**Deciding between running standalone servers or a managed domain**

Which use cases are appropriate for managed domain and which are appropriate for standalone servers? A managed domain is all about coordinated multi-server management -- with it JBoss AS 7 provides a central point through which users can manage multiple servers, with rich capabilities to keep those servers' configurations consistent and the ability to roll out configuration changes (including deployments) to the servers in a coordinated fashion.

It's important to understand that the choice between a managed domain and standalone servers is all about how your servers are managed, not what capabilities they have to service end user requests. This distinction is particularly important when it comes to high availability clusters. It's important to understand that HA functionality is orthogonal to running standalone servers or a managed domain. That is, a group of standalone servers can be configured to form an HA cluster. The domain and standalone modes determine how the servers are managed, not what capabilities they provide.

So, given all that:

* A single server installation gains nothing from running in a managed domain, so running a standalone server is a better choice.
* For multi-server production environments, the choice of running a managed domain versus standalone servers comes down to whether the user wants to use the centralized management capabilities a managed domain provides. Some enterprises have developed their own sophisticated multi-server management capabilities and are comfortable coordinating changes across a number of independent JBoss AS 7 instances. For these enterprises, a multi-server architecture comprised of individual standalone servers is a good option.
* Running a standalone server is better suited for most development scenarios. Any individual server configuration that can be achieved in a managed domain can also be achieved in a standalone server, so even if the application being developed will eventually run in production on a managed domain installation, much (probably most) development can be done using a standalone server.
* Running a managed domain mode can be helpful in some advanced development scenarios; i.e. those involving interaction between multiple JBoss AS 7 instances. Developers may find that setting up various servers as members of a domain is an efficient way to launch a multi-server cluster.

#### <ldap />

|  |
| --- |
| <authentication>    <ldap connection="..." base-dn="..." recursive="..." user-dn="...">      <username-filter attribute="..." />      <advanced-filter filter="..." />    </ldap>  </authentication> |

The ldap element is used to define how LDAP searches will be used to authenticate a user, this works by first connecting to LDAP and performing a search using the supplied user name to identity the distinguished name of the user and then a subsequent connection is made to the server using the password supplied by the user - if this second connection is a success then authentication succeeds.

|  |  |
| --- | --- |
|  | Due to the verification approach used this configuration causes the authentication mechanisms selected for the protocol to cause the password to be sent from the client in plain text, the following Jira issue is to investigating proxying a Digest authentication with the LDAP server so no plain text password is needed [AS7-4195](https://issues.jboss.org/browse/AS7-4195). |

* **connection** (mandatory) - The name of the connection to use to connect to LDAP.
* **base-dn** (mandatory) - The distinguished name of the context to use to begin the search from.
* **recursive** (optional) - Should the filter be executed recursively? Defaults to false.
* **user-dn** (optional) - After the user has been found specifies which attribute to read for the users distinguished name, defaults to 'dn'.

Within the ldap element only one of <username-filter /> or <advanced-filter /> can be specified.

### LDAP Authentication

The following example demonstrates an example configuration making use of Active Directory to verify the users username and password.

<management>

  <security-realms>

    <security-realm name="ManagementRealm">

      <authentication>

        <ldap connection="EC2" base-dn="CN=Users,DC=darranl,DC=jboss,DC=org">

          <username-filter attribute="sAMAccountName" />

        </ldap>

      </authentication>

    </security-realm>

  </security-realms>

  <outbound-connections>

    <ldap name="EC2" url="ldap://127.0.0.1:9797" search-dn="CN=as7,CN=Users,DC=darranl,DC=jboss,DC=org" search-credential="password"/>

  </outbound-connections>

</management>

**Application Realm**

Now in order to use **LDAP** for Authentication, you can use the **LdapExtended** Login module, entering the values of the bindDN and bindCredential contained in **slapd.conf**. You need to specify as well which organization unit contains the users, through the **baseCtxDN** option and as well the organization which contains the roles through the **rolesCtxDN**. Additionally you need to specify the following properties:  
The **baseFilter** option is a search filter used to locate the context of the user to authenticate.   
The **roleFilter** is as well a search filter used to locate the roles associated with the authenticated user.   
The **searchScope** sets the search scope to one of the strings. ONELEVEL\_SCOPE searches directly under the named roles context.  
Finally the **allowEmptyPasswords**: It is a flag indicating if empty(length==0) passwords should be passed to the LDAP server.

