

APPLICATION SERVER (TOMCAT)



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

	Application Server	Web Server
What is it?	A server that exposes business logic to client applications through various protocols including HTTP.	A server that handles HTTP protocol.
Job	Application server is used to serve web based applications and enterprise based applications(i.e servlets, jsps and ejbs...). Application servers may contain a web server internally.	Web server is used to serve web based applications.(i.e servlets and jsps)
Functions	To deliver various applications to another device, it allows everyone in the network to run software off of the same machine.	Keeping HTML, PHP, <u>ASP</u> , etc files available for the web browsers to view when a user accesses the site on the web.
Supports	distributed transaction and EJB's	Servlets and JSP
Resource utilization	High	Low

2. What is Catalina?

ANS: 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.

3. Describe tomcat directory structure.

ANS: 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

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

ANS:

STEP 1: Create a new test user, a new database and a single test table. Your MySQL user **must** have a password assigned. The driver will fail if you try to connect with an empty password.

```
mysql> GRANT ALL PRIVILEGES ON *.* TO DIKSHA@localhost IDENTIFIED BY 'DIKSHA' WITH GRANT OPTION;
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> CREATE DATABASE Tomcat;
Query OK, 1 row affected (0.00 sec)

mysql> USE Tomcat;
Database changed
```

```
mysql> CREATE TABLE Tomcatdb ( ID int not null auto_increment primary key,foo varchar(25),bar int);
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> DESC Tomcatdb;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| ID    | int(11)       | NO   | PRI | NULL    | auto_increment |
| foo   | varchar(25)   | YES  |     | NULL    |                |
| bar   | int(11)       | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

STEP 2: Next insert some test data into the testdata table.

```
mysql> INSERT INTO Tomcatdb VALUES (NULL, 'HELLO WORLD' ,14245132);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Tomcatdb;
+-----+-----+-----+
| ID | foo          | bar          |
+-----+-----+-----+
|  1 | HELLO WORLD | 14245132    |
+-----+-----+-----+
1 row in set (0.00 sec)
```

STEP 3:Context configuration

Configure the JNDI DataSource in Tomcat by adding a declaration for your resource to your [Context](#).

```

<Context>

  <!-- Default set of monitored resources. If one of these changes, the      -->
  <!-- web application will be reloaded.                                     -->
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
  <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
  <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

  <!-- Uncomment this to disable session persistence across Tomcat restarts -->
  <!--
  <Manager pathname="" />
  -->

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

</Context>
~
~
~
"context.xml" 38L, 1687C written                                     36,54      Bot

```

```

root@diksha:/var/lib/tomcat9/webapps# ls
ROOT sample sample.war
root@diksha:/var/lib/tomcat9/webapps# mkdir DB_page
root@diksha:/var/lib/tomcat9/webapps# cd DB_page/
root@diksha:/var/lib/tomcat9/webapps/DB_page# mkdir WEB-INF
root@diksha:/var/lib/tomcat9/webapps/DB_page# vim WEB-INF/web.xml
root@diksha:/var/lib/tomcat9/webapps/DB_page#

```

STEP 4: web.xml configuration

Now create a WEB-INF/web.xml for this test application.


```

<%@ 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 Tomcatdb
</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>
~
"test.jsp" [New] 22L, 430C written

```

```

root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF# mkdir lib
root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF# cd lib/
root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF/lib# cp /home/diksha/Downloads/standard.jar .
root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF/lib# cp /home/diksha/Downloads/jstl.jar .
root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF/lib# ll
total 436
drwxr-xr-x 2 root root  4096 Feb 24 18:02 ./
drwxr-xr-x 3 root root  4096 Feb 24 18:01 ../
-rw-r--r-- 1 root root 23773 Feb 24 18:02 jstl.jar
-rw-r--r-- 1 root root 413170 Feb 24 18:02 standard.jar
root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF/lib# cd ..
root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF# ll
total 16
drwxr-xr-x 3 root root 4096 Feb 24 18:01 ./
drwxr-xr-x 3 root root 4096 Feb 24 17:57 ../
drwxr-xr-x 2 root root 4096 Feb 24 18:02 lib/
-rw-r--r-- 1 root root  500 Feb 24 17:54 web.xml
root@diksha:/var/lib/tomcat9/webapps/DB_page/WEB-INF# cd ..
root@diksha:/var/lib/tomcat9/webapps/DB_page# ll
total 16
drwxr-xr-x 3 root root 4096 Feb 24 17:57 ./
drwxrwxr-x 5 tomcat tomcat 4096 Feb 24 17:52 ../
-rw-r--r-- 1 root root  430 Feb 24 17:57 test.jsp

```

