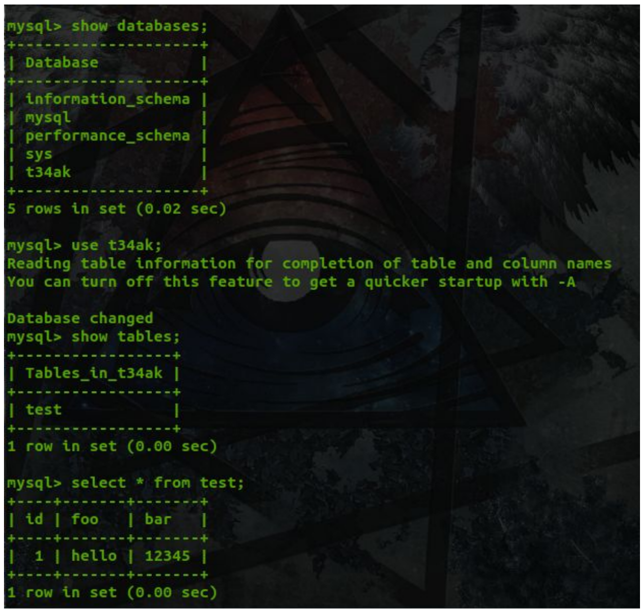
Ques 4. Connect any sample.war to MySQL running on localhost.

Ans 4. First we have to create a mysql account in this case it is t34ak@localhost is the username with "G3!m@" as a password



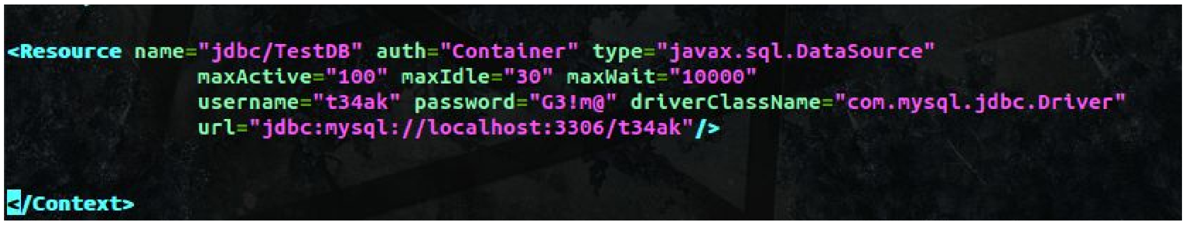
```
mysql> CREATE USER 't34ak'@'localhost' IDENTIFIED BY 'G3!m@';
```

Next we have to make the tables in the database



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| t34ak              |
+--------------------+
5 rows in set (0.02 sec)

mysql> use t34ak;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_t34ak |
+-----------------+
| test            |
+-----------------+
1 row in set (0.00 sec)

mysql> select * from test;
+----+-------+-------+
| id | foo   | bar   |
+----+-------+-------+
|  1 | hello | 12345 |
+----+-------+-------+
1 row in set (0.00 sec)
```

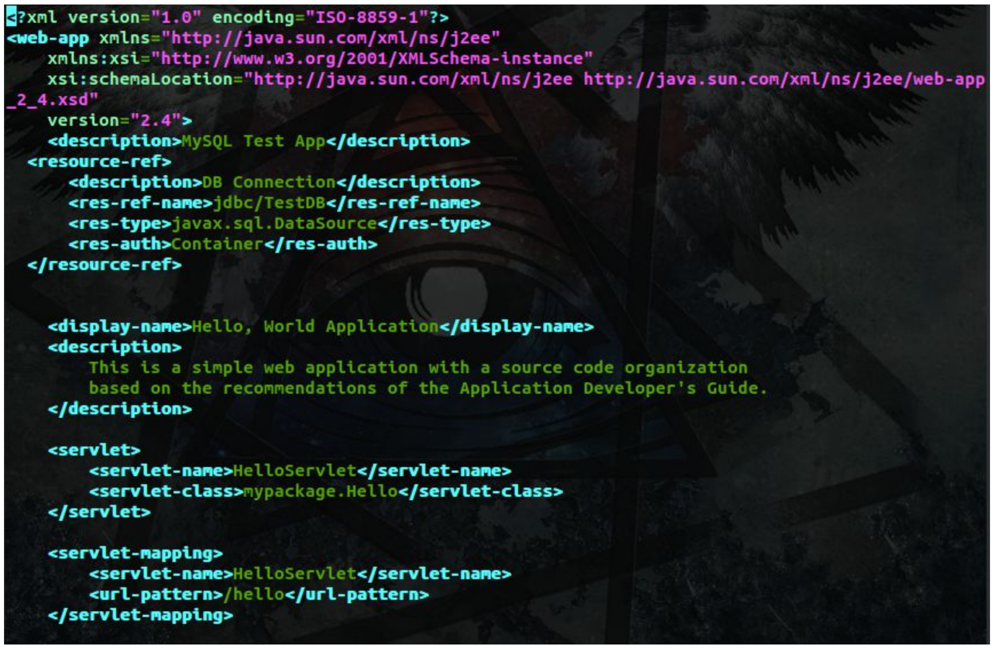
Next we have to make changes in the */var/lib/tomcat9/conf/context.xml* file and create a resource for your mysql credentials. This is use to connect to mysql database with the credentials of the database.



