

The New Generation of Lightweight Server - WebSphere Liberty Profile

Zhu Xiulei (xiuleizh@cn.ibm.com)

Zhang Guannan (zguannan@cn.ibm.com)

2012-07-20



Agenda

- Introduction
- Light Touch Administration of Liberty Servers
- Security
- Performance
- Summary

Liberty Introduction

Overview

Why need Liberty? – Industry request !

Cloud Deploy and appliance

- High density deployment
- Small size
- Small memory footprint
- Quick start
- Simple and unified config

Seamlessly Migration

- Do not need develop on tomcat and then migrate to Websphere

Modulization and Dynamic

- Customize server runtime
- Dynamic load modules
- Change config not requiring restart

Developer Friendly

- Debug
- Restart
- Deploy
- Config
- Update
- Install

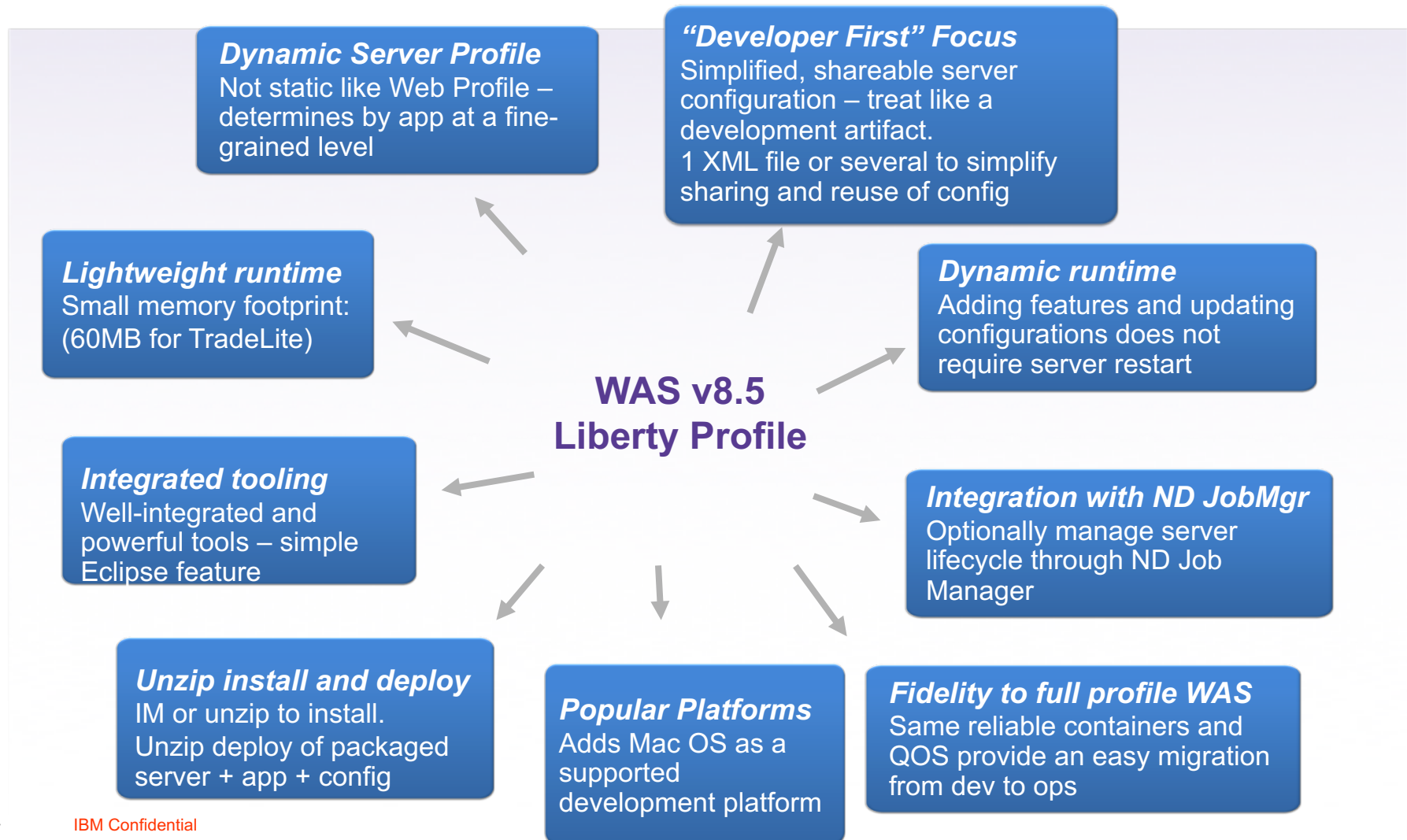
Frequency

Time consuming



WAS V8.5: Introduces the lightweight WAS “Liberty” Profile

A highly composeable, dynamic Server profile



What is WebSphere Application Server Liberty Profile?

- A new flexible dynamic profile of WAS which is focused FIRST on the development experience.
 - **Composable** -- the function is very modular and flexibly decoupled, allowing you to specify just what function you need for the applications you are serving.
 - **LightWeight** -- the Liberty Profile uses a number of approaches to optimize the loading of functions, which results in a footprint significantly less than traditional WebSphere Application Server.
 - **Dynamic** -- many of the runtime functions may be dynamically updated by simply changing the configuration XML; applications may be added or updated by simply replacing the application file in the file system directory.
 - **Fast** -- due to the composable design and other factors, the Liberty Profile is able to execute very quickly.
 - **Simplified Configuration** -- one single config file for quick time to productivity, covering all aspects of the server, the applications, and the resources required by the applications. This makes it easy to share configurations across development teams and environments.
- Fidelity with full-profile WAS
 - But radically refactored to focus on the development experience
- Initially focused on development and test of web, mobile and OSGi apps.

What is WebSphere Application Server Liberty Profile?