Here's the configuration to be added as security-domain for a **JBoss AS 7** installation:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <security-domain name="LDAPAuth">      <authentication>        <login-module code="LdapExtended" flag="required">          <module-option name="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory"/>          <module-option name="java.naming.provider.url" value="ldap://localhost:389"/>          <module-option name="java.naming.security.authentication" value="simple"/>          <module-option name="bindDN" value="uid=admin,dc=acme,dc=com"/>          <module-option name="bindCredential" value="secret"/>          <module-option name="baseCtxDN" value="ou=People,dc=acme,dc=com"/>          <module-option name="baseFilter" value="(uid={0})"/>          <module-option name="rolesCtxDN" value="ou=Roles,dc=acme,dc=com"/>          <module-option name="roleFilter" value="(member={1})"/>          <module-option name="roleAttributeID" value="cn"/>          <module-option name="searchScope" value="ONELEVEL\_SCOPE"/>          <module-option name="allowEmptyPasswords" value="true"/>        </login-module>      </authentication>  </security-domain> |

If you are running a **JBoss 4/5/6** security domain, here's the corresponding configuration:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <application-policy name="LDAPAuth">      <authentication>        <login-module code="org.jboss.security.auth.spi.LdapExtLoginModule" flag="required" >           <module-option name="java.naming.factory.initial">com.sun.jndi.ldap.LdapCtxFactory</module-option>           <module-option name="java.naming.provider.url">ldap://localhost:389</module-option>           <module-option name="java.naming.security.authentication">simple</module-option>           <module-option name="bindDN">uid=admin,dc=acme,dc=com</module-option>           <module-option name="bindCredential">secret</module-option>           <module-option name="baseCtxDN">ou=People,dc=acme,dc=com</module-option>           <module-option name="baseFilter">(uid={0})</module-option>             <module-option name="rolesCtxDN">ou=Roles,dc=acme,dc=com</module-option>           <module-option name="roleFilter">(member={1})</module-option>           <module-option name="roleAttributeID">cn</module-option>           <module-option name="searchScope">ONELEVEL\_SCOPE</module-option>           <module-option name="allowEmptyPasswords">true</module-option>        </login-module>      </authentication>  </application-policy> |

Done with the application server configuration, last step will be enabling security at application level; supposing you are going to secure a web application, the first step will be declaring the protected resources in your web.xml file, which are allowed just to the **Manager** role:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee <a href="http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd</a>"      id="WebApp\_ID" version="3.0">        <security-constraint>          <web-resource-collection>              <web-resource-name>HtmlAuth</web-resource-name>              <description>application security constraints  </description>              <url-pattern>/\*</url-pattern>              <http-method>GET</http-method>              <http-method>POST</http-method>          </web-resource-collection>          <auth-constraint>              <role-name>Manager</role-name>          </auth-constraint>      </security-constraint>      <login-config>          <auth-method>BASIC</auth-method>          <realm-name>LDAPAuth realm</realm-name>      </login-config>      <security-role>          <role-name>Manager</role-name>      </security-role>  </web-app> |

And here’s the **jboss-web.xml** configuration file which links the application to the LDAPAuth security domain. This file needs to be placed into the **WEB-INF** folder of your web application:

|  |  |
| --- | --- |
| 1  2  3 | <jboss-web>      <security-domain>java:/jaas/LDAPAuth</security-domain>  </jboss-web> |