**Learning Tomcat**

**Starting a tomcat**

<go to your tomcat installation directory>/bin/<double click> startup.bat

Open Tomcat Manager App through GUI

Tomcat > conf > tomcat-users.xml >

<role rolename="manager-gui"/>

<user username="root" password="root" roles="manager-gui"/>

Go to <http://localhost:8080> -> click on manager app

Tomcat Security and SSL configuration

1. Enabling SSL

Go to C:\Program Files\Java\jdk-11.0.3\bin :: for java 11

Go to C:\Program Files\Java\jre-11.0.3\bin :: for java8

Open command prompt and type

**keytool -genkey -keyalg RSA -alias myKey -keystore E:\java-study-material\tomcat\demo-app-key.jks**

password -> 123456

1. Tomcat > conf > server.xml

Uncomment Connector port which specifies to port 8443

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true">

<SSLHostConfig>

<Certificate certificateKeystoreFile="E:\java-study-material\tomcat\demo-app-key.jks"

type="RSA" certificateKeystorePassword="123456"/>

</SSLHostConfig>

</Connector>

Redirect every http://localhost:8080 call to <https://localhost:8443>

Tomcat > webapps > manager > web-inf > web.xml

Add below configuration inside security-constraints

<security-constraints>

<user-data-constraint>

<transport-guarantee>CONFIDENTIAL</transport-guarantee>

</user-data-constraint>

</security-constraints>

**Creating Virtual Host**

Physical Host – has dedicated memory

Virtual Host – Share memory and cache. Multiple domains get hosted in a single Tomcat Server i.e each of the domains has the same IP.

etc/hosts -> create 3 domains tagged to specific IP

127.0.0.1 [www.website1.com](http://www.website1.com)

127.0.0.1 www.website2.com

127.0.0.1 [www.website3.com](http://www.website3.com)

Create 3 folders webapp1/webapp2/webapp3 in tomcat folder where webapps lies.

Create a folder ROOT inside each of the webapp\* folder. And place the extracted war file contents inside each of the ROOT folder

Go to tomcat > conf > server.xml. Add three Host tag for 3 different domains

<Host name="www.website1.com" appBase="webapp1" unpackWARs="true" autoDeploy="true"/>

<Host name="www.website2.com" appBase="webapp2" unpackWARs="true" autoDeploy="true"/>

<Host name="www.website3.com" appBase="webapp3" unpackWARs="true" autoDeploy="true"/>

**Clustering**

Vertical Clustering

Horizontal Clustering

**Creating vertical Cluster**

Create 3 instance folder instance1/instance2/instance3

Copy all the folders from tomcat except bin and lib to each of the instances. bin and lib will be shared by each of the instances.

Change 3 ports shutdown port/ AJP port/ connector 8080 port for each of the instances

Instance1 > conf > server.xml 🡪 change the above 3 ports to avoid conflicts.

Create a startup/shutdown script for each of the instances.