- Support for Liberty Profile in Rational Application Developer
 - Enterprise development - advanced programming, cloud, collaboration, and quality tools
 - Available stand-alone or bundled in WAS - Tools Edition and WAS Network Deployment - Tools Edition
- Support for Liberty Profile in WebSphere Application Server Development Tools for Eclipse
 - Subset of RAD focused on core programming models
 - Simple Eclipse feature update for Eclipse 3.6(Helios) or 3.7(Indigo)
 - Available unsupported at no charge, or supported for a fee.
 - WAS for Developers - Tools Edition for Eclipse
- Development tools can install the runtime
- Additional platform support for z/OS - runtime only
and Mac OS runtime & tools (development only not supported for production)
- Choice of either Oracle or IBM's Java (where available)
 - IBM JDK the minimum supported level is 6.0 (J9 2.6) SR 1
 - Oracle JDK the minimum supported level is Java 6 update 26

Liberty Introduce

Usage scenarios

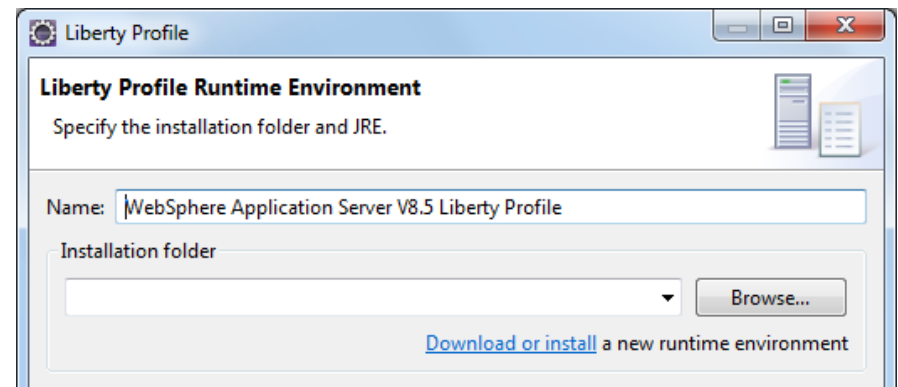
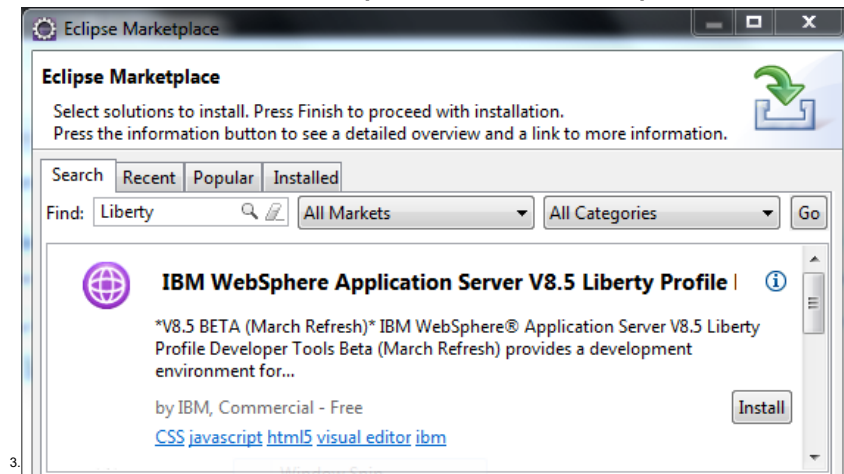


No Hurdles to Install

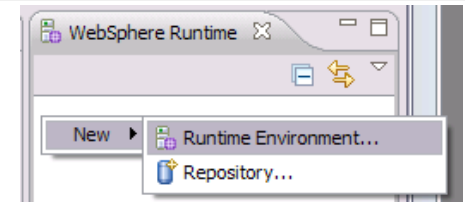
- Tools and runtime are **no-charge** for development. No time limit
- Eclipse feature install for tools; 40MB zip download for server profile.
 - Installation Manager also supported → same installed result.
- Archive install

2 minutes from “Nothing” to “Done” :

1. Install WAS Developer Tools for Eclipse Feature



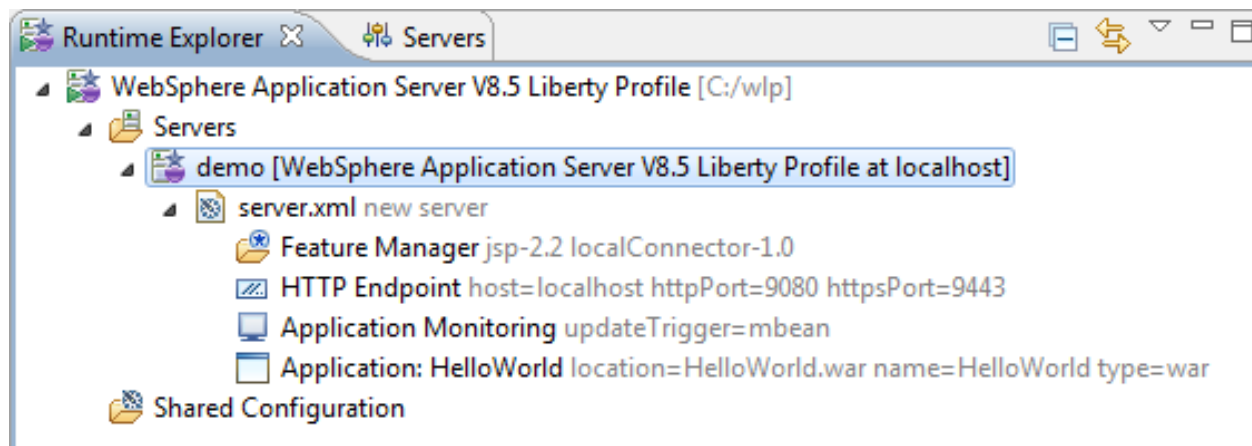
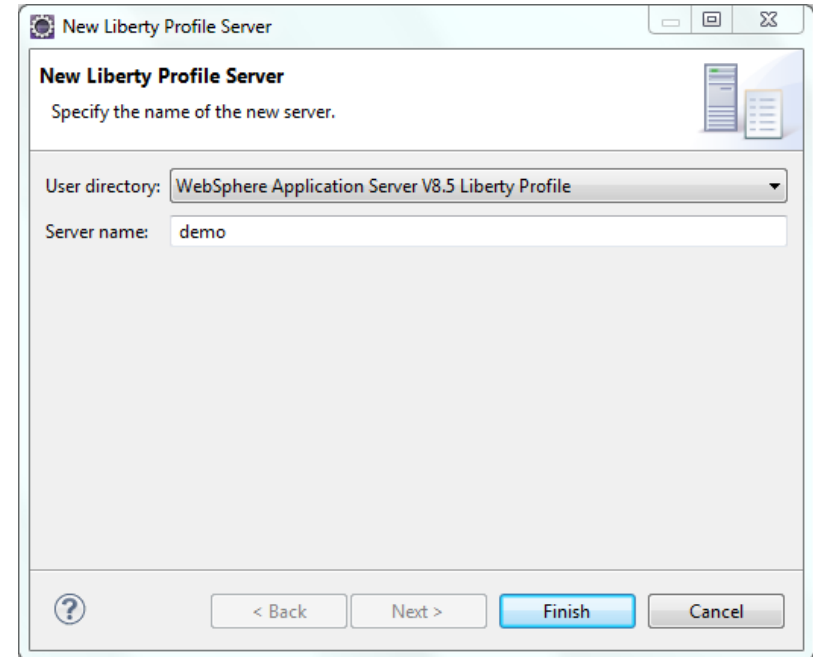
2. Use the Tools to download the WAS Liberty Profile
or download 40MB zip from WASdev.net



