Administering Apache Geronimo 2.x

David Jencks

ApacheCon

Leading the Wave
of Open Source

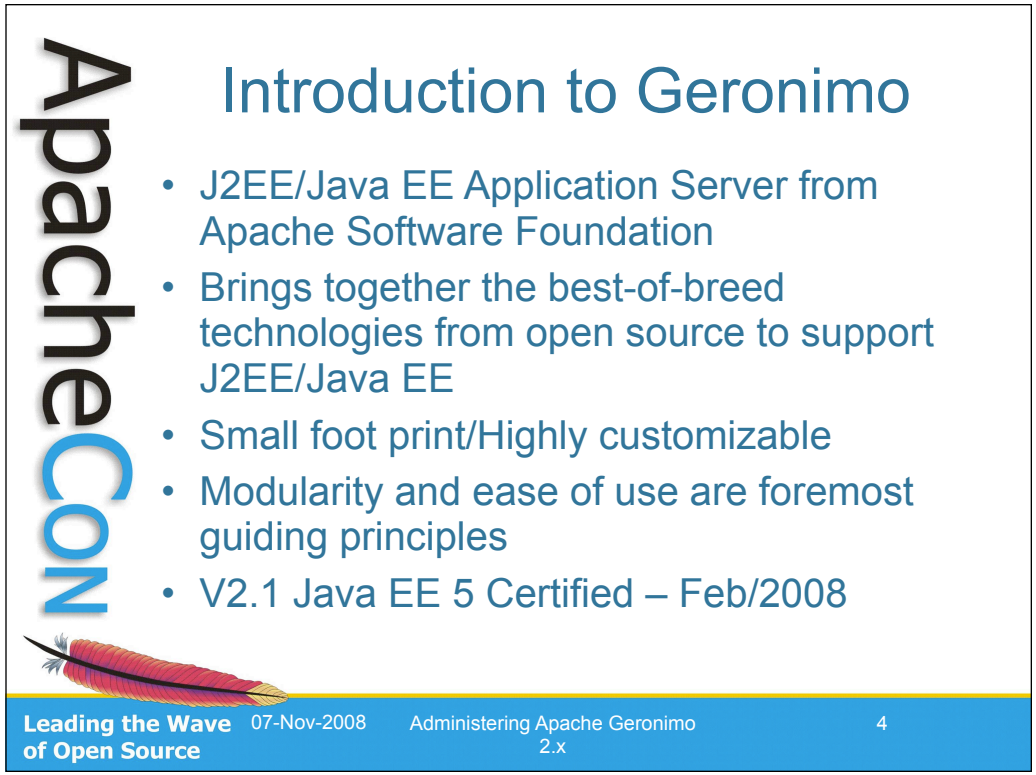
ApacheCon

Agenda

- Introduction to Geronimo
- Modifying stuff in Geronimo
- Adding stuff to Geronimo
- Replacing stuff in Geronimo
- Looking at stuff in Geronimo
- Summary

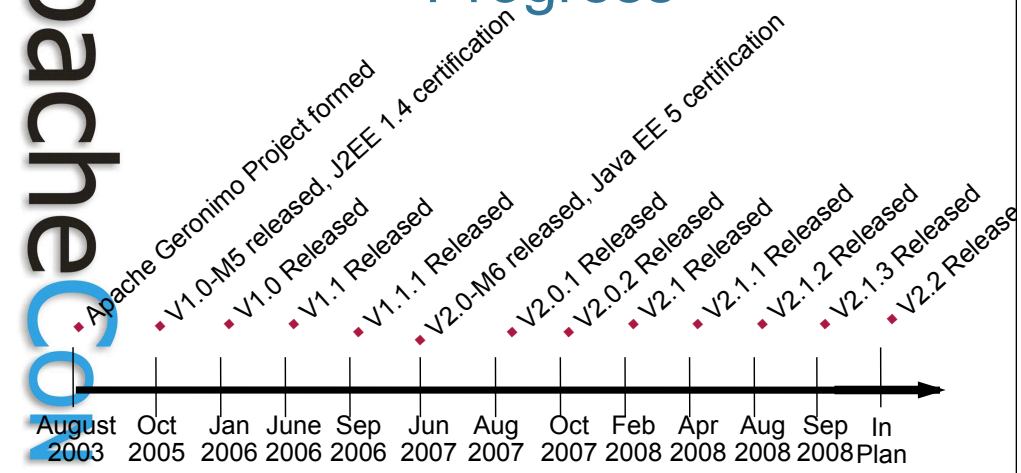
Agenda

- Introduction to Geronimo
- Modifying stuff in Geronimo
- Adding stuff to Geronimo
- Replacing stuff in Geronimo
- Looking at stuff in Geronimo
- Summary



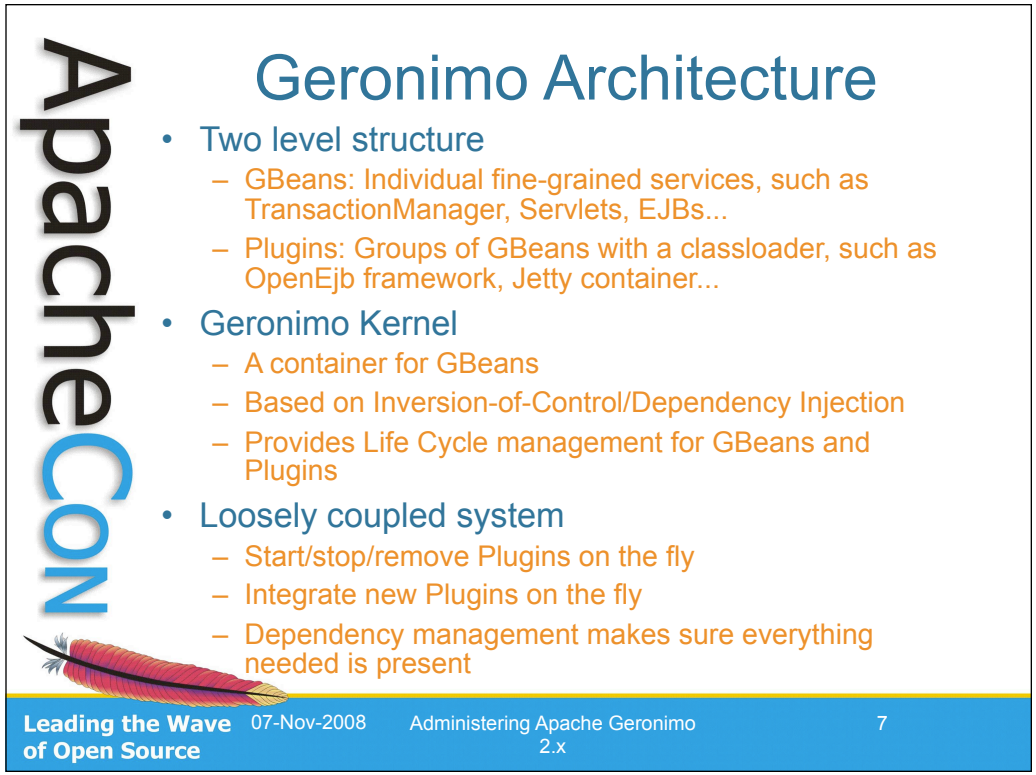
- Geronimo does not believe in re-inventing the wheel. It brings together best-of-breed technologies like Tomcat, Jetty, OpenEJB, ActiveMQ, Derby, Axis2, etc to deliver a J2EE/Java EE compliant container.
- Significant mile stones for Geronimo
    1. V1.0-M5 is J2EE 1.4 certified in Oct-2005.
    2. V2.0-M6 is Java EE 5.0 certified in Jun-2007.
    3. V2.1, Java EE 5.0 certified in Feb-2008, is the latest release.
- **Ease of use** is the foremost guiding principle.
    - Powerful Geronimo Console for users.
    - Rich set of development tools for developers – Geronimo Eclipse Plugin, JBoss to Geronimo (J2G) migration tool

Geronimo History and Progress



Geronimo Philosophies

- Completely and uniformly modular
  - Servers are assembled completely out of plugins
- Extensible
  - Plugins can be added and removed at any time
- Predictable and amenable to version control
  - Plugins are normally only affected by the plugins they depend on
  - Plugins contain all their configuration information
  - Plugins are best constructed using a build system such as maven



1. Geronimo Beans, GBeans in short, are the building blocks of Geronimo. In Geronimo, almost everything is a GBean. For e.g., Web Container, Application, Servlet, Connector etc. Configuration is a GBean that groups other GBeans.
2. Geronimo Kernel is a container for GBeans based on Inversion-of-Control/Dependency Injection. Kernel provides life cycle management for GBeans like start/stop, load/unload. GBeans can declare dependencies on other GBeans.
3. GBean architecture results in a loosely coupled system and enables starting/stopping/removing of components/services, integrating new components on the fly by way of plug-ins.