```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
               maxActive="100" maxIdle="30" maxWait="10000"
               username="t34ak" password="G3!m@" driverClassName="com.mysql.jdbc.Driver"
               url="jdbc:mysql://localhost:3306/t34ak"/>

</Context>
```

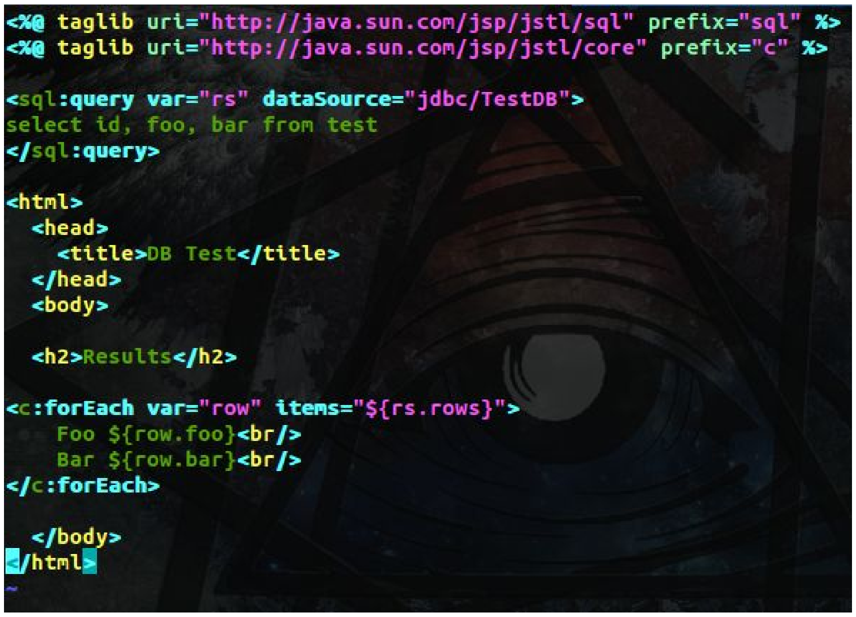
Create the resource reference file web.xml file present in the /var/lib/tomcat9/webapps/sample/WEB-INF/ directory

This file is use to create the reference of the resource which we made in the context.xml .



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app
_2_4.xsd"
    version="2.4">
    <description>MySQL Test App</description>
  <resource-ref>
      <description>DB Connection</description>
      <res-ref-name>jdbc/TestDB</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
  </resource-ref>


    <display-name>Hello, World Application</display-name>
    <description>
        This is a simple web application with a source code organization
        based on the recommendations of the Application Developer's Guide.
    </description>

    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>mypackage.Hello</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
```

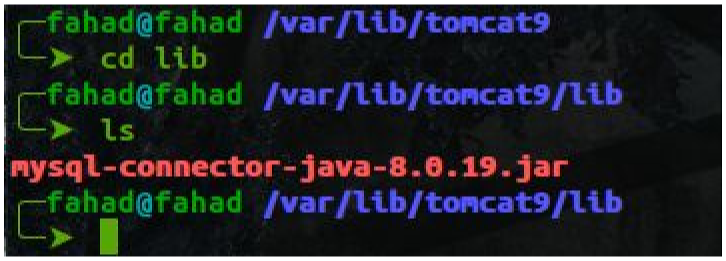
Now we have to create the test.jsp file to show the connectivity between mysql and tomcat