```

root@diksha:/var/lib/tomcat9/webapps/DB_page# cd ..
root@diksha:/var/lib/tomcat9/webapps# ll
total 28
drwxrwxr-x 5 tomcat tomcat 4096 Feb 24 17:52 ./
drwxr-xr-x 5 root root 4096 Feb 24 17:34 ../
drwxr-xr-x 3 root root 4096 Feb 24 17:57 DB_page/
drwxr-xr-x 3 root root 4096 Feb 23 21:03 ROOT/
drwxr-x--- 5 tomcat tomcat 4096 Feb 24 17:23 sample/
-rw-r--r-- 1 root root 4606 Feb 24 17:23 sample.war

```

```

diksha@diksha:~/Downloads$ sudo dpkg -i mysql-connector-java_8.0.19-1ubuntu18.04_all.deb
[sudo] password for diksha:
(Reading database ... 171268 files and directories currently installed.)
Preparing to unpack mysql-connector-java_8.0.19-1ubuntu18.04_all.deb ...
Unpacking mysql-connector-java (8.0.19-1ubuntu18.04) over (8.0.19-1ubuntu18.04)
...
Setting up mysql-connector-java (8.0.19-1ubuntu18.04) ...

```

```

diksha@diksha:~/Downloads$ dpkg -x mysql-connector-java_8.0.19-1ubuntu18.04_all.deb .
diksha@diksha:~/Downloads$ ll
total 133048
drwxr-xr-x 4 diksha diksha 4096 Dec 4 17:47 ./
drwxr-xr-x 30 diksha diksha 4096 Feb 24 17:48 ../
-rw-rw-r-- 1 diksha diksha 1084361 Feb 12 13:12 'Advanced Linux.pdf'
-rw-rw-r-- 1 diksha diksha 942422 Feb 24 17:11 'ASSESSMENT 11_VPC.pdf'
-rw-rw-r-- 1 diksha diksha 1759502 Feb 12 15:18 'assessment1 .pdf'
-rw-rw-r-- 1 diksha diksha 1346547 Feb 17 15:00 'assessment5 Nginx.pdf'
-rw-rw-r-- 1 diksha diksha 2174167 Feb 18 16:12 'Assessment6_mongodb n mysql.'

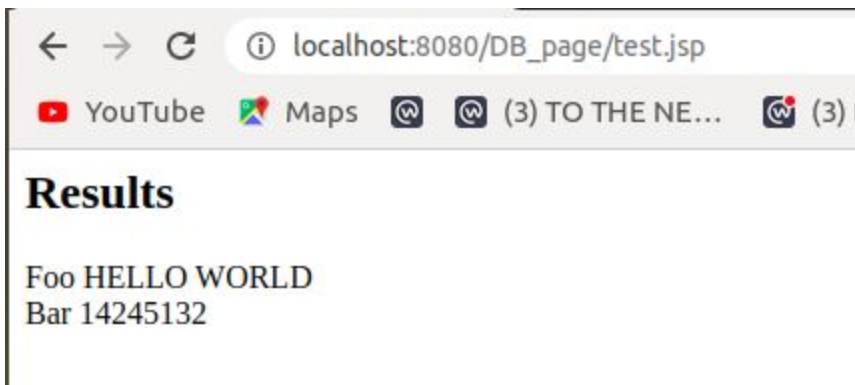
```



```





diksha@diksha:~/Downloads$ cd usr/share/
doc/ java/
diksha@diksha:~/Downloads$ cd usr/share/java/
diksha@diksha:~/Downloads/usr/share/java$ ll
total 2312
drwxr-xr-x 2 diksha diksha 4096 Dec 4 17:47 ./
drwxr-xr-x 4 diksha diksha 4096 Dec 4 17:47 ../
-rw-r--r-- 1 diksha diksha 2356694 Dec 4 17:47 mysql-connector-java-8.0.19.jar
diksha@diksha:~/Downloads/usr/share/java$ sudo cp mysql-connector-java-8.0.19.jar /var/lib/tomcat9/lib/.
diksha@diksha:~/Downloads/usr/share/java$ cd /var/lib/tomcat9/lib/.
diksha@diksha:/var/lib/tomcat9/lib$ ll
total 2312
drwxr-xr-x 2 tomcat tomcat 4096 Feb 24 18:05 ./
drwxr-xr-x 5 root root 4096 Feb 24 18:03 ../
-rw-r--r-- 1 root root 2356694 Feb 24 18:05 mysql-connector-java-8.0.19.jar
(reverse-i-search)`': 








```




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