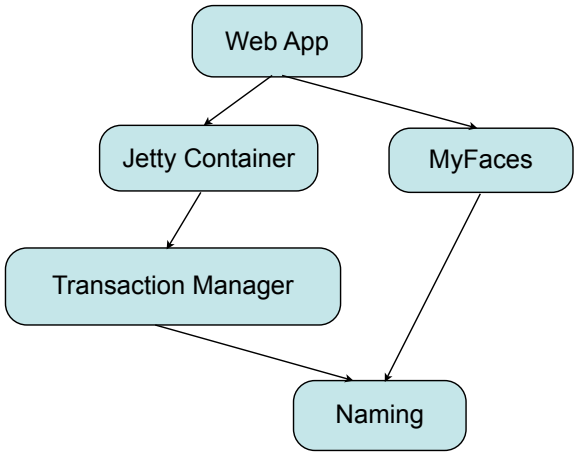
Two Java EE 5.0 certified server assemblies:

- Geronimo Tomcat distribution: Apache Tomcat web container + Apache Axis2 for Web Services
- Geronimo Jetty distribution: Jetty web container + Apache CXF for Web Services

Plugins

- Apache Directory, Apache Roller to name a few

Dependencies among Plugins



Administration Strategies

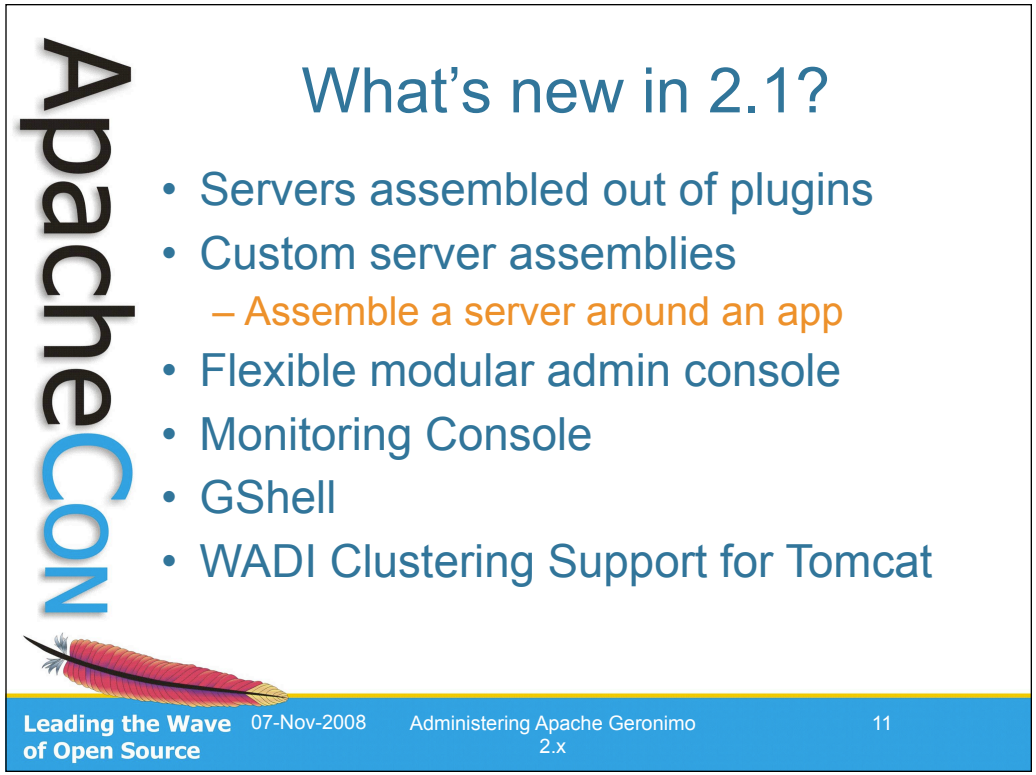
- Direct modification of config files or use of admin console (this talk)
  - Fine for experimentation and familiarizing yourself with geronimo
  - Good for emergency repairs
  - Not amenable to repeatability, version control, or multiple servers
- Building configuration into plugins and custom servers with maven (next talk)
  - Directly part of your build system
  - Under version control
  - Repeatable
  - Amenable to customization for dev, qa, and production

ApacheCon

What it contains?

- Apache Tomcat
- Jetty (Mort Bay)
- Apache Derby
- Apache OpenEJB
- Apache ActiveMQ
- Apache OpenJPA
- Apache Axis
- Apache Axis2
- Apache CXF
- Apache Yoko
- Apache Commons
- Apache jUDDI
- Apache Log4J
- HOWL
- TRANQL
- Castor
- WADI
- CGLIB

And many more…



•Starting with Geronimo 2.1, the servers are assembled entirely out of plugins.

•Lightweight server assemblies may be created that contain only the functional components required by your application(s). A new server containing a specified set of plugins can be assembled from an existing server using a gshell command or the admin console.

•The admin console is now component-based to mirror the server capabilities. This allows the admin console to provide flexible administrative capabilities that will mirrors the capabilities of a custom server assembly.

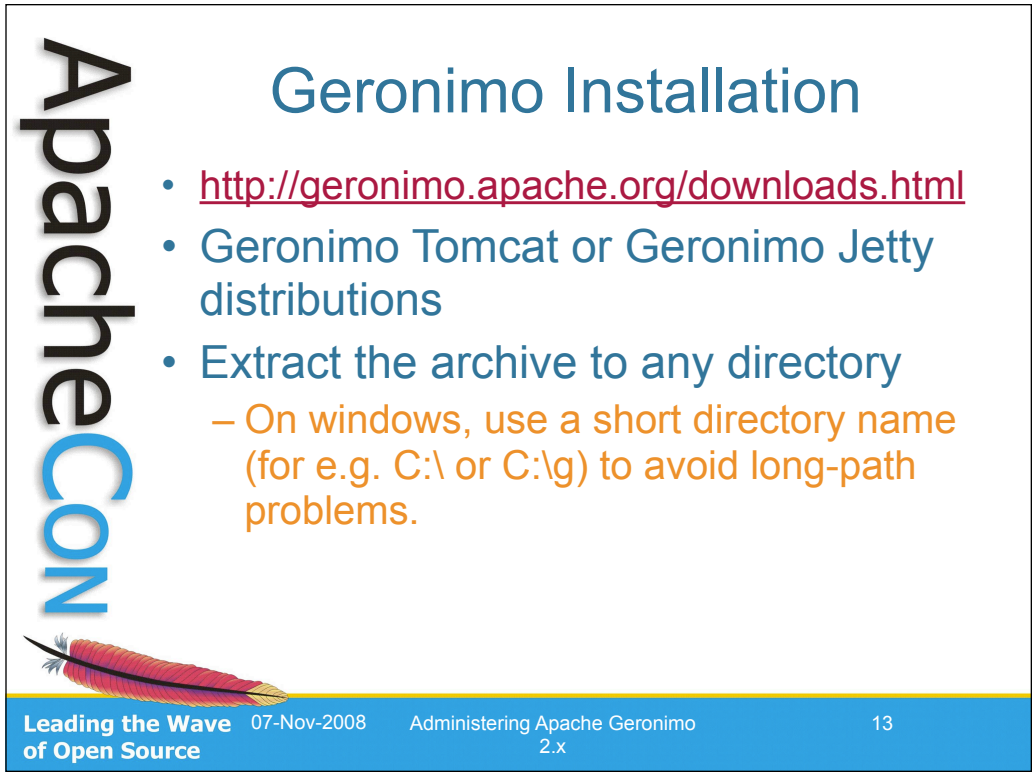
•The monitoring console plugin provides monitoring support in the Geronimo admin console. The monitoring console can gather statistics and performance data from multiple Geronimo servers and graphically display this data to users.

•GShell is a command-line processing environment that can be used for the execution of Geronimo commands. GShell is an extensible environment and includes support for editing, command history, and tab completion.

•WADI can now be used to support clustering of web applications for Geronimo configurations which use the Tomcat Web Container (WADI support for Jetty was in previous releases). Applications can be deployed to administratively-defined groups of Geronimo servers

How to get involved?