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<sql:query var="rs" dataSource="jdbc/TestDB">
select id, foo, bar from test
</sql:query>

<html>
  <head>
    <title>DB Test</title>
  </head>
  <body>

  <h2>Results</h2>

<c:forEach var="row" items="${rs.rows}">
    Foo ${row.foo}<br/>
    Bar ${row.bar}<br/>
</c:forEach>

  </body>
</html>
~
```

Now we have to download mysql-connector-java-8.0.19.jar and store it to the /var/lib/tomcat9/lib

It isJDBC Connector to retrieve data from mysql database.

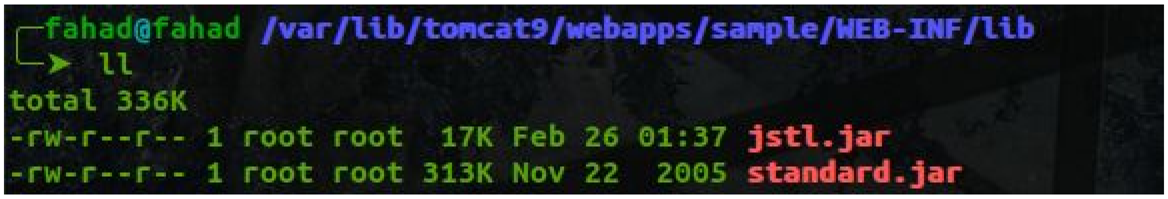


```
┌─fahad@fahad /var/lib/tomcat9
└─➤ cd lib
┌─fahad@fahad /var/lib/tomcat9/lib
└─➤ ls
mysql-connector-java-8.0.19.jar
┌─fahad@fahad /var/lib/tomcat9/lib
└─➤ 
```

We also have to download jstl.jar and standard.jar and paste it to the

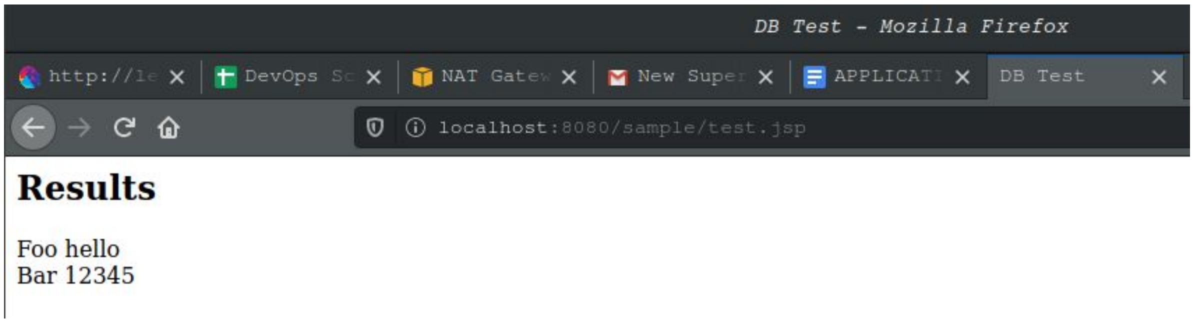
/var/lib/tomcat9/webapps/sample/WEB-INF/lib

It is sue for proper connection of jdbc drivers