```
<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="webapps"
      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="webapps"
      unpackWARs="true" autoDeploy="true">
    </Host>
  </Engine>
</Service>
```

    localhost:8082/sample/

 Apps  YouTube  Maps   (3) TO THE NE...  (3) Diksha To...  Learn

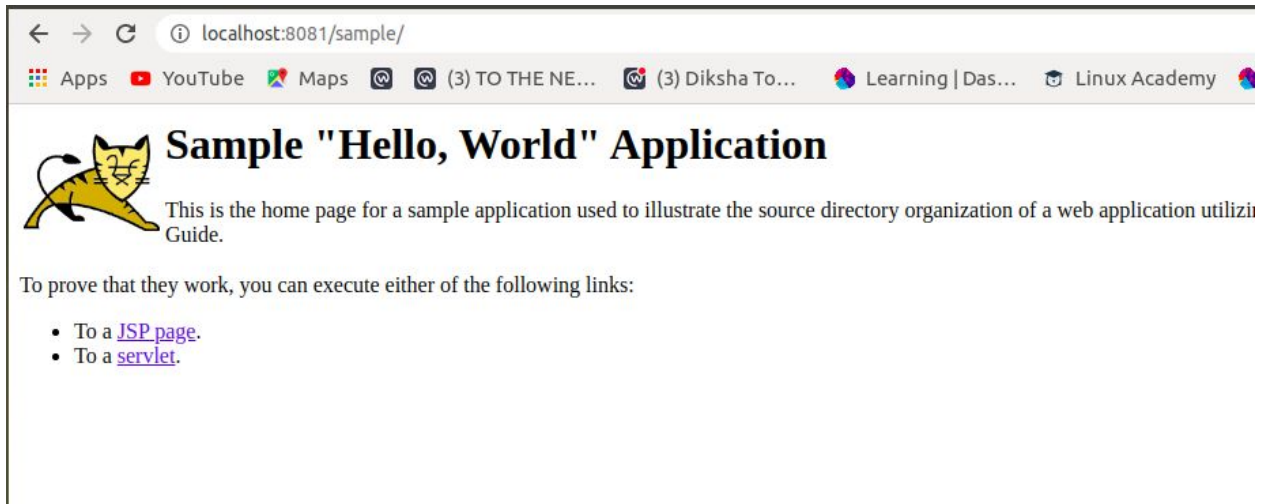


Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory of the Tomcat Guide.

To prove that they work, you can execute either of the following links:

- To a [JSP page](#).
- To a [servlet](#).

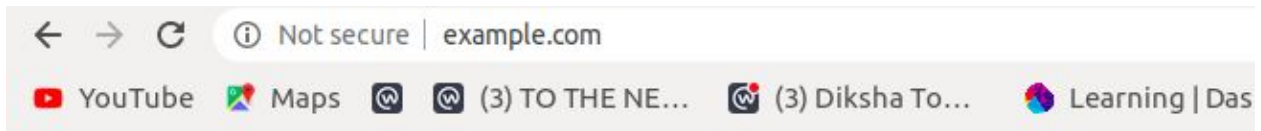


6. Use nginx as reverse proxy for tomcat application.

```
diksha@diksha:~$ keytool -genkey -alias tomcat -keyalg RSA
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Diksha Tomar
What is the name of your organizational unit?
  [Unknown]:  DevOps
What is the name of your organization?
  [Unknown]:  TTn
What is the name of your City or Locality?
  [Unknown]:  Noida
```

STEP 1: Make a new configuration file for it.