set CATALINA\_HOME=E:\java-study-material\tomcat\apache-tomcat-8.5.38

set CATALINA\_BASE=E:\java-study-material\tomcat\instance3

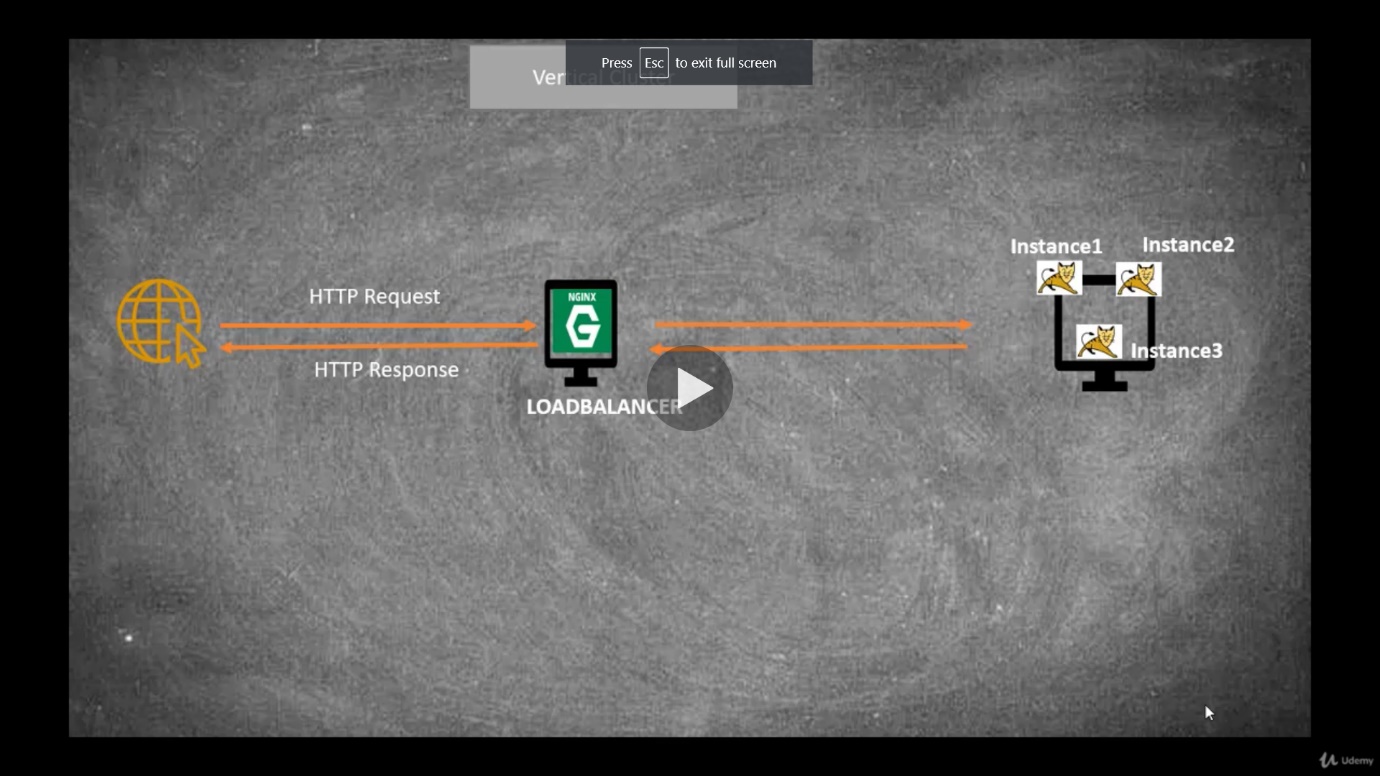