```
┌─fahad@fahad /var/lib/tomcat9/webapps/sample/WEB-INF/lib
└─➤ ll
total 336K
-rw-r--r-- 1 root root  17K Feb 26 01:37 jstl.jar
-rw-r--r-- 1 root root 313K Nov 22  2005 standard.jar
```

Then on going to web-browser and typing http://localhost:8080/sample/test.jsp url it will show the result



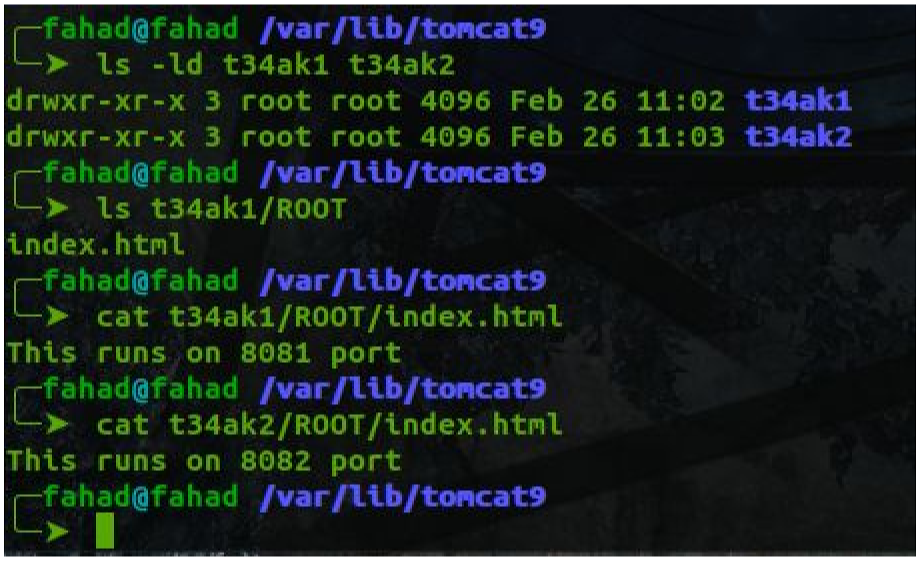
Results

Foo hello
Bar 12345

Ques 5. Run multiple services on different ports with different connectors (AJP/HTTP) on same tomcat installation.

Ans 5.

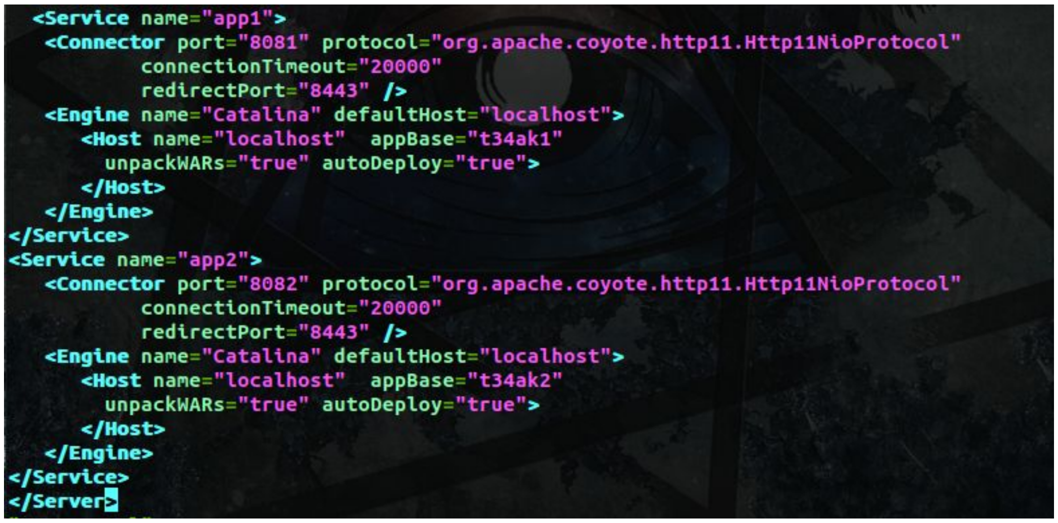
First we have to make two directories and in each directory we have to make the root folder and under which we can type our codes file