www.wasdev.net

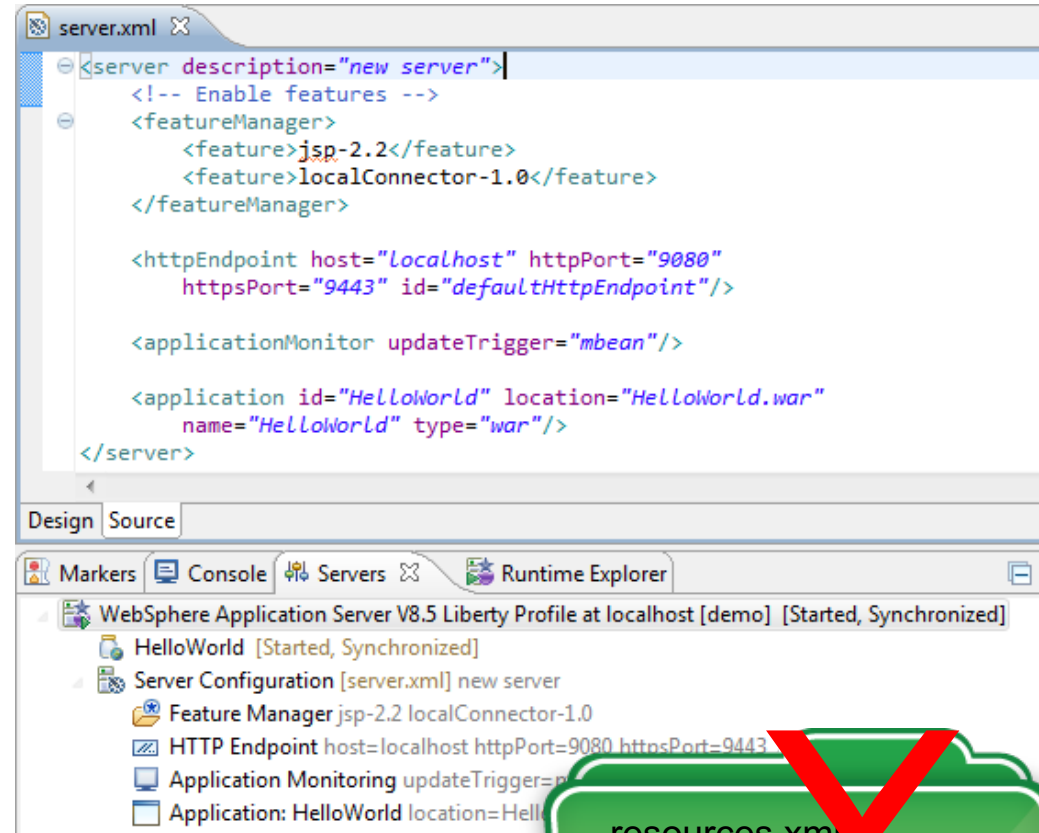
Create a Lightweight WAS Server in WDT

- Create a lightweight WAS server configuration in seconds



Simplified Server Configuration

- Simplest case: One XML file for all server config
- Editable within the workspace
- Exportable, shareable, versionable



No need for Admin Console, wsadmin,
or extended EARs



Simplified Configuration

```
<server>
```

```
<featureManager>
```

```
<feature>jsp-2.2</feature>
```

```
<feature>jdbc-4.0</feature>
```

```
</featureManager>
```

```
<logging traceSpecification="webcontainer=all=enabled:*=info=enabled" />
```

```
<application name="tradelite" location="tradelite.war" />
```

```
<dataSource jndiName="jdbc/TradeDataSource">
```

```
<properties.derby.embedded databaseName="${server.config.dir}/tradedb"/>
```

```
</dataSource>
```

```
</server>
```

features control which capabilities (bundles) are installed in the server

'singleton' configurations specify properties for a runtime service like logging

'instance' configurations specify multiple resources like applications and datasource definitions

Any of this configuration could be put into a separate xml file and 'included' in this 'master' configuration file

Flexible Configuration

▪ Shareable config snippets

```
<server>
  ...
  <include location="http://cfgserver/global.xml" />
  <include location="${shared.config.dir}/datasource.xml" />
</server>
```

server.xml

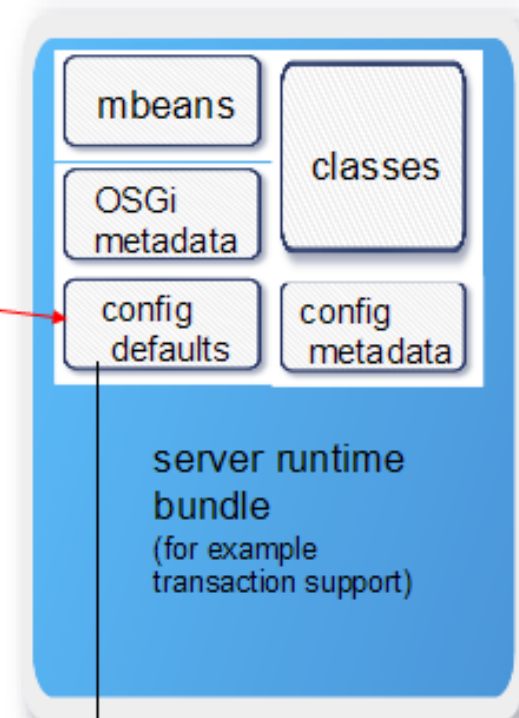
- Config can be componentized at any level of granularity, from 1 file to many.
 - Can use WDT to associate config snippets with a server config.
- Visualization through WDT tools as a single logical view.
- Team development: keep the application and configuration components together.

