USAID
FROM THE AMERICAN PEOPLE

MOBILE APPLICATION TO SECURE TENURE (MAST)

MAST Deployment Environment Set-up Guide (Draft v.2.1)

Abstract

This document provides the details for the download and installation of various software/tools used to deploy MAST web infrastructure in production environment.

Contents

1 INTRODUCTION 2
2 INSTALL AND CONFIGURE MAST2 DEPLOYMENT ENVIRONMENT 2
2.1 Upgrade Ubuntu Packages 2
2.2 Install Java JDK 2
2.3 Install PostgreSQL 3
2.4 Install PostGIS 3
2.5 Restore database 3
2.6 Install ApacheTomcat 7 4
2.7 Configure ApacheTomcat 7 4
2.8 Deploying Geoserver and WAR files 10
2.9 Testing Deployment 10

1 INTRODUCTION

This guide describes the setup of development environment of MAST2 web infrastructure in Linux based Ubuntu 14.04 LTS operating system.

The objective of this document is to provide the details of environment setup including download and installation of various software/tools that can be used to deploy MAST2 web infrastructure in production environment.

2 INSTALL AND CONFIGURE MAST2 DEPLOYMENT ENVIRONMENT

All the steps mentioned in the Deployment Guide will either use Putty-SSH client to get terminal access for installation or WinSCP to copy data from local computer to Linux server.

Descriptions provided in this guide assume that you are using a single server as Database and Application server. In case of different server for application and database, please follow the relevant instructions per server.

Before installing deployment environment user need to upgrade Ubuntu packages.

2.1 Upgrade Ubuntu Packages

To upgrade Ubuntu packages type following on terminal prompt and press enter:

- $Sudo apt-get update
- $Sudo apt-get upgrade

2.2 Install Java JDK

To install java using package add oracle java repository using below mentioned command on the terminal:

- $sudo add-apt-repository ppa:webupd8team/java
- $sudo apt-get update

To install java run command as mentioned below:

- $sudo apt-get install oracle-java7-installer

This command will configure the java_home path along with installations.

2.3 Install PostgreSQL

PostgreSQL is available in Ubuntu repository by default. Run the below mentioned commands on terminal to install PostgreSQL database:

$apt-get install postgresql postgresql-contrib

Use following command to change default password of Postgres user:

$sudo -u postgres psql template1

Above prompt would move to template prompt. In template prompt type

templete# \password postgres

Enter and confirm the new password that would be used as admin password. Note down the password as this would be used during postGIS installation and database access.

To exit Postgres template promt press ctlr+D

To start/Stop PostgreSQL service use command:

$sudo /sbin/service postgresql stop/start

2.4 Install PostGIS

Use the below command in Ubuntu terminal to install PostGIS.

$sudo apt-get install postgresql-9.3-postgis

Creating a PostGIS database template is the way to go if you want to make it easy the creations of many GIS database on the same server. Without creating this template, you would need to repeat this steps every time you need to create a PostGIS database.

$sudo su postgres

$createdb template_postgis

$createlang plpgsql template_postgis

$psql -d template_postgis -f /usr/share/postgresql-9.3-postgis/lwpostgis.sql

$psql -d template_postgis -f /usr/share/postgresql-9.3-postgis/spatial_ref_sys.sql

2.5 Restore database

Download PostgreSQL database backup tar file from GitHub (https://github.com/MASTUSAID/DB-SCRIPTS2) and copy into the database server.

Use the below command to restore the tar file to the database:

$pg_restore -c -i -U postgres -d MAST_Configurations -v "full_db.tar" -W

2.6 Install ApacheTomcat 7

Download the Apache Tomcat 7 64 binary file for Linux environment from:

https://tomcat.apache.org/download-70.cgi

Copy and extract the tar file in the Ubuntu server. Tar file contains all the dependencies required, no there is no need to run any installations.

2.7 Configure ApacheTomcat 7

Memory configurations and Garbage collector settings

Memory configurations can be specified in setenv.sh file under Tomcat/bin folder.

Following configuration values are used in production server:

-Duser.timezone=UTC

-server -d64

-XX:NewSize=700m

-XX:MaxNewSize=700m

-Xms1024m

-Xmx2048m

-XX:MaxPermSize=1024m

-XX:+UseParNewGC

-XX:+UseConcMarkSweepGC

-XX:+CMSParallelRemarkEnabled

-XX:SurvivorRatio=20

-XX:ParallelGCThreads=8

Copy the following files in Tomcat/Lib folders

postgresql-9.3-1103-jdbc4.jar

postgis-jdbc-2.0.0.jar

Configuring Context .xml and Server.xml

Server.xml and context.xml are available under Tomcat/conf folder

Context.xml: Add the line marked in RED font to context.xml file (Blue colored font describes the content of default context.xml file).

```
<?xml version='1.0' encoding='utf-8'?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<!-- The contents of this file will be loaded for each web application -->
<Context>

    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>

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

    <!-- Uncomment this to enable Comet connection tacking (provides events
         on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->
    <ResourceLink global="jdbc/dbname" name="jdbc/dbname"
type="javax.sql.DataSource"/>
</Context>
```

Server.xml: Edit the server.xml file. Add the lines as shown in red font in the server.xml file. Change the database/user and password of the Postgres database to be configured with Spatialvue (as provided during Postgres installation).