```
┌─fahad@fahad /var/lib/tomcat9
└─➤ ls -ld t34ak1 t34ak2
drwxr-xr-x 3 root root 4096 Feb 26 11:02 t34ak1
drwxr-xr-x 3 root root 4096 Feb 26 11:03 t34ak2
┌─fahad@fahad /var/lib/tomcat9
└─➤ ls t34ak1/ROOT
index.html
┌─fahad@fahad /var/lib/tomcat9
└─➤ cat t34ak1/ROOT/index.html
This runs on 8081 port
┌─fahad@fahad /var/lib/tomcat9
└─➤ cat t34ak2/ROOT/index.html
This runs on 8082 port
┌─fahad@fahad /var/lib/tomcat9
└─➤ 
```

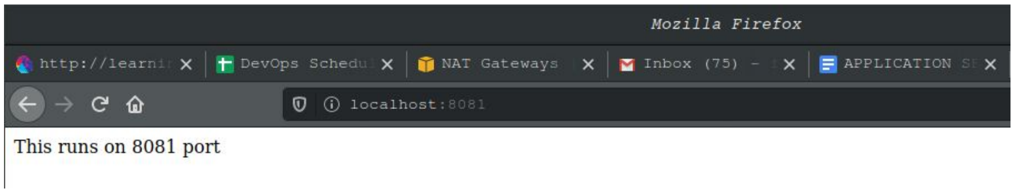
Now we have to add service port and app name in server .xml

For proper working of web apps on different ports



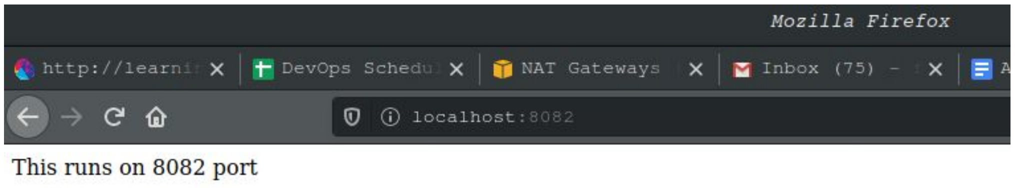
```
  <Service name="app1">
   <Connector port="8081" protocol="org.apache.coyote.http11.Http11NioProtocol"
              connectionTimeout="20000"
              redirectPort="8443" />
   <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost"  appBase="t34ak1"
        unpackWARs="true" autoDeploy="true">
      </Host>
   </Engine>
</Service>
<Service name="app2">
   <Connector port="8082" protocol="org.apache.coyote.http11.Http11NioProtocol"
              connectionTimeout="20000"
              redirectPort="8443" />
   <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost"  appBase="t34ak2"
        unpackWARs="true" autoDeploy="true">
      </Host>
   </Engine>