Flexible Configuration

- Configuration defaults are specified by contributing features
- Configuration by exception – any property can be overridden in user-specified server config
- Config can be changed dynamically, changes are “observed” and are injected back into the contributing feature immediately

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>transaction-1.1</feature>
  </featureManager>
  <transaction timeout="30" />
</server>
```

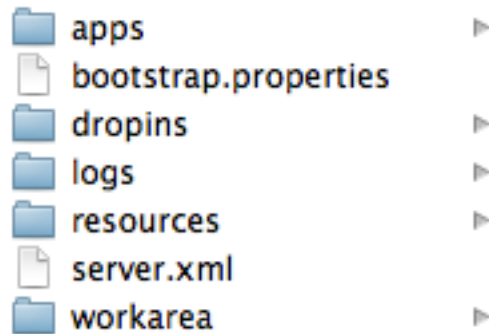
server.xml



```
id=com.ibm.ws.transaction
name=transaction
attribute id=timeout
default=100
```

Application Deploy

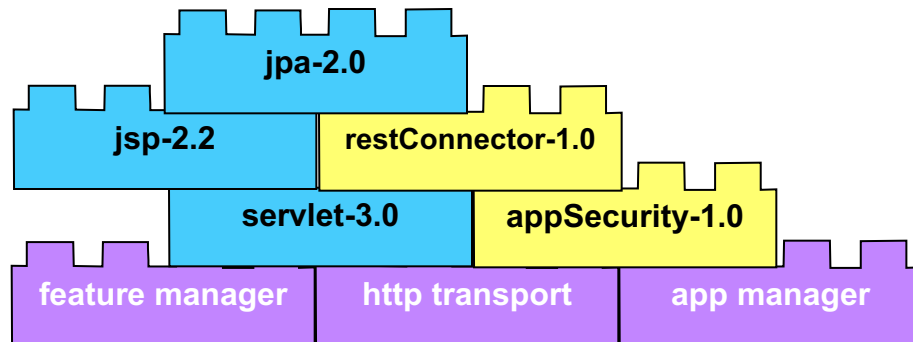
- Applications deployed via
 - Monitored directory
 - Configuration
- Run apps from the file system, or a URL
- Can search server specific apps or shared apps directories



Highly composable runtime based on 'features'



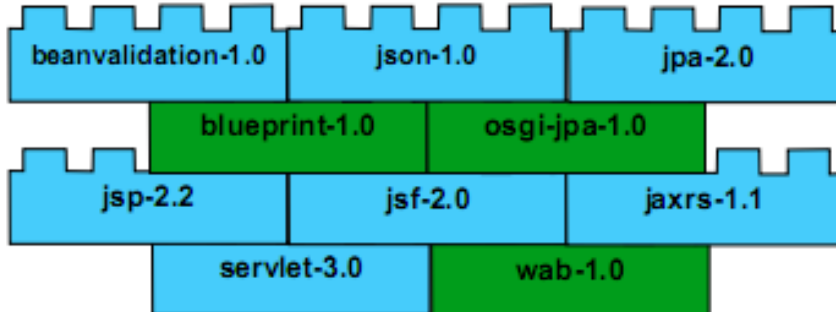
Full WAS Profile



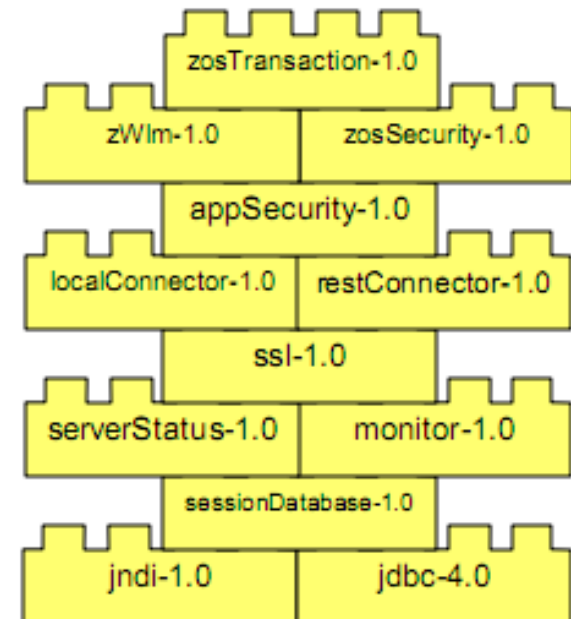
**WAS v8.5
Liberty Profile**

Dynamic enablement of feature set in application

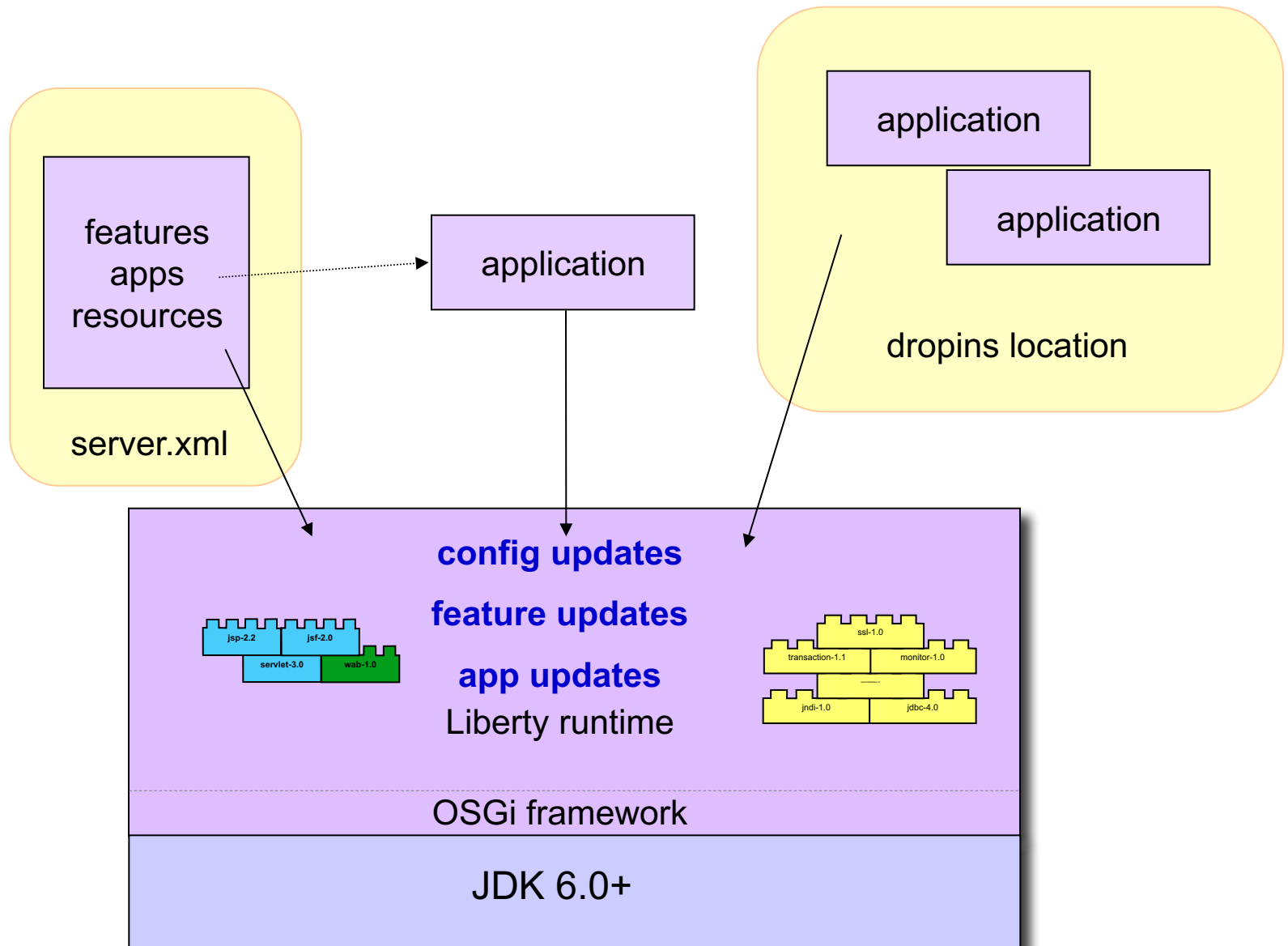
Application



Runtime

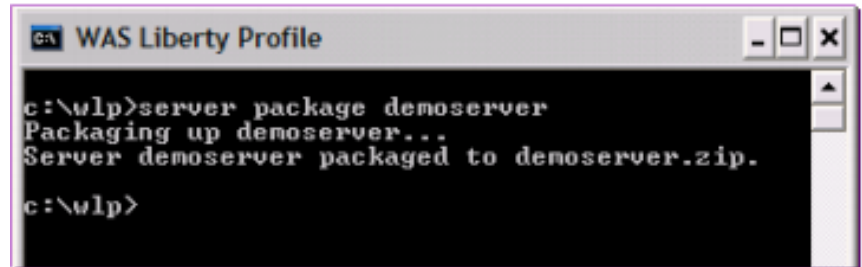
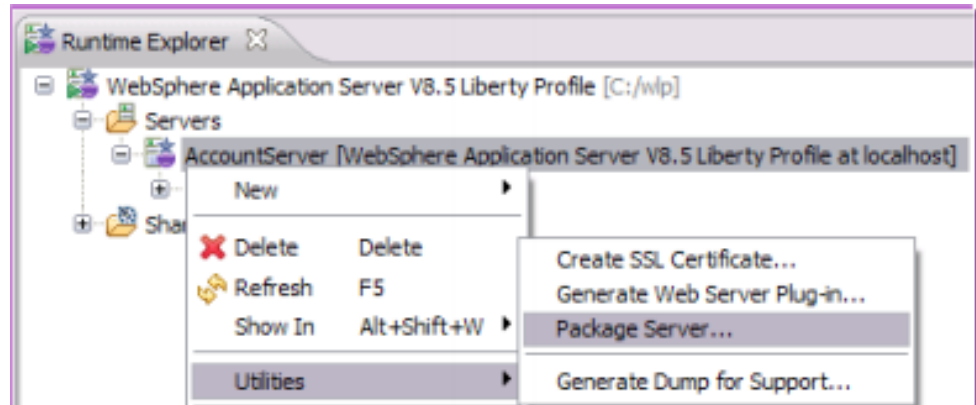


Dynamic Runtime Provisioning



New Deployment Options

- Package up a compressed archive of a configured Liberty server type along with its applications
 - Directly from Eclipse environment