cd %CATALINA\_HOME%\bin

startup.bat

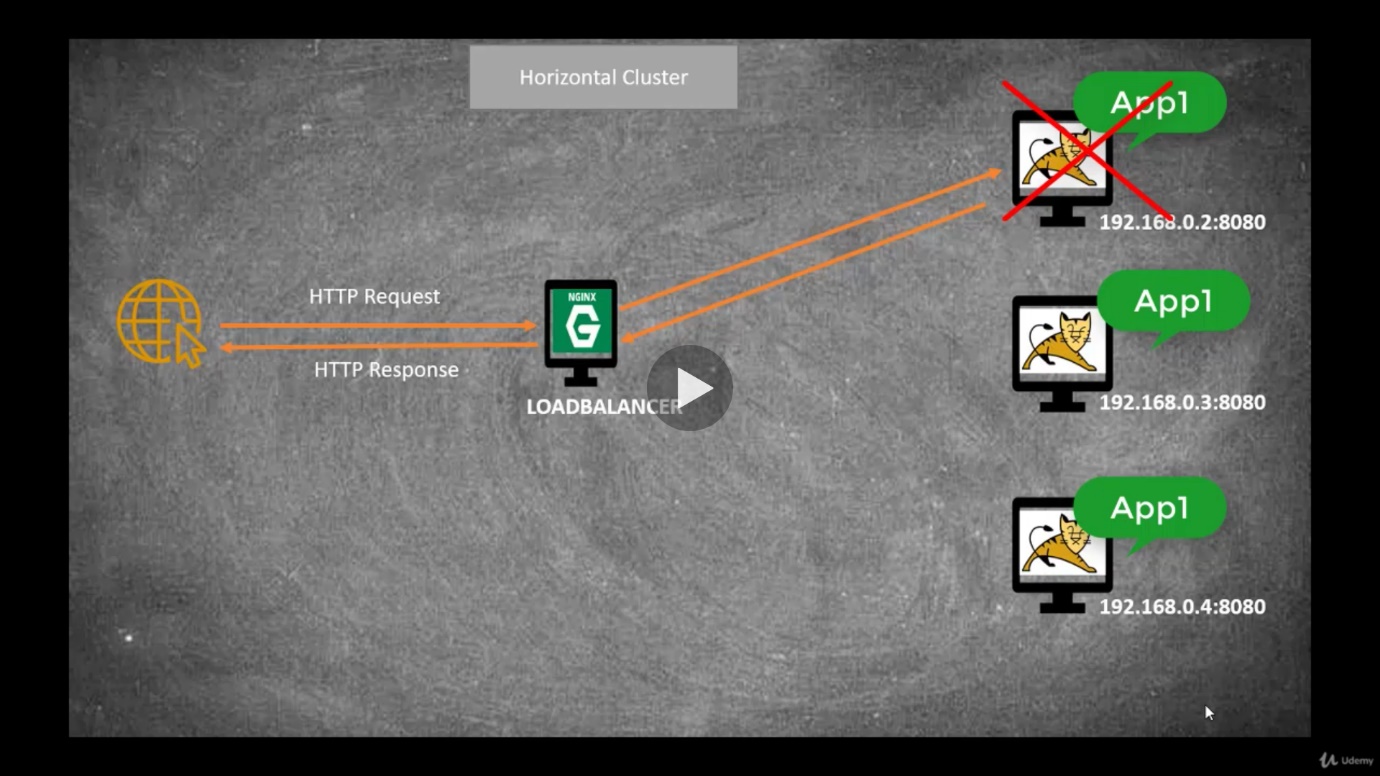
For shutdown scripts, just change startup.bat to shutdown.bat