- Geronimo project web site
  - http://geronimo.apache.org/
- Mailing lists
  - user@geronimo.apache.org
  - dev@geronimo.apache.org
- Wiki
  - http://cwiki.apache.org/geronimo/

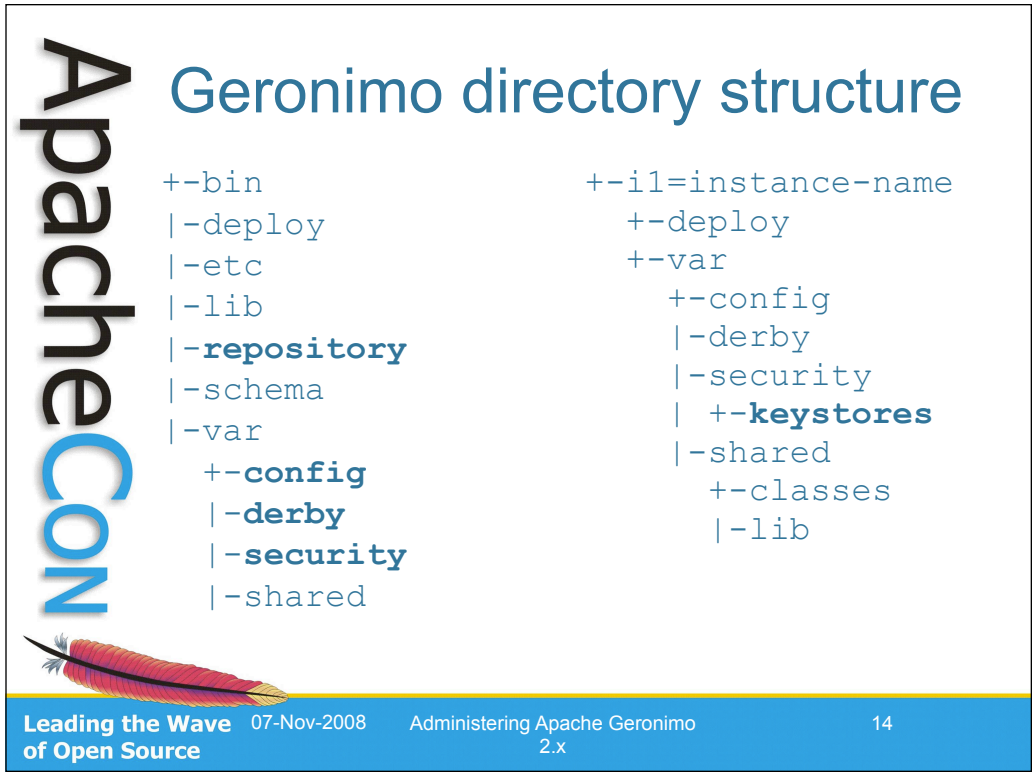


Two Java EE 5.0 certified server assemblies available for download:

•Geronimo Tomcat distribution: Apache Tomcat web container + Apache Axis2 for Web Services

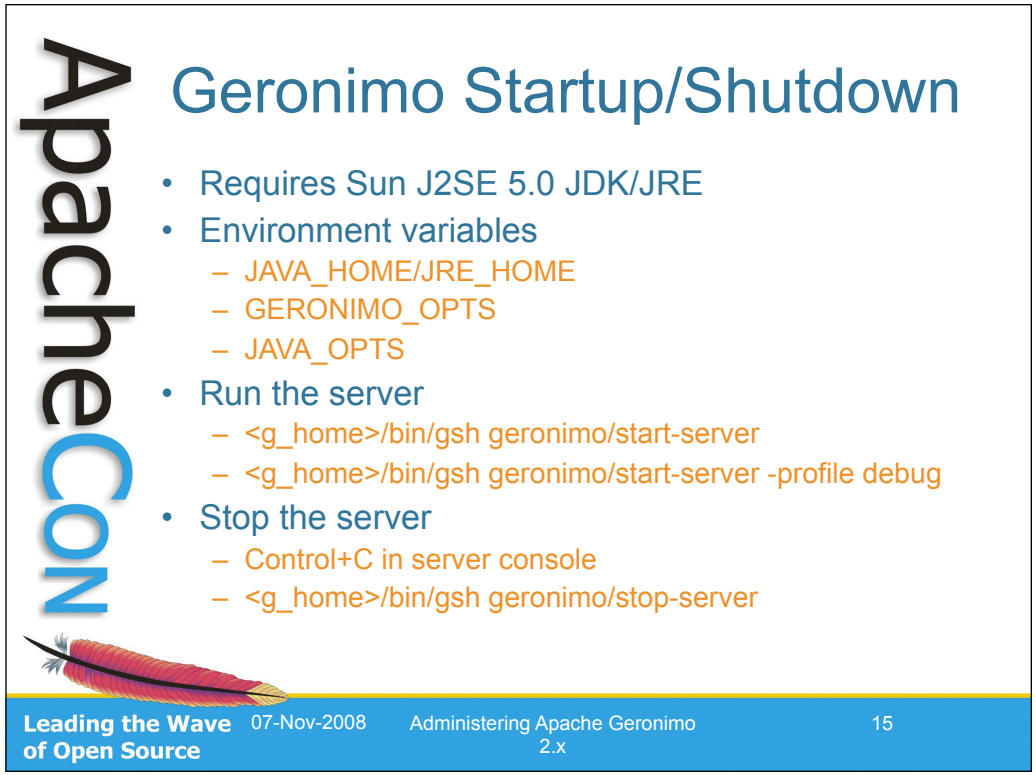
•Geronimo Jetty distribution: Jetty web container + Apache CXF for Web Services

On windows, extract the archive to root of a drive or use a short directory name with one or two letters (for e.g., **C:** or **C:\g**) to avoid problems due to long paths. **Note:** Extracting the archive and then moving the directory to a directory with short name does not help.



Directories that may contain sensitive information:

- repository
- var\config
- var\security
- var\derby



Prerequisite for Geronimo 2.1 is Sun J2SE 5.0 JDK/JRE. Geronimo may run on higher version of JRE/JDK but has not been certified.
Set JAVA_HOME or JRE_HOME environment variable to 5.0 JRE.

Use GERONIMO_OPTS to any system properties for the JVM.

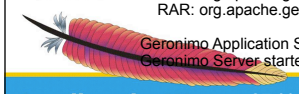
For e.g. GERONIMO_OPTS=-Dorg.apache.geronimo.server.name=i1

Use JAVA_OPTS to set any non standard options for the JVM.

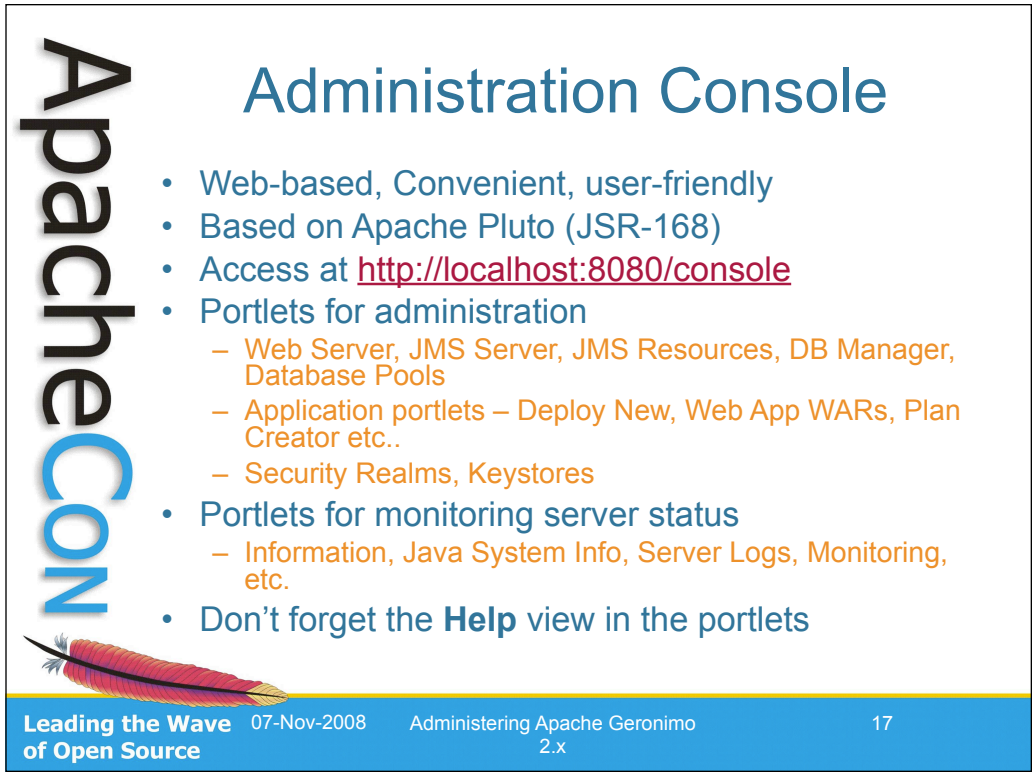
For e.g. JAVA_OPTS=-Xmx512m

Start the server using any of the following commands:

- To run in foreground: **<g_home>/bin/geronimo run**
- To run in background: **<g_home>/bin/geronimo start** or **<g_home>/bin/startup**
- To run in foreground under JPDA debugger: **<g_home>/bin/geronimo jpda run**
- To see help: **<g_home>/bin/geronimo**
- To run in G-shell: **<g_home>/bin/start-server**