 - Resulting zip can be copied to integration or production environment and unzipped.
- For test automation outside the IDE, a command-line program to manage the lifecycle of server instances:
 - Create [serverName]
 - Start and stop [serverName]
 - Package [serverName]
 - Status [serverName]
- Updates to configuration of running server are effective immediately.
- Add/remove apps dynamically by drag/drop to monitored directory.



What This Means For Developers

- A new WAS server type for Eclipse which is focused on the development experience
 - Fast to download, unzip and set up
 - No more waiting for application updates or server restarts

- Good fidelity retained through evolution of the WAS server type
 - Simple promotion of application through development, test and deployment systems

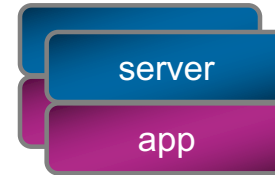
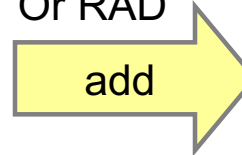
Light Touch Administration

Creating the Production Image

Installation Manager,
Or zip download



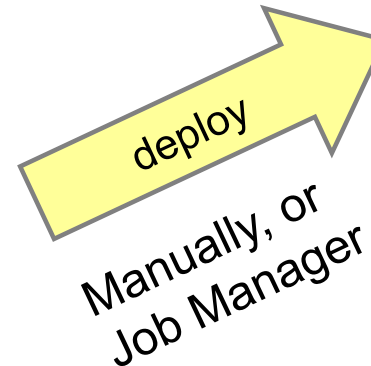
Manually,
Eclipse,
Or RAD



Manually,
Eclipse,
or RAD



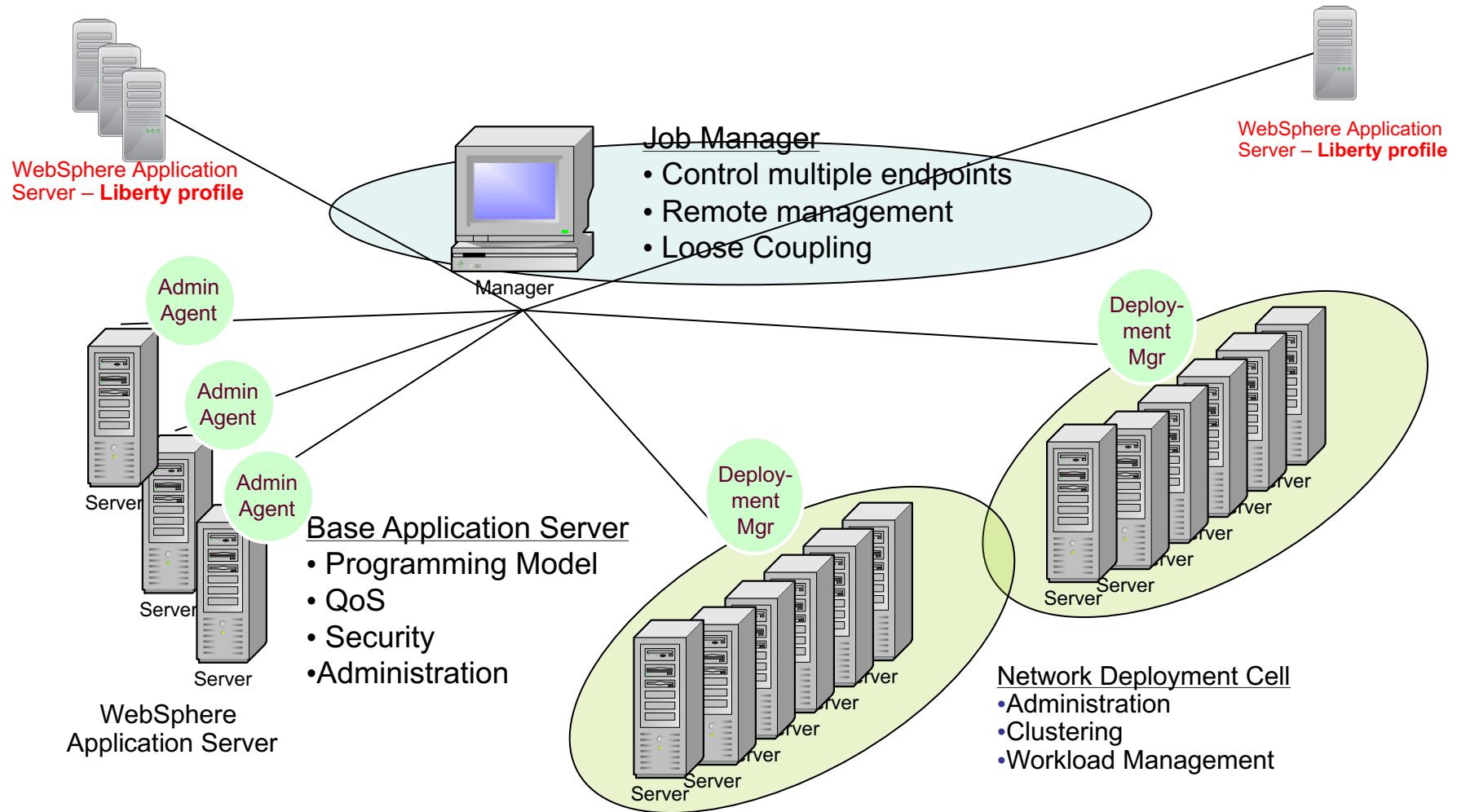
Repackage everything,
or subset needed by server



What This Means For Production

- An “Embedded Server” profile is a production instance of the configured Liberty server type
 - Think of zipping up the application, configuration and server type you just tested on