```
diksha@diksha: /etc/nginx/sites-enabled
File Edit View Search Terminal Help
server {
    listen 80;
    server_name example.com;
    location / {
        proxy_pass http://127.0.0.1:8080/sample/;
    }
}
```



Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory Guide.

To prove that they work, you can execute either of the following links:

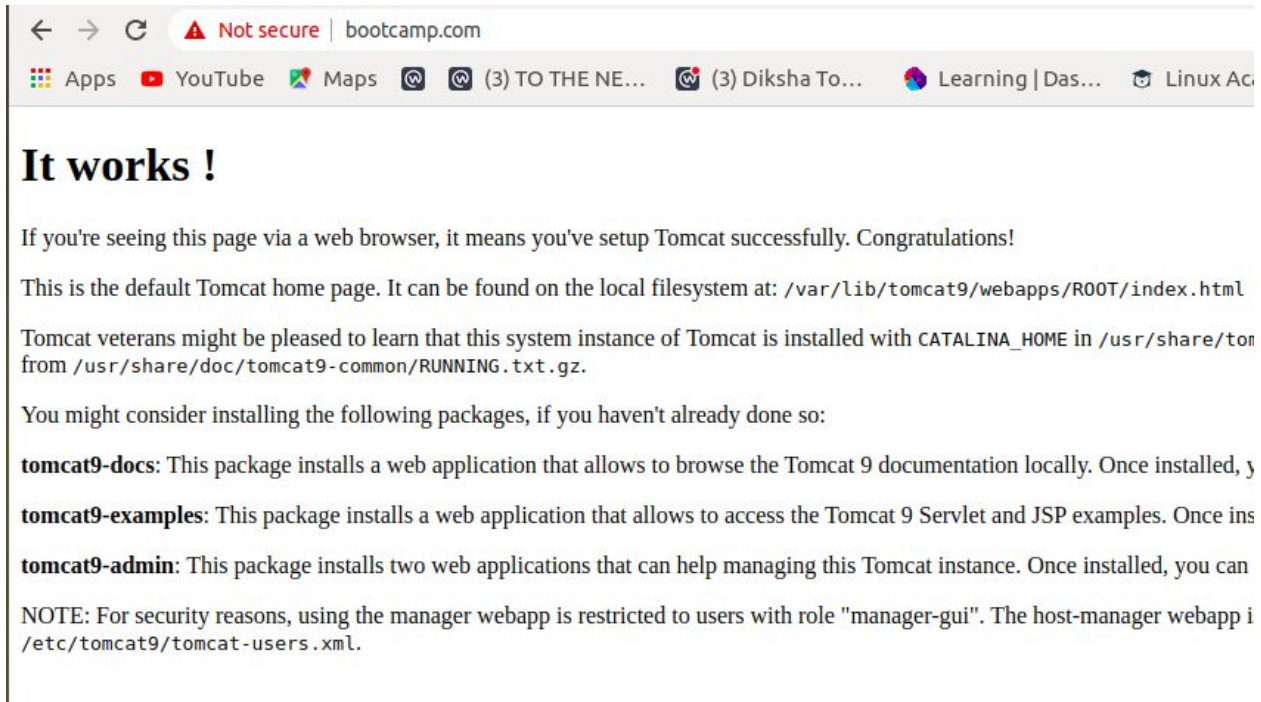
- To a [JSP page](#).
- To a [servlet](#).

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

STEP 1: Make new configuration file for it in nginx

```
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/public.pem;
    ssl_certificate_key /etc/nginx/ssl/private.key;
    location /{
        proxy_pass http://127.0.0.1:8080;
    }
}