</Service>
</Server>
```

Then on going to web-browser and typing http://localhost:8081 url it will show the result for first web app



This runs on 8081 port

Then on going to web-browser and typing http://localhost:8082 url it will show the result for second web app



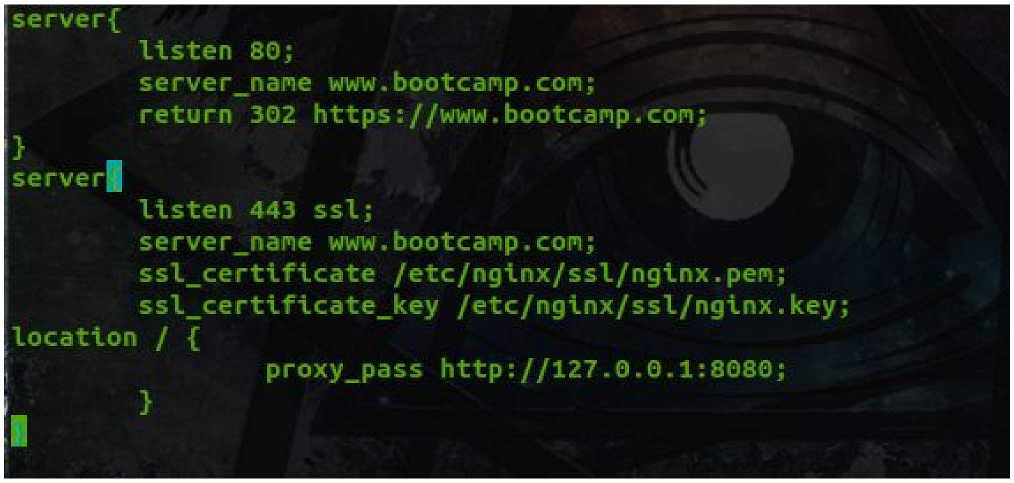
This runs on 8082 port

Ques 6.1 Use nginx as reverse proxy for tomcat application.

Setup self signed certificate on that nginx for bootcamp.com.

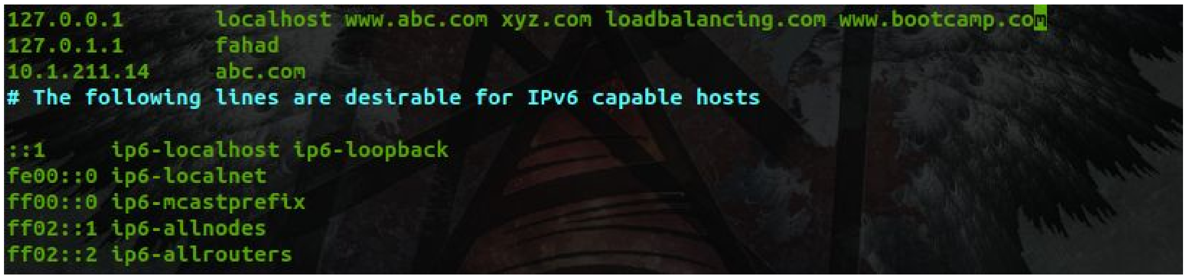
Ans 6.1

First we have to edit the conf file and add the ssl keys to the www.bootcamp.com



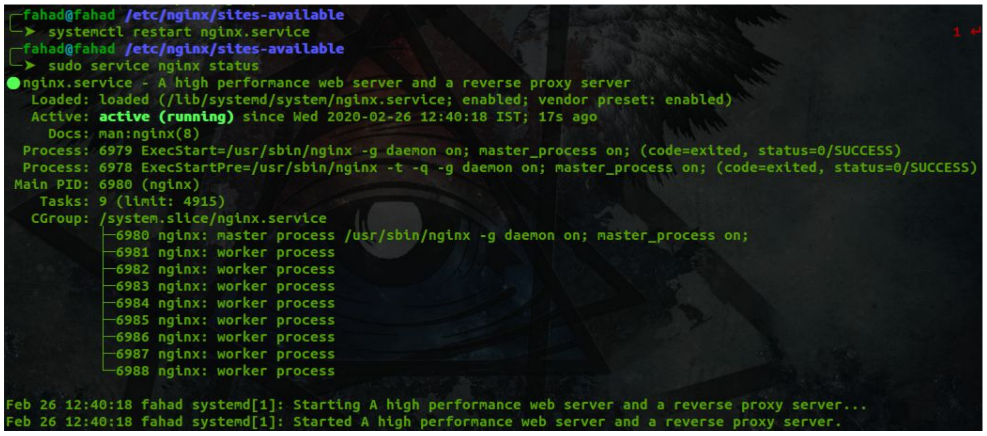
```
server{
        listen 80;
        server_name www.bootcamp.com;
        return 302 https://www.bootcamp.com;
}
server{
        listen 443 ssl;
        server_name www.bootcamp.com;
        ssl_certificate /etc/nginx/ssl/nginx.pem;
        ssl_certificate_key /etc/nginx/ssl/nginx.key;
location / {
                proxy_pass http://127.0.0.1:8080;
        }
}
```

Now we have to make entry of www.bootcamp.com in the **/etc/hosts**



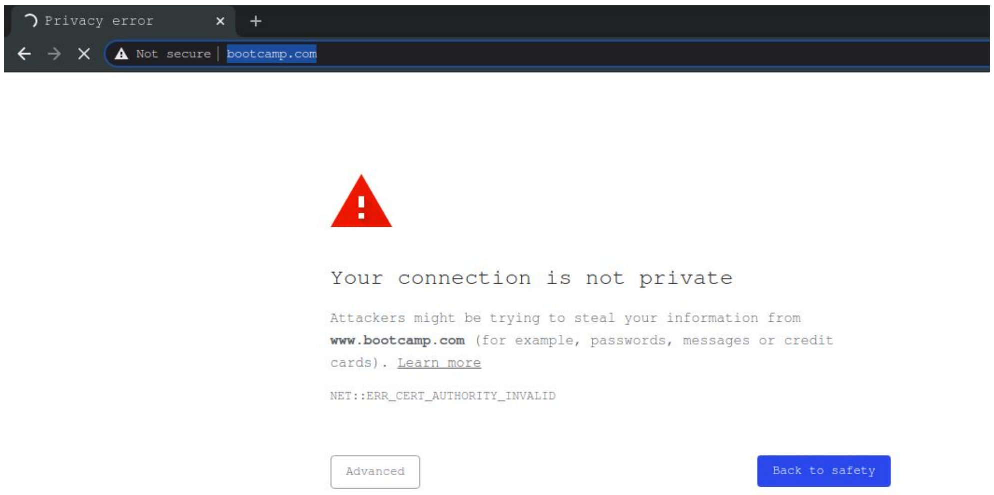
```
127.0.0.1       localhost www.abc.com xyz.com loadbalancing.com www.bootcamp.com
127.0.1.1       fahad
10.1.211.14     abc.com
# The following lines are desirable for IPv6 capable hosts