Now open the startup script for each of the instances

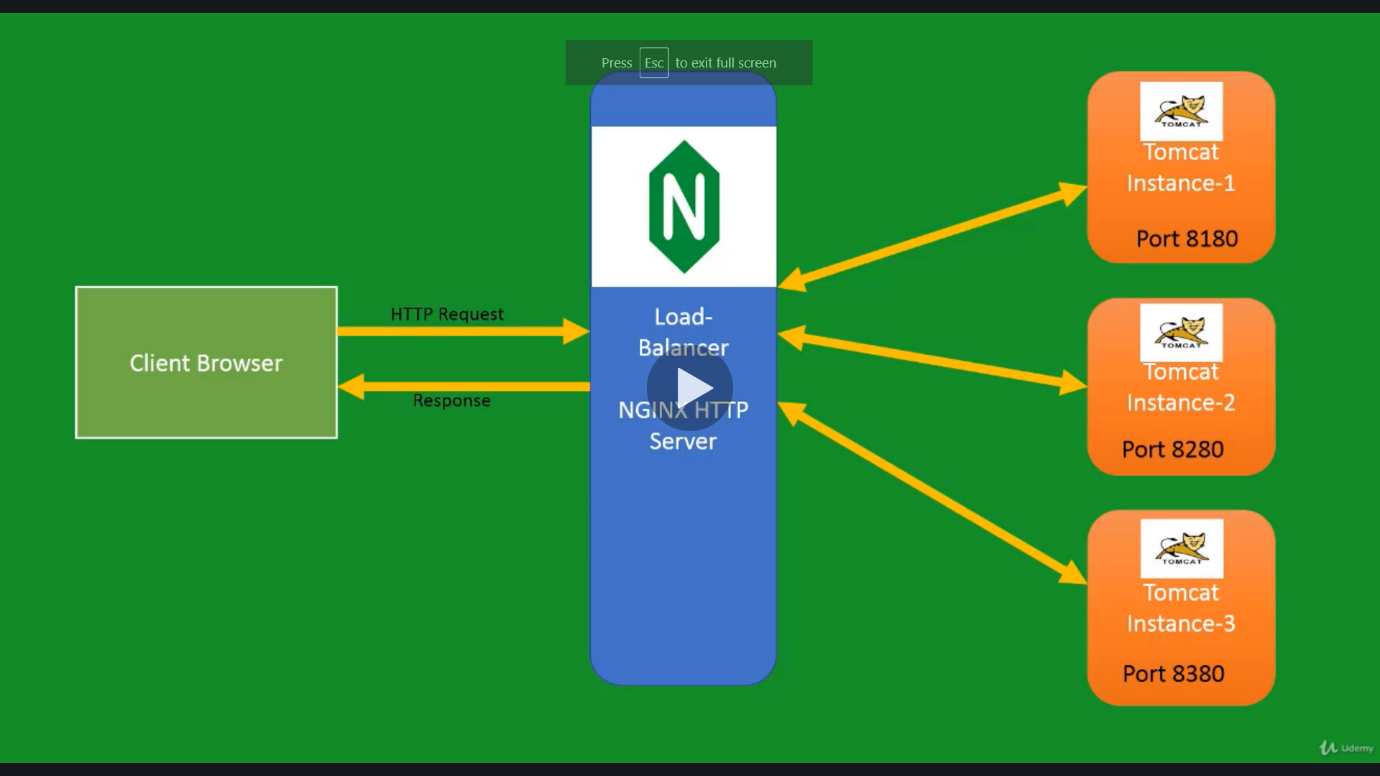
**Vertical**



**Horizontal Cluster**



**Load Balancing using Nginx (Software load balancer)**



Two types of load balancers – s/w and h/w load balancers

Download Nginx

<http://nginx.org/en/download.html>

Go to Nginx > conf > nginx.conf > http block

upstream tomcat-servers {

server 127.0.0.1:8180;

server 127.0.0.1:8280;

server 127.0.0.1:8380;

}

location / {

**proxy\_pass http://tomcat-servers;**

#root html;

#index index.html index.htm;

}

Now first start the Nginx using

start nginx [For stopping -> nginx -s quit]

Then start all the individual instances and hit the URL <http://localhost>

**Session Management**

**Sticky Session**

To apply sticky session

Just add ip\_hash in server block where we add the list of server IPs

**Session Replication :**

HttpSession session = request.getSession();

This is provided by the one of the session manager. There are 4 types of session manager

Standard Manager -> Whole Object is replicated.Not recommended

PersistenceManager -> It saves the session data in flat file/databases. Not recommended.

DeltaManager -> Instead of replicating the whole object it replicates only the updated changes. Same as StandardManager

BackupManager -> Same as Delta Manager. But instead of replicating updated changes to all the nodes, it replicates the updated data to some (1 or 2) nodes.

Nodes Terminologies

Unicast -> One node sending info to one node

Broadcast -> one to many

Multicast -> many to many

Creating JNDI

This is used to get the database connections from a a key,value resource.

Tomcat > conf > context.xml

<Resource

name="jdbc/usersDB"

auth="Container"

type="javax.sql.DataSource"

maxActive="100"

maxIdle="30"

maxWait="10000"

driverClassName="com.mysql.cj.jdbc.Driver"

url="jdbc:mysql://localhost:3306/usersDB"

username="root"

password="root"

/>

Go to applicatrion -> web.xml

<resource-ref>

<description>DB Connection</description>

<res-ref-name>jdbc/usersDB</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

Make the connection in servlet class as

Context initContext = new InitialContext();

Context envContext = (Context) initContext.lookup("java:comp/env");

DataSource ds = (DataSource) envContext.lookup("jdbc/UsersDB");

Connection conn = ds.getConnection();

Display data in jsp file as

<sql:query var="listUsers" dataSource="jdbc/UsersDB">

select username, email from users;

</sql:query>