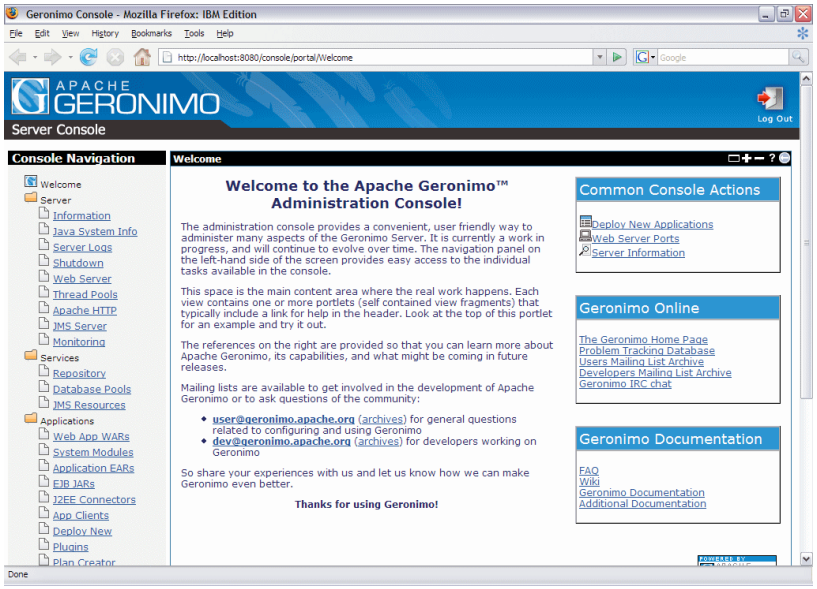
```
pb:failover-jetty-demo-2.2-SNAPSHOT david$ ./geronimo-jetty-farm-controller-2.2-SNAPSHOT/bin/gsh geronimo/start-server
Apache Geronimo (2.2-SNAPSHOT)

Type 'help' for more information.
-------------------------------------------------------------------------------------------------------------------------
Launching Geronimo Server...
Booting Geronimo Kernel (in Java 1.5.0_16)...
Module  1/20 org.apache.geronimo.framework/j2ee-system/2.2-SNAPSHOT/car                    started in   .000s
Module  2/20 org.apache.geronimo.framework/jee-specs/2.2-SNAPSHOT/car                      started in   .000s
Module  3/20 org.apache.geronimo.framework/rmi-naming/2.2-SNAPSHOT/car                     started in   .135s
Module  4/20 org.apache.geronimo.framework/j2ee-security/2.2-SNAPSHOT/car                  started in   .122s
Module  5/20 org.apache.geronimo.framework/gshell-framework/2.2-SNAPSHOT/car               started in   .000s
Module  6/20 org.apache.geronimo.framework/gshell-remote/2.2-SNAPSHOT/car                  started in   .000s
Module  7/20 org.apache.geronimo.framework/transformer-agent/2.2-SNAPSHOT/car              started in   .000s
Module  8/20 org.apache.geronimo.plugins.classloaders/geronimo-javaee-deployment_1.1MR3_spec/2.2-SNAPSHOT/car starte
Module  9/20 org.apache.geronimo.framework/server-security-config/2.2-SNAPSHOT/car        started in   .033s
Module 10/20 org.apache.geronimo.framework/plugin/2.2-SNAPSHOT/car                        started in   .076s
Module 11/20 org.apache.geronimo.framework/gshell-geronimo/2.2-SNAPSHOT/car               started in   .000s
Module 12/20 org.apache.geronimo.configs/j2ee-server/2.2-SNAPSHOT/car                     started in   .029s
Module 13/20 org.apache.geronimo.configs/transaction/2.2-SNAPSHOT/car                     started in   .228s
Module 14/20 org.apache.geronimo.configs/openjpa/2.2-SNAPSHOT/car                         started in   .006s
Module 15/20 org.apache.geronimo.configs/derby/2.2-SNAPSHOT/car                           started in   .000s
Module 16/20 org.apache.geronimo.configs/system-database/2.2-SNAPSHOT/car                 started in   .641s
Module 17/20 org.apache.geronimo.configs/plugin-farm-datasource/2.2-SNAPSHOT/car          started in   .135s
Module 18/20 org.apache.geronimo.framework/xmlbeans/2.2-SNAPSHOT/car                      started in   .000s
Module 19/20 org.apache.geronimo.framework/geronimo-gbean-deployer/2.2-SNAPSHOT/car       started in   .266s
Module 20/20 org.apache.geronimo.configs/plugin-farm/2.2-SNAPSHOT/car                     started in   .358s
Startup completed in 2.841s seconds
 Listening on Ports:
   1099 127.0.0.1 RMI Naming
   1527 127.0.0.1 Derby Connector
   9999 127.0.0.1 JMX Remoting Connector

 Started Application Modules:
   RAR: org.apache.geronimo.configs/plugin-farm-datasource/2.2-SNAPSHOT/car
   RAR: org.apache.geronimo.configs/system-database/2.2-SNAPSHOT/car

 Geronimo Application Server started
 Geronimo Server started in 0:00:05.433
```



•Web-based Geronimo Console provides a convenient and user-friendly way to administer many aspect of Geronimo.
•Based on Apache Pluto (JSR-168).
•Access at http://localhost:8080/console with default username **system** with a password **manager**.
•Flexible for v2.1 onwards. Add remove portlets dynamically.
•Portlets for administration
- •Web Server – To administer Tomcat/Jetty connectors
- •Shutdown – To shutdown Geronimo
- •JMS Server – To administer ActiveMQ connectors
- •JMS Resources – To administer JMS connection factories, queues, topics
- •Database Pools – To administer database pools for internal Derby and a whole lot of other database products.
- •Applications portlets
  - •Deploy new applications/services
  - •Start/stop/uninstall applications/services
  - •Plan Creator – Lets you deploy an application by creating a deployment plan on the fly
- •Security Realms – Create/Edit security realms
- •Keystores – Manage keystores/digital certificates
- •DB Manager – Create/delete databases, view tables/table data, run SQL

•Provides portlets for monitoring server status
- •Information – Server version, uptime, JVM version and memory usage graph
- •Java System Info – Java System properties
- •Server Logs – Geronimo logs, Web server logs, Derby logs
- •Monitoring – Monitor remote Geronimo servers
- •DB Info – Details of Embedded Derby

ApacheCon


Geronimo Console - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
http://localhost:8080/console/portal/Welcome
APACHE GERONIMO
Server Console
Log Out
Console Navigation
Welcome
Server
Information
Java System Info
Server Logs
Shutdown
Web Server
Thread Pools
Apache HTTP
JMS Server
Monitoring
Services
Repository
Database Pools
JMS Resources
Applications
Web App WARs
System Modules
Application EARs
EJB JARs
J2EE Connectors
App Clients
Deploy New
Plugins
Plan Creator
Welcome
Welcome to the Apache Geronimo™ Administration Console!
The administration console provides a convenient, user friendly way to administer many aspects of the Geronimo Server. It is currently a work in progress, and will continue to evolve over time. The navigation panel on the left-hand side of the screen provides easy access to the individual tasks available in the console.
This space is the main content area where the real work happens. Each view contains one or more portlets (self contained view fragments) that typically include a link for help in the header. Look at the top of this portlet for an example and try it out.
The references on the right are provided so that you can learn more about Apache Geronimo, its capabilities, and what might be coming in future releases.
Mailing lists are available to get involved in the development of Apache Geronimo or to ask questions of the community:
user@geronimo.apache.org (archives) for general questions related to configuring and using Geronimo
dev@geronimo.apache.org (archives) for developers working on Geronimo
So share your experiences with us and let us know how we can make Geronimo even better.
Thanks for using Geronimo!
Common Console Actions
Deploy New Applications
Web Server Ports
Server Information
Geronimo Online
The Geronimo Home Page
Problem Tracking Database
Users Mailing List Archive
Developers Mailing List Archive
Geronimo IRC chat
Geronimo Documentation
FAQ
Wiki
Geronimo Documentation
Additional Documentation
Done

Lightning trip through the admin console

Agenda

- Introduction to Geronimo
- Modifying stuff in Geronimo
- Adding stuff to Geronimo
- Replacing stuff in Geronimo
- Looking at stuff in Geronimo
- Summary

Important configuration files

- Located in <g_home>/var/config
- config.xml
  - List of plugins to start
  - Overridden GBean attributes
  - New GBeans (deprecated, better to use a new plugin)
- config-substitutions.properties
  - Properties used in config.xml
- artifact_aliases.properties
  - Substitute one plugin or jar for another

Customizing a GBean

- jetty web connector gbean in a plan:

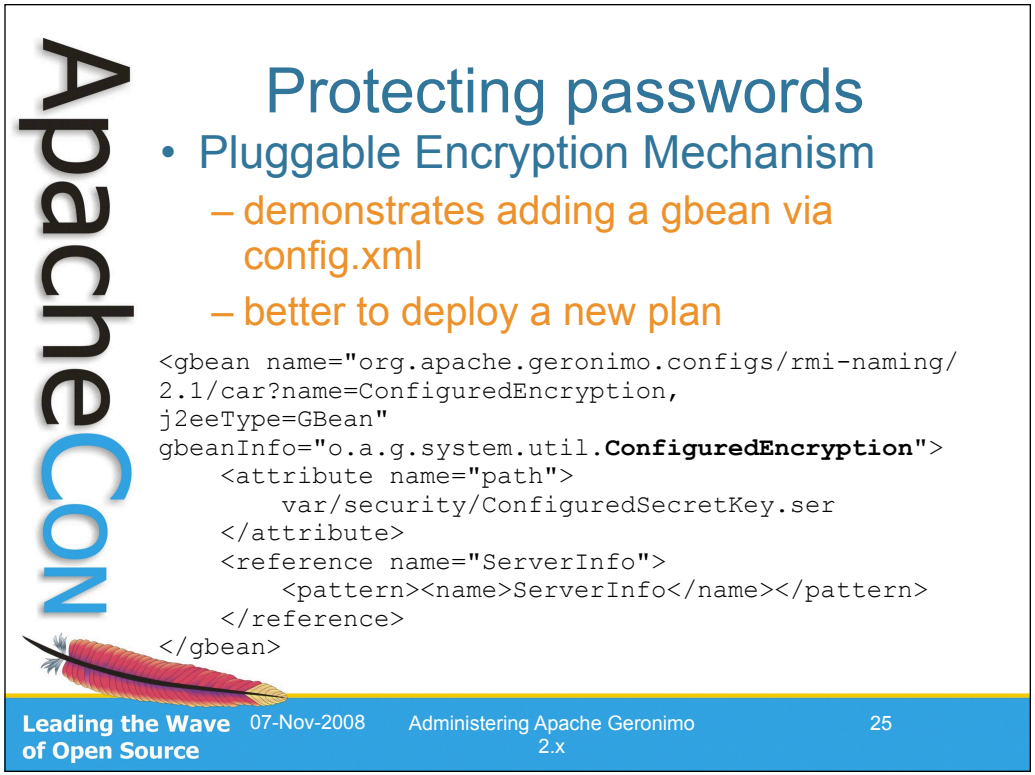
```
<gbean name="JettySSLConnector"
class="org.apache.geronimo.jetty6.connector.HTTPSSelectChannelConnector">
    <attribute name="host">localhost</attribute>
    <attribute name="port">8443</attribute>
    <attribute name="headerBufferSizeBytes">8192</attribute>
    <attribute name="keyStore">geronimo-default</attribute>
    <attribute name="keyAlias">geronimo</attribute>
    <attribute name="trustStore">geronimo-default</attribute>
    <attribute name="clientAuthRequired">false</attribute>
    <attribute name="algorithm">Default</attribute>
    <attribute name="secureProtocol">TLS</attribute>
    <attribute name="maxThreads">50</attribute>
    <reference name="JettyContainer">
      <name>JettyWebContainer</name>
    </reference>
    <reference name="ThreadPool">
      <name>DefaultThreadPool</name>
    </reference>
    <reference name="KeystoreManager">
      <name>KeystoreManager</name>
    </reference>
</gbean>
```

Modifying GBean in var/config/config.xml

```
   <module name="org.apache.geronimo.configs/jetty6/2.1.4-SNAPSHOT/car">
....
       <gbean name="JettySSLConnector">
          <attribute name="host">${ServerHostname}</attribute>
          <attribute name="port">${HTTPSPort + PortOffset}</attribute>
<!-- added attribute -->
          <attribute name="clientAuthRequired">true</attribute>
<!-- added attribute with new substitution variable -->
          <attribute name="maxThreads">${JettySSLMaxThreads}</attribute>
       </gbean>
    </module>
```

Modifying values in var/config/config-substitutions.properties

```
#Existing variables, new values
HTTPSPort=8443
ServerHostname=10.11.55.1
PortOffset=10

#New variables values
JettySSLMaxThreads=55
```



By default, passwords are encrypted with a hard-coded key.

<gbean name="org.apache.geronimo.configs/rmi-naming/2.1/car?name=ConfiguredEncryption,j2eeType=GBean"
gbeanInfo="org.apache.geronimo.system.util.ConfiguredEncryption">
<attribute name="path">var/security/ConfiguredSecretKey.ser</attribute>
<reference name="ServerInfo">
<pattern><name>ServerInfo</name></pattern>
</reference>
</gbean>

This will create a key the first time its started, after that it will keep using the saved key at the location specified. If you put a serialized SecretKeySpec there it will use it instead.

Obscuring passwords is not secure

- User passwords should be in ldap not a file
- any “encryption” only obscures passwords from casual leaks.... key is on disk nearby
- For actual security you need whatever needs the key to authenticate
  - e.g. map user credentials to db credentials.

Agenda

- Introduction to Geronimo
- Modifying stuff in Geronimo
- Adding stuff to Geronimo
- Replacing stuff in Geronimo
- Looking at stuff in Geronimo
- Summary

Plans

- Deploying anything to geronimo needs a plan
- Includes id
- Includes classloader description
  - parent classloaders
  - jars
- Specifies services (GBeans)
  - plain gbean description
  - from javaee app components
  - custom namespaces

Writing plans

- use admin console to generate plan
  - jdbc pools, jms connectors
  - security realms
  - javaee apps
  - plan is not saved when you deploy directly from console
  - copy plan out of console and save it
  - Change the id to something appropriate
- copy one from somewhere such as a geronimo plugin

Deploying plans

- Console will encourage you to directly deploy a plan it wrote for you
- Console “deploy new”
  - supply plan and artifact if for javaee
- GShell
  - ./bin/gsh deploy/deploy <module> <plan>
- “InPlace” option for javaee apps under development.

Datasource example

- Generate original plan in console
- Modify to set up both jta and non-jta datasources (e.g. for openjpa)
- edit transaction settings
- Use appropriate id.

ApacheCon

```
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2">
  <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2">
    <dep:moduleId>
      <dep:groupId>com.myco</dep:groupId>
      <dep:artifactId>derby-example</dep:artifactId>
      <dep:version>1.0</dep:version>
      <dep:type>car</dep:type>
    </dep:moduleId>
    <dep:dependencies>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>system-database</dep:artifactId>
        <dep:version>2.1.4-SNAPSHOT</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
    </dep:dependencies>
  </dep:environment>
  <resourceadapter>
    <outbound-resourceadapter>
      <connection-definition>
        <connectionfactory-interface>javax.sql.DataSource</connectionfactory-interface>
        <connectiondefinition-instance>
          <name>derby-example</name>
          <config-property-setting name="DatabaseName">derby-example</config-property-setting>
          <connectionmanager>
            <xa-transaction>
              <transaction-caching/>
            </xa-transaction>
            <single-pool>
              <max-size>20</max-size>
              <min-size>0</min-size>
              <idle-timeout-minutes>1</idle-timeout-minutes>
              <match-one/>
            </single-pool>
          </connectionmanager>
        </connectiondefinition-instance>
```

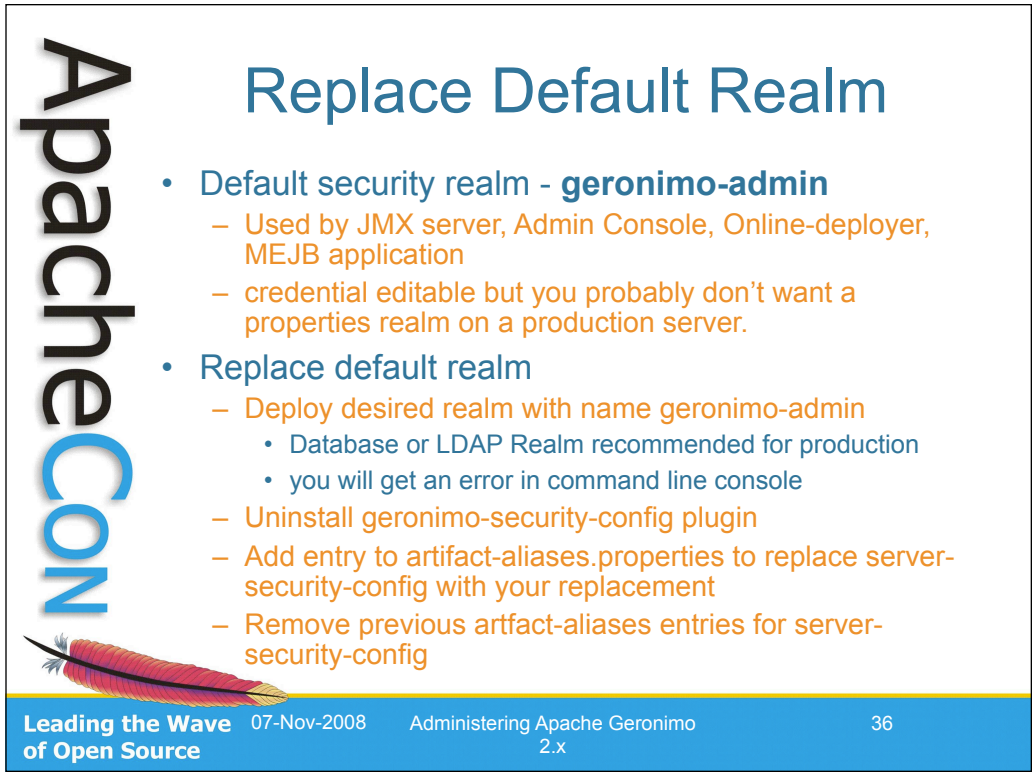
```
            <connectiondefinition-instance>
               <name>derby-example-notx</name>
               <config-property-setting name="DatabaseName">derby-example</config-property-setting>
               <connectionmanager>
                  <no-transaction/>
                  <single-pool>
                     <max-size>20</max-size>
                     <min-size>0</min-size>
                     <idle-timeout-minutes>1</idle-timeout-minutes>
                     <match-one/>
                  </single-pool>
               </connectionmanager>
            </connectiondefinition-instance>
         </connection-definition>
      </outbound-resourceadapter>
   </resourceadapter>
</connector>
```