::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Now we have to restart the nginx



```
┌─fahad@fahad /etc/nginx/sites-available
└─➤ systemctl restart nginx.service                                                    1 ↵
┌─fahad@fahad /etc/nginx/sites-available
└─➤ sudo service nginx status
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-02-26 12:40:18 IST; 17s ago
     Docs: man:nginx(8)
  Process: 6979 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 6978 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 6980 (nginx)
    Tasks: 9 (limit: 4915)
   CGroup: /system.slice/nginx.service
           ├─6980 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           ├─6981 nginx: worker process
           ├─6982 nginx: worker process
           ├─6983 nginx: worker process
           ├─6984 nginx: worker process
           ├─6985 nginx: worker process
           ├─6986 nginx: worker process
           ├─6987 nginx: worker process
           └─6988 nginx: worker process

Feb 26 12:40:18 fahad systemd[1]: Starting A high performance web server and a reverse proxy server...
Feb 26 12:40:18 fahad systemd[1]: Started A high performance web server and a reverse proxy server.
```

Then on going to web-browser and typing http://localhost:8082 url it will show the result

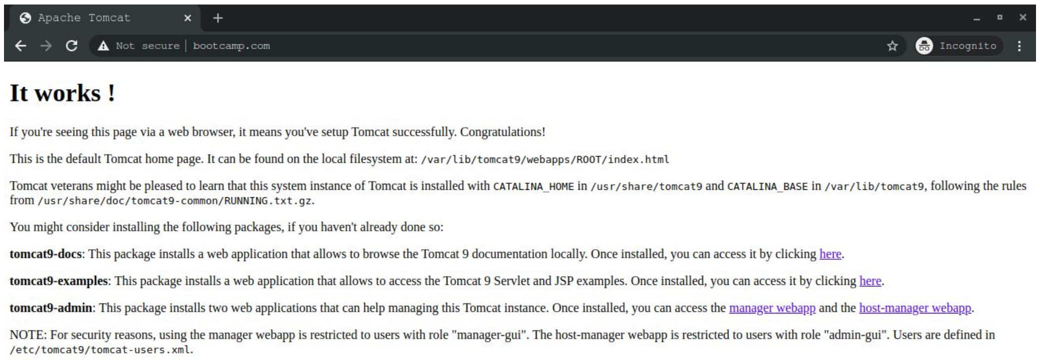


Your connection is not private

Attackers might be trying to steal your information from **www.bootcamp.com** (for example, passwords, messages or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

In above image it is asking for the security now we just have to click on advance then allow it and it will redirect us to the give below image



It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat9/webapps/ROOT/index.html`

Tomcat veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat9` and `CATALINA_BASE` in `/var/lib/tomcat9`, following the rules from `/usr/share/doc/tomcat9-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

tomcat9-docs: This package installs a web application that allows to browse the Tomcat 9 documentation locally. Once installed, you can access it by clicking here.

tomcat9-examples: This package installs a web application that allows to access the Tomcat 9 Servlet and JSP examples. Once installed, you can access it by clicking here.

tomcat9-admin: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the manager webapp and the host-manager webapp.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat9/tomcat-users.xml`.