"tomcatssl.config" [New] 15L, 338C written
```

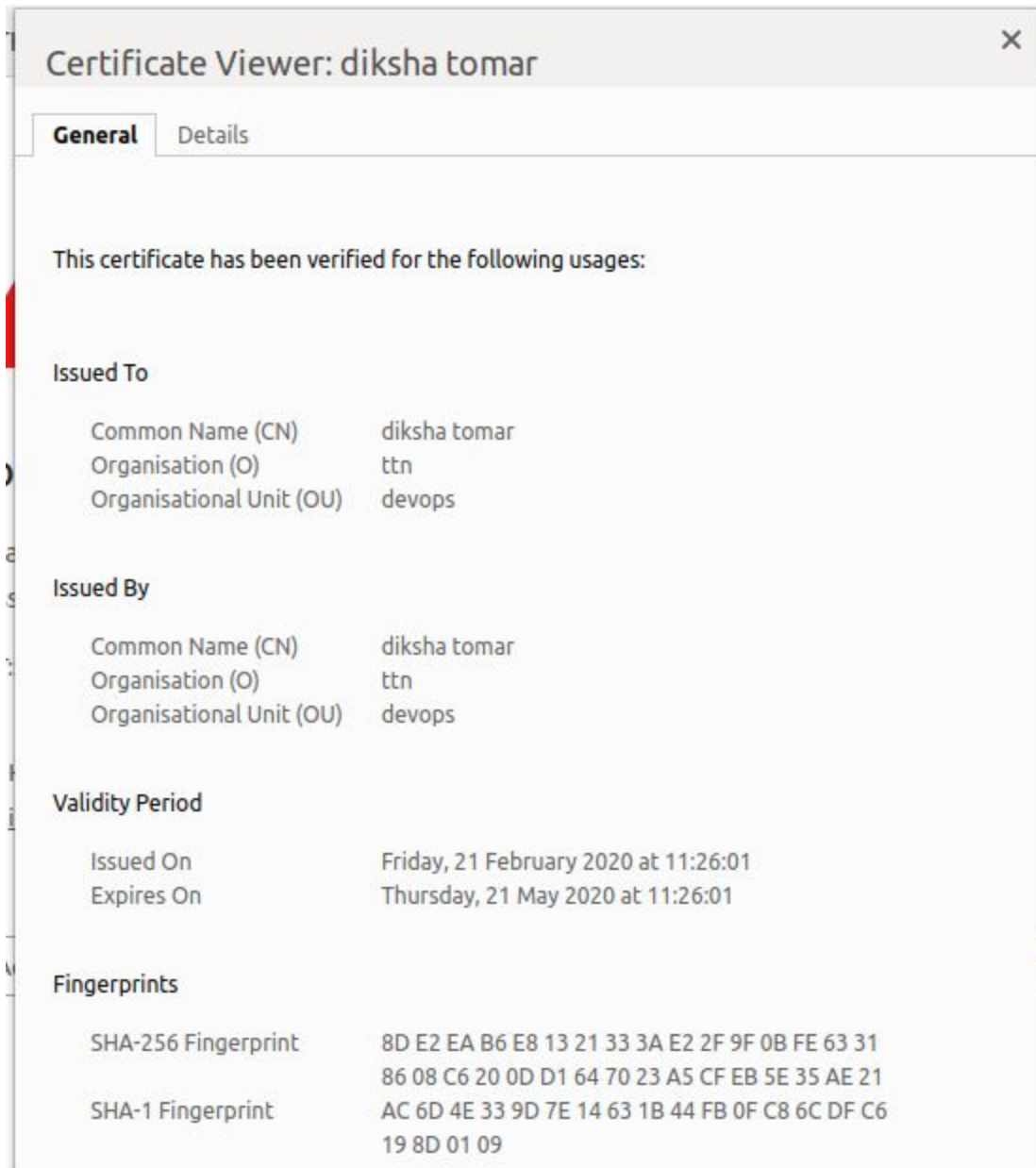
Applying ssl on tomcat

It will create a .keystore file on your user home directory. It will be on /home/[username].

STEP 2: Make entry in server.xml

```
<Connector SSLEnabled="true" acceptCount="100" clientAuth="false"
  disableUploadTimeout="true" enableLookups="false" maxThreads="25"
  port="8443" keystoreFile="/home/diksha/.keystore" keystorePass="diksha20"
  protocol="org.apache.coyote.http11.Http11NioProtocol" scheme="https"
  secure="true" sslProtocol="TLS" />
<!--
```

STEP 3: Start tomcat service and try to access <https://localhost:8443>.



8. What is the difference between proxy_pass & proxy_pass reverse?

ANS: A reverse proxy server is an intermediate connection point positioned at a network's edge. It receives initial HTTP connection requests, acting like the actual endpoint.

Essentially your network's traffic cop, the reverse proxy serves as a gateway between users and your application origin server. In so doing it handles all policy management and traffic routing.

A reverse proxy operates by:

1. Receiving a user connection request
2. Completing a TCP three-way handshake, terminating the initial connection
3. Connecting with the origin server and forwarding the original request

A forward proxy server is also positioned at your network's edge, but regulates outbound traffic according to preset policies in shared networks. Additionally, it disguises a client's IP address and blocks malicious incoming traffic.

Forward proxies are typically used internally by large organizations, such as universities and enterprises, to:

1. Block employees from visiting certain websites.
2. Monitor employee online activity
3. Block malicious traffic from reaching an origin server
4. Improve the user experience by caching external site content