Agenda

- Introduction to Geronimo
- Modifying stuff in Geronimo
- Adding stuff to Geronimo
- Replacing stuff in Geronimo
- Looking at stuff in Geronimo
- Summary

Why replace?

- Plugins form directed acyclic graph
- A plugin looks for services in its ancestor graph
- To replace a service, you need to get the plugin it's in into the ancestor graph of anything that uses it
- Provide an "alias" that lets your plugin substitute for the original
- Alias goes in var/config/artifact-aliases.properties



- Use OS provided security to restrict user access to <g_home>
  - var/config/config.xml may contain sensitive information.
  - var/security directory contains keystores and properties files used by default security realm.

- Default security realm **geronimo-admin** is used by JMX server, Admin Console, Online-deployer and MEJB application

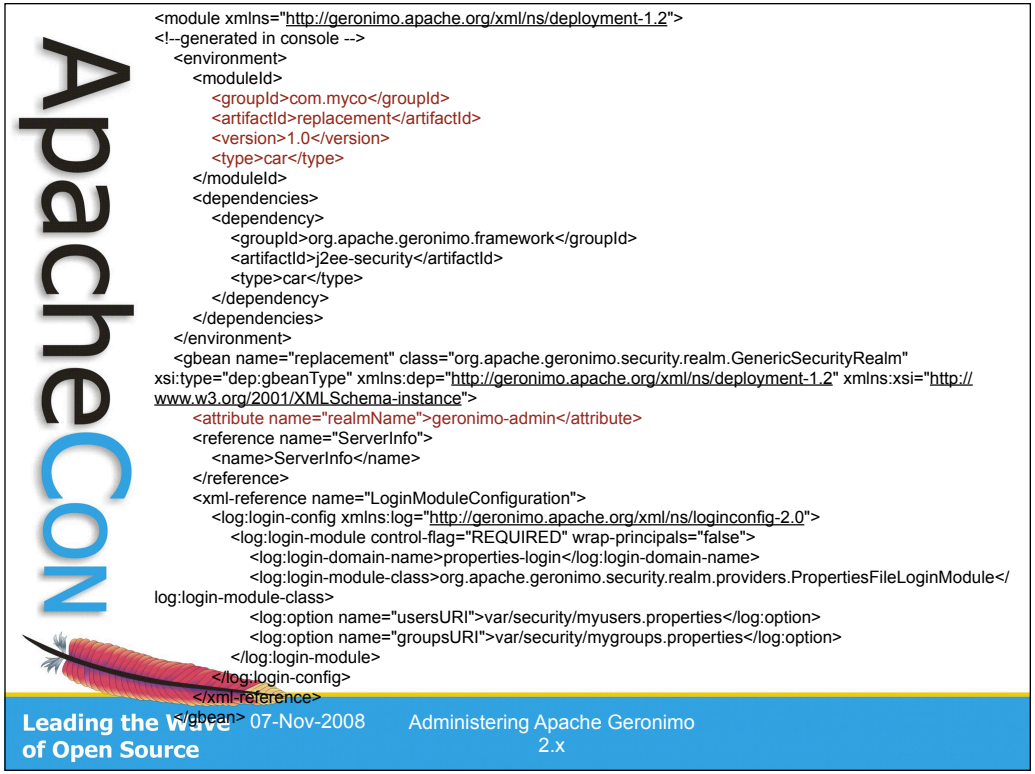
- Edit var/security/users.properties to change the password. The format is username=password with each user on a new line.
- User should belong to **admin** group.

Replace default realm:

1. Deploy a new realm with realm-name **geronimo-admin**
   - Either using offline deployer or online-deployer
2. Edit config.xml to disable the default-realm.

```
<module name="org.apache.geronimo.framework/server-security-config/2.1/car">
   <gbean name="geronimo-admin" load="false"/>
</module>
```

Use a Database or LDAP realm.


ApacheCon
<module xmlns="http://geronimo.apache.org/xml/ns/deployment-1.2">
<!--generated in console -->
<environment>
<moduleId>
<groupId>com.myco</groupId>
<artifactId>replacement</artifactId>
<version>1.0</version>
<type>car</type>
</moduleId>
<dependencies>
<dependency>
<groupId>org.apache.geronimo.framework</groupId>
<artifactId>j2ee-security</artifactId>
<type>car</type>
</dependency>
</dependencies>
</environment>
<gbean name="replacement" class="org.apache.geronimo.security.realm.GenericSecurityRealm"
xsi:type="dep:gbeanType" xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
<attribute name="realmName">geronimo-admin</attribute>
<reference name="ServerInfo">
<name>ServerInfo</name>
</reference>
<xml-reference name="LoginModuleConfiguration">
<log:login-config xmlns:log="http://geronimo.apache.org/xml/ns/loginconfig-2.0">
<log:login-module control-flag="REQUIRED" wrap-principals="false">
<log:login-domain-name>properties-login</log:login-domain-name>
<log:login-module-class>org.apache.geronimo.security.realm.providers.PropertiesFileLoginModule</
log:login-module-class>
<log:option name="usersURI">var/security/myusers.properties</log:option>
<log:option name="groupsURI">var/security/mygroups.properties</log:option>
</log:login-module>
</log:login-config>
</xml-reference>
</gbean>
Leading the Wave
of Open Source
07-Nov-2008
Administering Apache Geronimo
2.x

```
<!--copied from geronimo default plan -->
   <gbean name="CredentialStore"
class="org.apache.geronimo.security.credentialstore.SimpleCredentialStoreImpl"/>
   <gbean name="geronimo-default" class="org.apache.geronimo.security.keystore.FileKeystoreInstance">
      <attribute name="keystoreName">geronimo-default</attribute>
      <attribute name="keystorePath">var/security/keystores/geronimo-default</attribute>
      <attribute name="keystorePassword">secret</attribute>
      <attribute name="keystoreType">JKS</attribute>
      <attribute name="keyPasswords">geronimo=secret</attribute>
      <reference name="ServerInfo"><name>ServerInfo</name></reference>
   </gbean>
</module>
```

artifact alias entries

org.apache.geronimo.framework/server-security-config//car=com.myco/replacement/1.0/car
org.apache.geronimo.framework/server-security-config/2.1.4-SNAPSHOT/car=com.myco/replacement/1.0/car

Agenda

- Introduction to Geronimo
- Modifying stuff in Geronimo
- Adding stuff to Geronimo
- Replacing stuff in Geronimo
- Looking at stuff in Geronimo
- Summary