 - Application centric – the server is pre-configured for a specific application(s)
- **Deployment options:**
 - Unmanaged unzip install
 - Managed ND Job Manager creates “Liberty deployments” by distributing and unzipping the embedded server package
 - “Light-touch” ND management: start and stop server
 - Server configuration remains via the same simplified XML config created in the development environment

Centralized Management through Job Manager



LTA main features

- Light Touch Administration (LTA) of Liberty Servers using Job Manager is a new feature of WebSphere® Application Server V8.5. It's also supports IBM zSeries.
- Central administration through job manager's remote target jobs
 - You can submit job manager jobs that support the full life cycle of Liberty profile resource deployment from initial install, to updates, to uninstall.
 - A deployment manager is not required, although you can use the job manager function available on a deployment manager to administer Liberty profile servers and their resources.
- Quick installation
 - Extract the Liberty profile resource and run the install Liberty profile resource job.
 - Use of Liberty profile resources requires no formal installation by a tool such as Installation Manager.
 - All resources are packaged as one or more compressed .zip files that are ready for use after extraction.
- Flexible sharing
 - You can share a resource such as a software development kit (SDK), runtime binary files, server configuration, and application binary files among many server instances.
 - After resources are deployed to a shared disk, the resources can be shared across computers.
- No agent is required on target hosts, reducing administration overhead.
- Non-destructive update enables easy installation of new versions of any resources. You can switch easily between old and new resources, or run concurrent versions of resources.

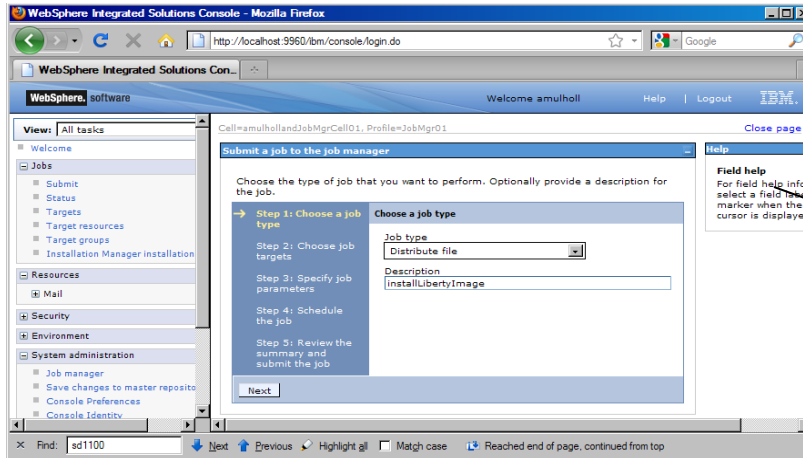
Job Manager Function Summary

- Register target hosts and create groups
- Submit jobs against targets or group :
 - inventory/status: discover what's on the target host and their current status
 - install/uninstall of liberty profile archives
 - start/stop liberty application servers
 - Copy files to/from remote targets
 - Execute commands on remote targets
 - Generate and merge plugin configuration
- Querying and understanding what's in the environment
- Simple server status:
 - Out of the box: status current to last status job
 - Configure serverstatus-1.0 feature if manual start/stop is used
 - Note: status not updated automatically on server crash

Light Touch Admin

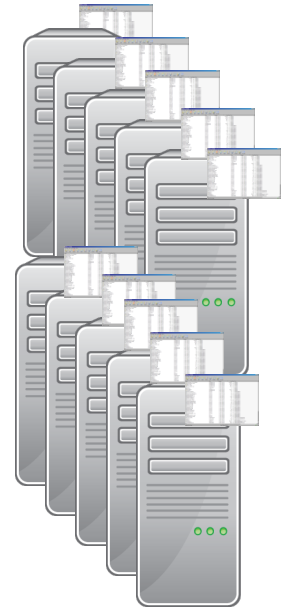
Usage scenarios

Define hosts/groups and deploy embedded server



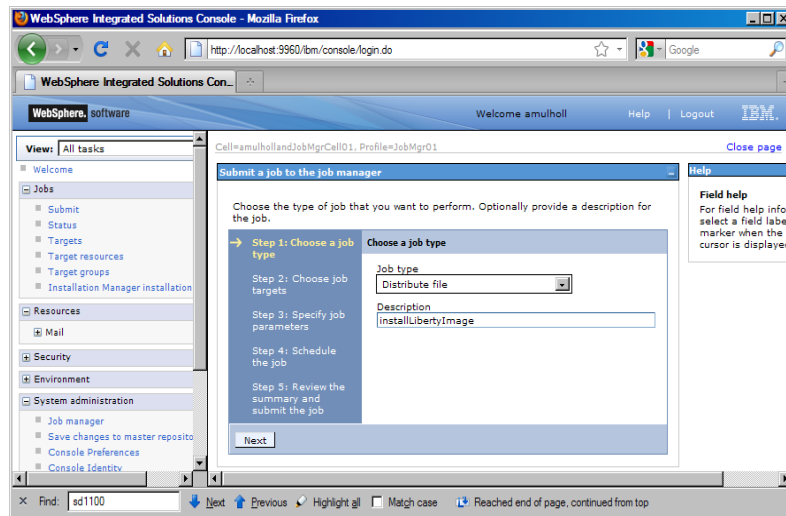