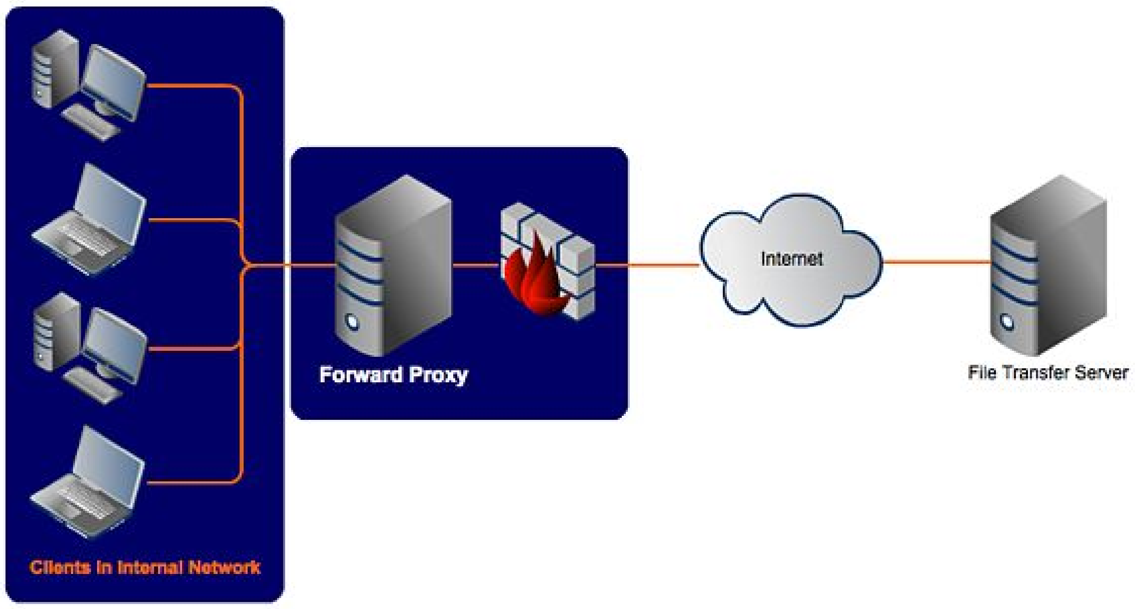
Ques 6.2 What is the difference between proxy_pass & proxy_pass reverse?

Ans 6.2

Proxy_pass

When people talk about a proxy server (often simply known as a "proxy"), more often than not they are referring to a forward proxy. Let me explain what this particular server does.

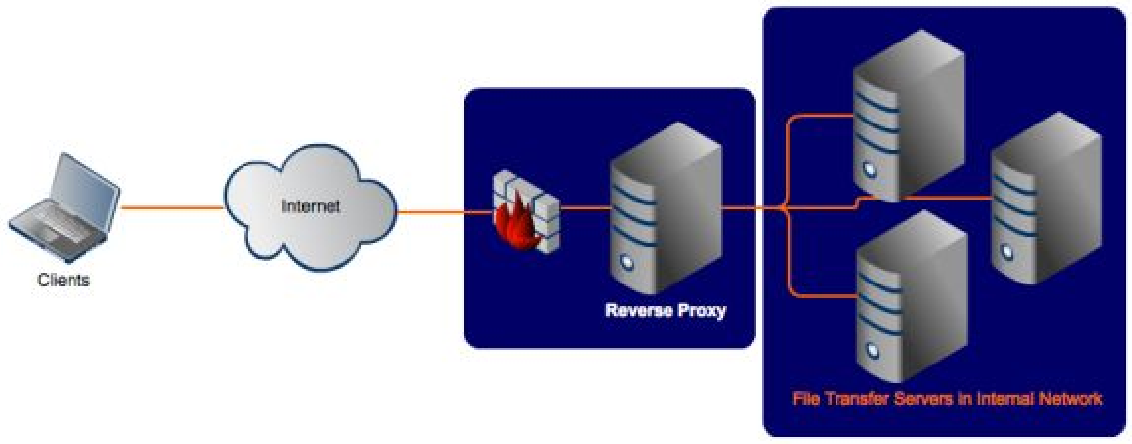
A forward proxy provides proxy services to a client or a group of clients. Oftentimes, these clients belong to a common internal network like the one shown below.



When one of these clients makes a connection attempt to that file transfer server on the Internet, its requests have to pass through the forward proxy first.

The Reverse Proxy

What is a reverse proxy? As its name implies, a reverse proxy does the exact opposite of what a forward proxy does. While a forward proxy proxies in behalf of clients (or requesting hosts), a reverse proxy proxies in behalf of servers. A reverse proxy accepts requests from external clients on behalf of servers stationed behind it just like what the figure below illustrates.



To the client in our example, it is the reverse proxy that is providing file transfer services. The client is oblivious to the file transfer servers behind the proxy, which are actually providing those services. In effect, whereas a forward proxy hides the identities of clients, a reverse proxy hides the identities of servers.