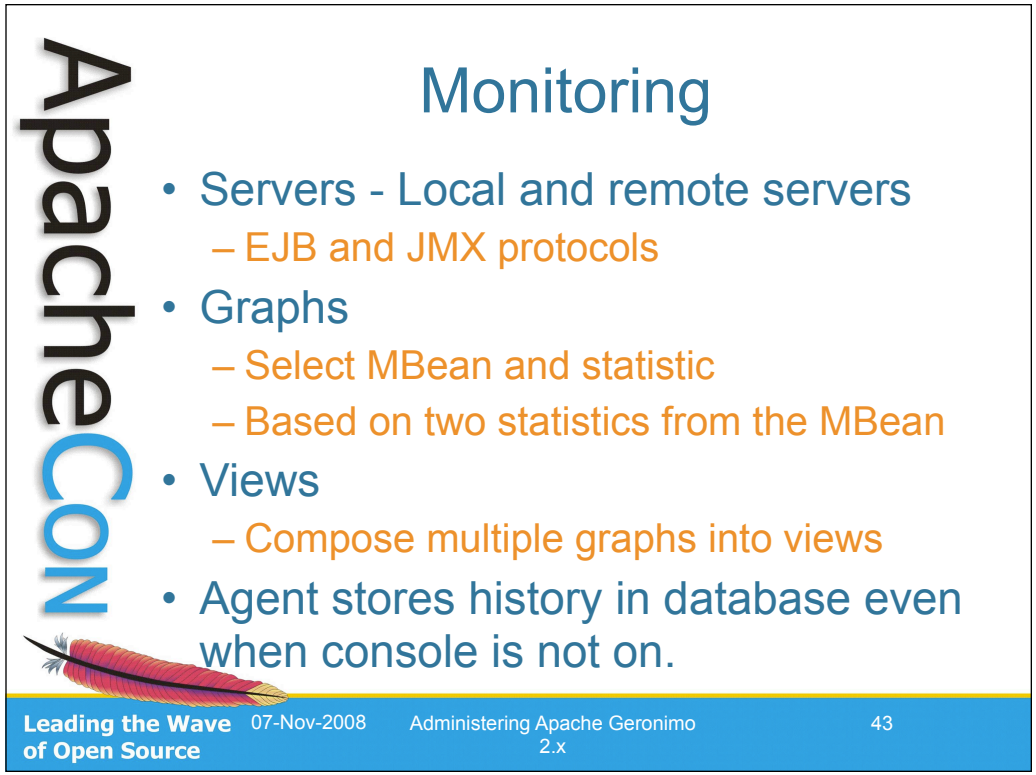
Keystores portlet provides an easy way to manage keystores, digital certificates, private keys and certificates.

Configuring HTTPS connectors.

Geronimo 2.1 has support for all types of keystores. Only JKS keystores supported in v2.0 and prior releases.

Certificate administration

- CA admin in console
  - set up certificate authority in server
  - validate client cert requests
  - fulfill client cert requests
- CAHelper app for client certificates
  - generates certificate signing requests
  - retrieves signed certificates
  - installs servers CA certificate into browser.

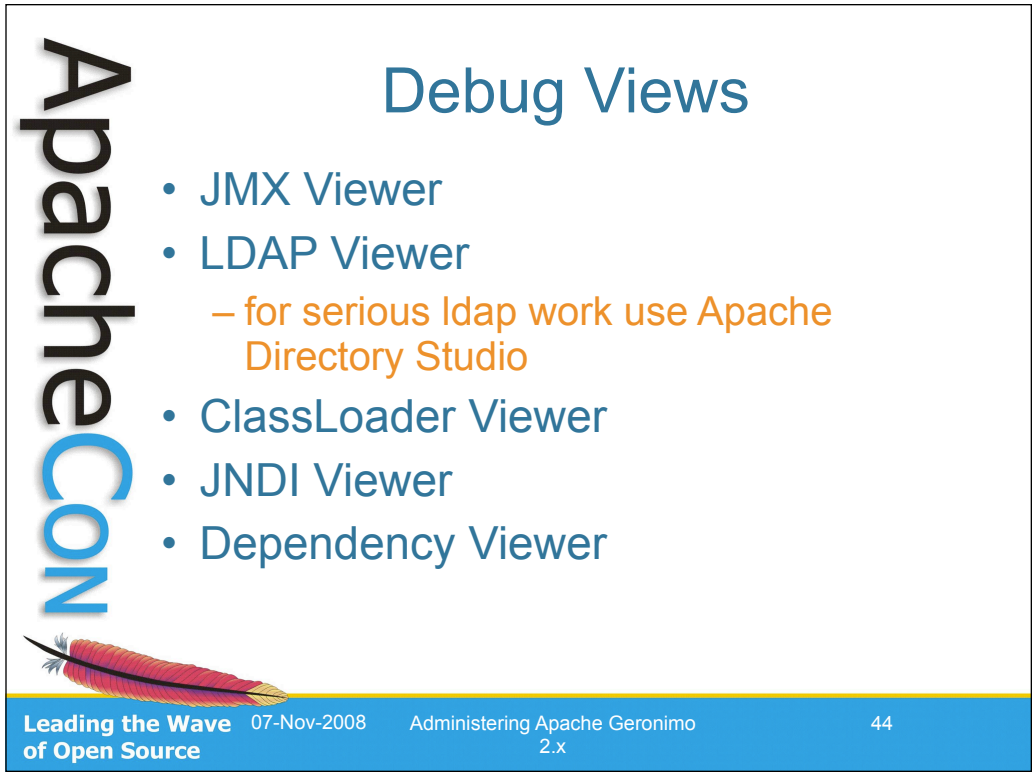


Monitor local and remote Geronimo servers.

EJB (using MEJB application) and JMX protocols. Note: JMX protocol needs JMX Agent to be running in the remote JVM.

Create Graphs for single statistic or two statistics using an arithmetic operation.

Views can be created by composing graphs.



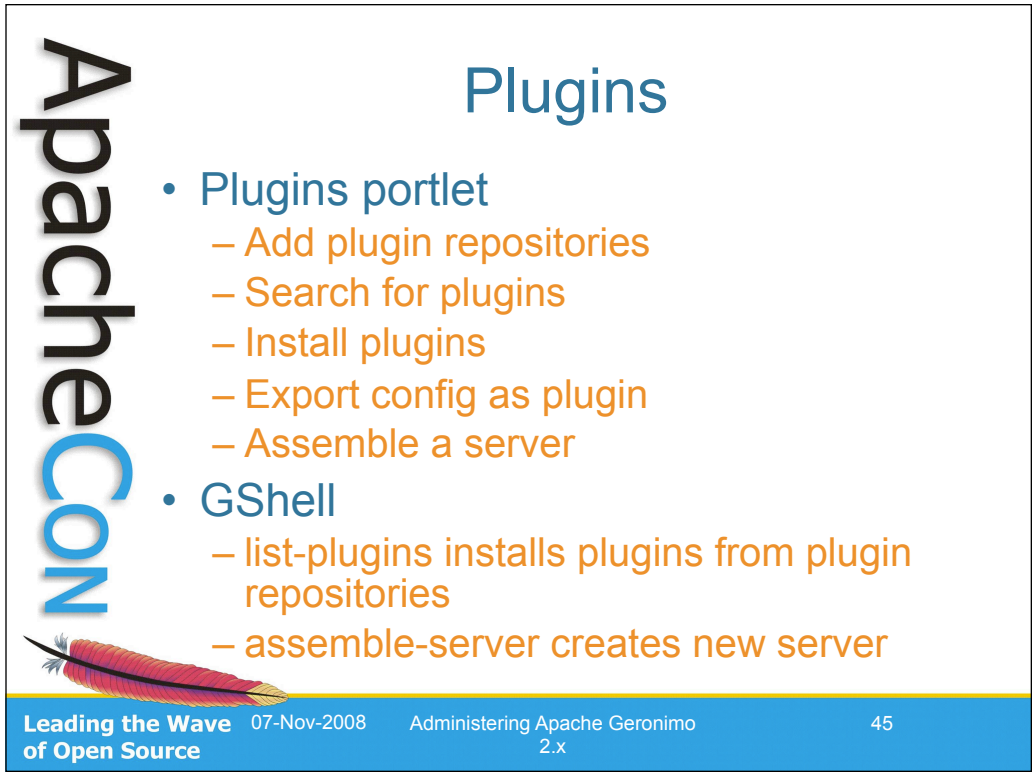
JMX Viewer – Integrated JMX client to view and operate on Geronimo MBeans.

LDAP Viewer – To browse an LDAP server.

ClassLoader Viewer – View classLoader hierarchies and search classes.

JNDI Viewer – To view the JNDI context of various modules.

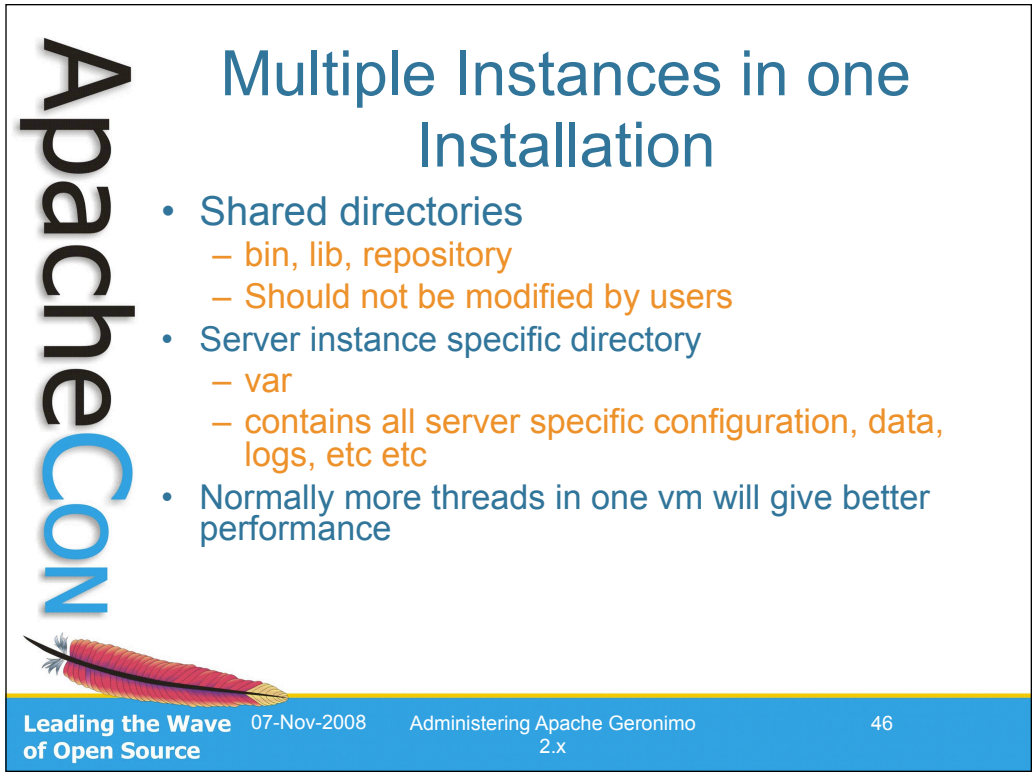
Dependency Viewer – To view the dependency hierarchy of the configurations in Geronimo.