Please note that these changes are to be done only in the lines (shown in red font below) that are to be added in server.xml.

```
<?xml version='1.0' encoding='utf-8'?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<!-- Note:  A "Server" is not itself a "Container", so you may not
     define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/server.html
 -->
<Server port="8005" shutdown="SHUTDOWN">
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on"
/>
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-
howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"
/>
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener"
/>

  <!-- Global JNDI resources
       Documentation at /docs/jndi-resources-howto.html
  -->
  <GlobalNamingResources>
    <!-- Editable user database that can also be used by
         UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
```

```
            type="org.apache.catalina.UserDatabase"
            description="User database that can be updated and saved"
            factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
            pathname="conf/tomcat-users.xml" />
    <Resource type="javax.sql.DataSource"
          name="jdbc/dbname"
          factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
          driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://host:5432/dbname"
          username="username"
          password="password"
                    testWhileIdle="true"
           testOnBorrow="true"
           testOnReturn="false"
           validationQuery="SELECT 1"
           validationInterval="30000"
           timeBetweenEvictionRunsMillis="5000"
           maxActive="100"
           minIdle="10"
           maxWait="10000"
           initialSize="10"
           removeAbandonedTimeout="60"
           removeAbandoned="true"
           logAbandoned="true"
           minEvictableIdleTimeMillis="30000"
           jmxEnabled="true"

jdbcInterceptors="org.apache.tomcat.jdbc.pool.interceptor.ConnectionState;
                  org.apache.tomcat.jdbc.pool.interceptor.StatementFinalizer;

org.apache.tomcat.jdbc.pool.interceptor.SlowQueryReportJmx(threshold=10000)"
          />
  </GlobalNamingResources>

  <!-- A "Service" is a collection of one or more "Connectors" that share
       a single "Container" Note:  A "Service" is not itself a "Container",
       so you may not define subcomponents such as "Valves" at this level.
       Documentation at /docs/config/service.html
   -->
  <Service name="Catalina">

    <!--The connectors can use a shared executor, you can define one or more named
thread pools-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
```

```
-->


<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
<!-- Define a SSL HTTP/1.1 Connector on port 8443
     This connector uses the JSSE configuration, when using APR, the
     connector should be using the OpenSSL style configuration
     described in the APR documentation -->
<!--
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS" />
-->

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />


<!-- An Engine represents the entry point (within Catalina) that processes
     every request.  The Engine implementation for Tomcat stand alone
     analyzes the HTTP headers included with the request, and passes them
     on to the appropriate Host (virtual host).
     Documentation at /docs/config/engine.html -->

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
-->
<Engine name="Catalina" defaultHost="localhost">

  <!--For clustering, please take a look at documentation at:
```

```
    /docs/cluster-howto.html  (simple how to)
    /docs/config/cluster.html (reference documentation) -->
<!--
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
-->

<!-- Use the LockOutRealm to prevent attempts to guess user passwords
     via a brute-force attack -->
<Realm className="org.apache.catalina.realm.LockOutRealm">
  <!-- This Realm uses the UserDatabase configured in the global JNDI
       resources under the key "UserDatabase".  Any edits
       that are performed against this UserDatabase are immediately
       available for use by the Realm.  -->
  <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
         resourceName="UserDatabase"/>
</Realm>

<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">

  <!-- SingleSignOn valve, share authentication between web applications
       Documentation at: /docs/config/valve.html -->
  <!--
  <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
  -->

  <!-- Access log processes all example.
       Documentation at: /docs/config/valve.html
       Note: The pattern used is equivalent to using pattern="common" -->
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
         prefix="localhost_access_log." suffix=".txt"
         pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" />

        </Host>
      </Engine>
    </Service>
  </Server>
```

2.8 Deploying Geoserver and WAR files

Copy the geoserver.war and mast.war files to Tomcat/Webapps folder.

From Ubuntu terminal change folder to Tomcat/Bin and type following command to start and stop Tomcat service:

To start service

$sudo ./startup.sh

To stop service

$sudo ./shutdown.sh

When the service starts tomcat would automatically extract the war file in webapp folder.

2.9 Testing Deployment

To check whether Geoserver and MAST2 web infrastructure have been deployed correctly user the below mentioned URL:

http://IP(ubuntu server):80/geoserver/web/

http://IP(ubuntu server):80/mast/