Both remote and file system based repositories are supported.

Currently installed configurations can be exported as plugins.

Assemble a server by selecting just the components you want. Geronimo will extract a server with just those components and required dependencies.



•Multiple instances share the bin, lib and repository directories

•Make a directory with the instance name under <g_home> (i.e. <g_home>/<inst_name>, call this <inst_home>) and copy <g_home>/var directory to <inst_home>/var.

•Each instance will have its own copy of **var** directory.

•Change PortOffset in <inst_home>/var/config/config-substitutions.properties. Use a different PortOffset (e.g. 10, 20, …) with each instance.

•The server instance to be run is specified using the system property **org.apache.geronimo.server.name.** Set GERONIMO_OPTS=-Dorg.apache.geronimo.server.name=<inst_name>

•Run the server using <g_home>/bin/geronimo

Creating Multiple Instances

- just copy var directory
  - to server1/var, server2/var, etc
  - copies all local modifications
- use gshell command (trunk)
  - creates a new server1/ directory and installs all the plugins from existing server on new server
  - does not copy customizations
  - untested (new in trunk)

Starting multiple instances

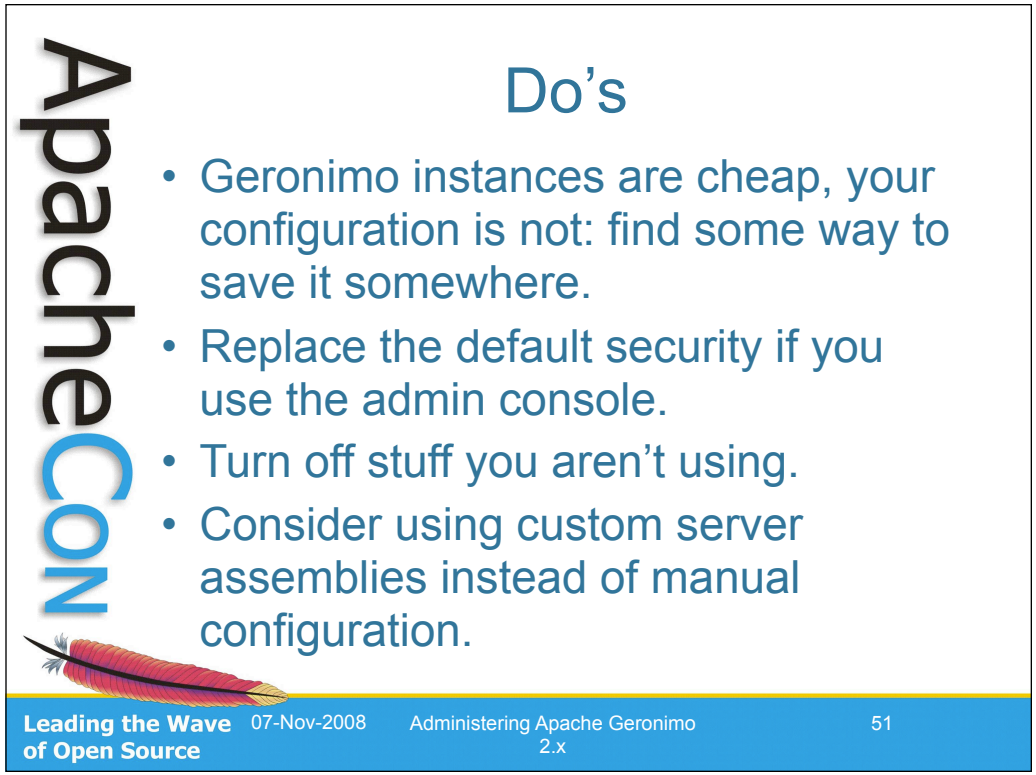
- Specify server name and PortOffset on command line
  - bin/gsh geronimo/start-server \
    -G config.substitution.PortOffset=10 \
    -G server.name=server1
- Possible to override other config substitution properties on command line if necessary

Agenda

- Introduction to Geronimo
- Modifying stuff in Geronimo
- Adding stuff to Geronimo
- Replacing stuff in Geronimo
- Looking at stuff in Geronimo
- Summary

A typical scenario

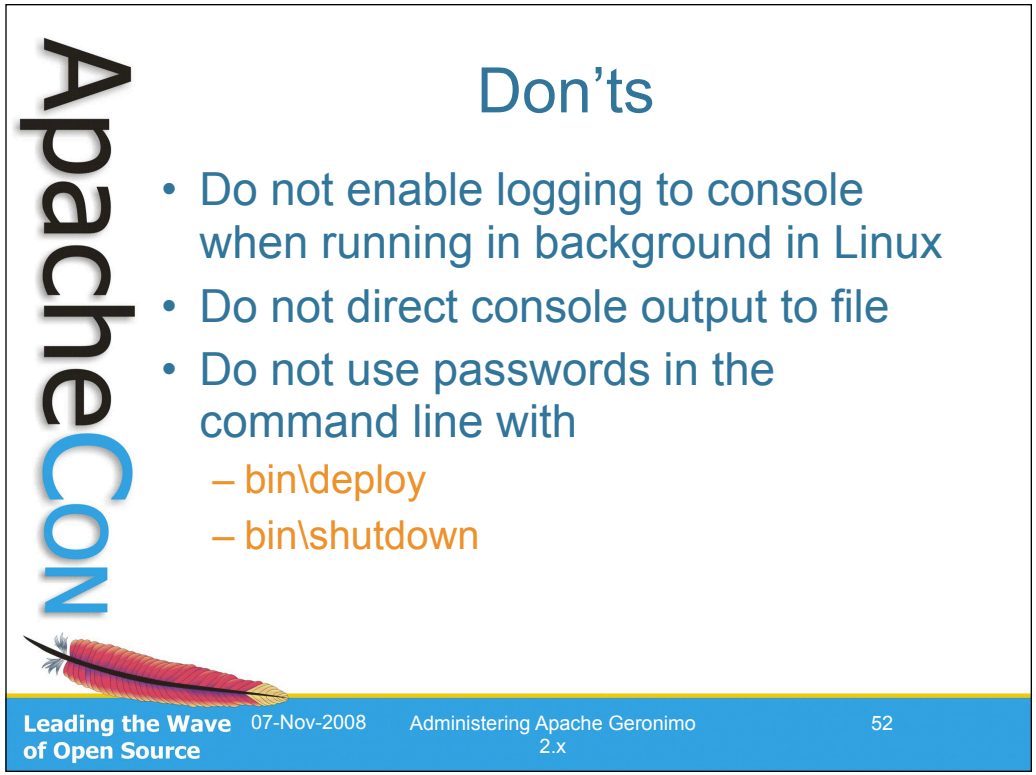
- Set up certificates
- Set up security realm
- Set up database connection pools
- Configure Activemq broker and connection pools
- Deploy Applications
- Check server Logs
- Set up monitoring



Use OS provided security to protect the geronimo installation directory.

Use Database/LDAP Realms

Use digested passwords with Database realm



Running in background results in output being written to a file which get locked until the server process is killed. This may result in a situation where the output file fills the disk and server need to be brought down to rotate the output file.

Using deploy and shutdown commands provide –password option, it is not recommended to use this option as the passwords may appear in command history.

Administering Apache Geronimo 2.x

Q & A

Question/Comments?

david_jencks@yahoo.com

Resources

- http://geronimo.apache.org
- http://cwiki.apache.org/geronimo/
- Geronimo Mailing lists
  - user@geronimo.apache.org
  - dev@geronimo.apache.org
- IBM developerWorks
  - http://www.ibm.com/developerworks/opensource/top-projects/geronimo.html