Unzip embedded server image onto remote hosts

Start, stop and view servers from central point



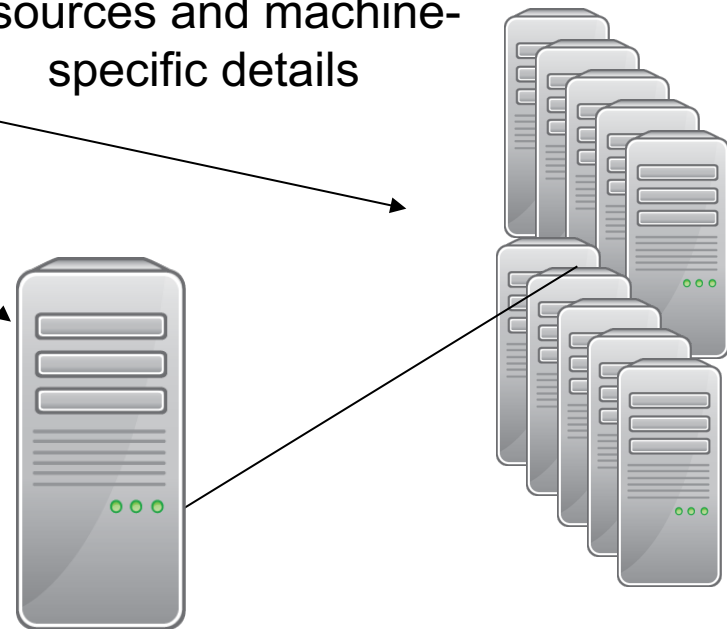
WebSphere ND Job Manager

Options to share JVM, binaries, config, and apps

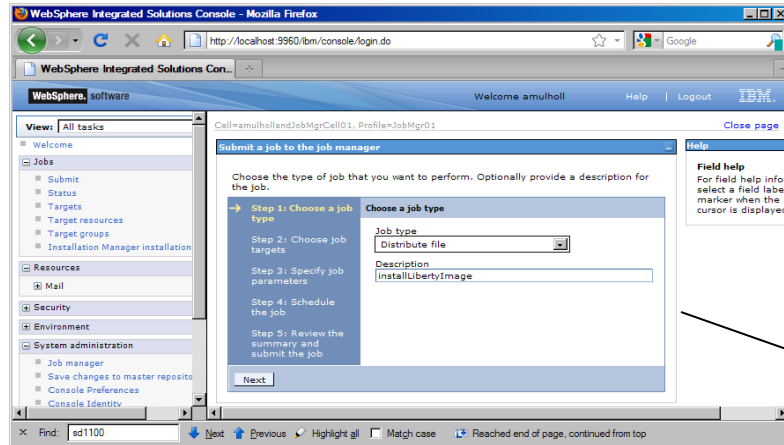


Job Manager manages variables for each host to ensure access to shared resources and machine-specific details

JVM, server runtime, configuration, and applications may all be accessed from shared storage



Generate & merge plugin config for multiple hosts



job type: mergeLibertyPlugin

▶ job target: a group, or list of hosts

▶ job params:

- name: name of the image

▶ Result:

- The plugin for each liberty config at `${LIBERTY_ROOT}/name` is generated, and collected. The final plugin is stored at `${SOME_PLUGIN_ROOT}/name/plugincfg.xml` on jobmgr

WebSphere ND Job Manager

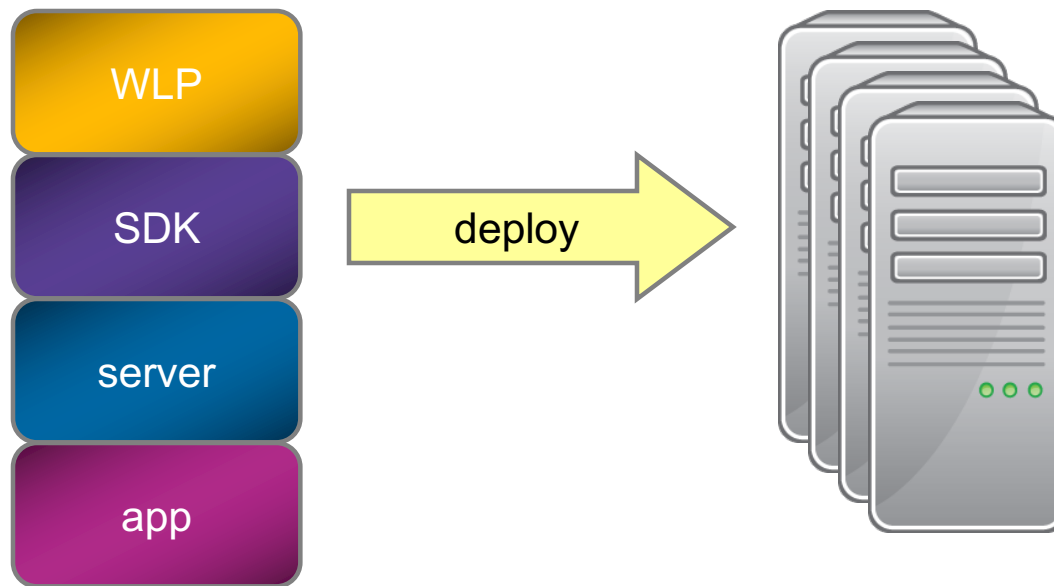


IHS + plugin

HTTP requests

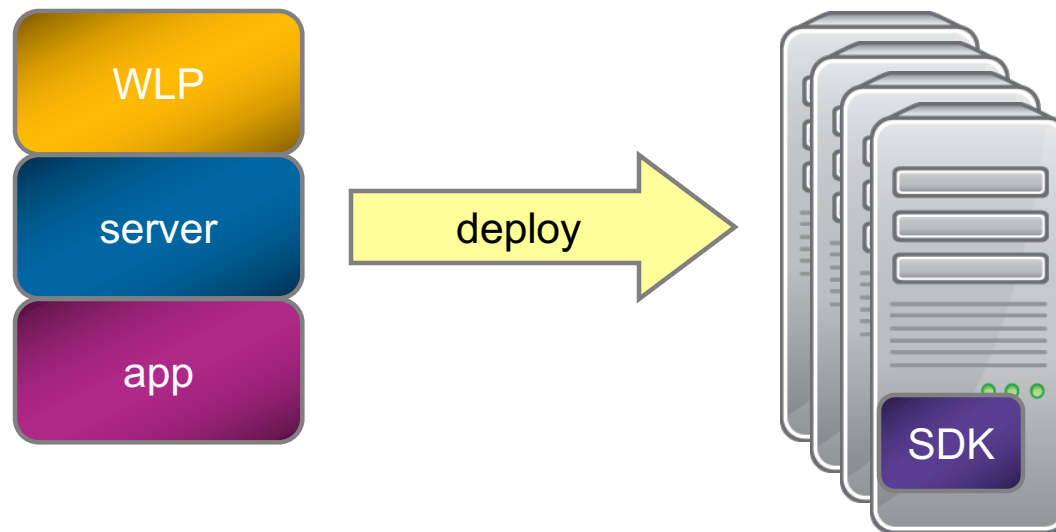


Self contained topology 1



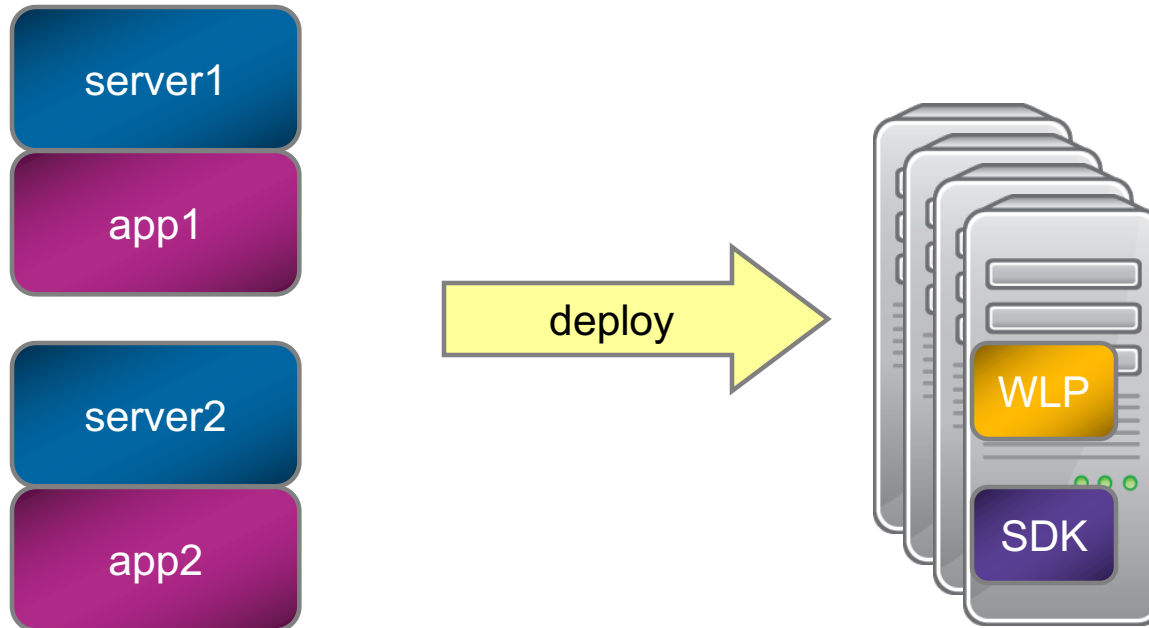
Archive contains everything needed

Self Contained Topology 2



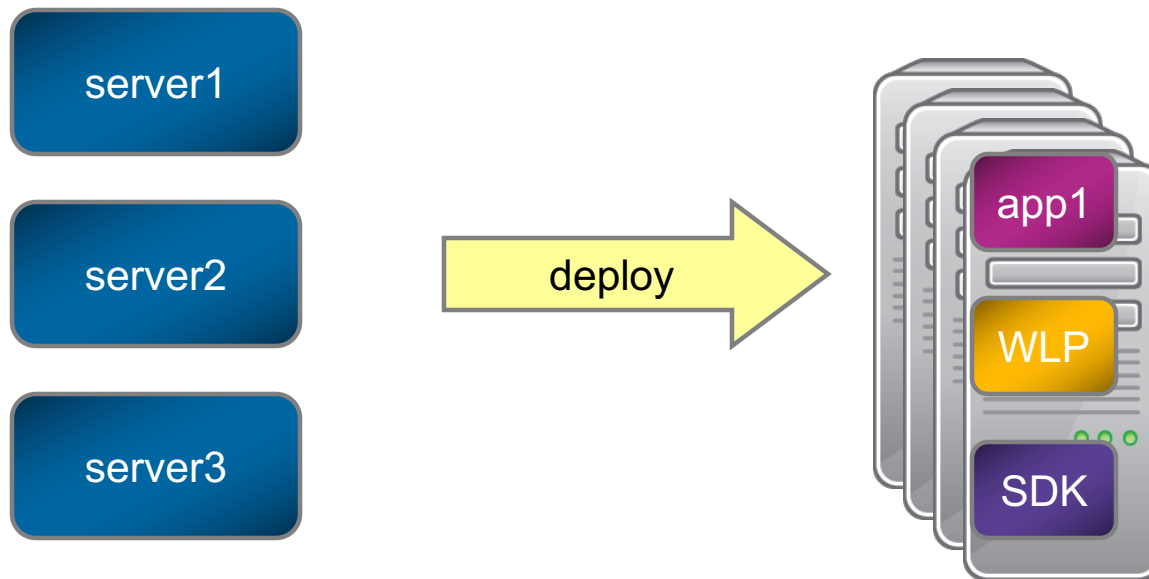
Archive contains everything except Java SDK
Java SDK pre-installed on each host

Shared topology 1



Each archive contains a different server and application
WLP and SDK pre-installed and shared by different servers

Shared Topology 2

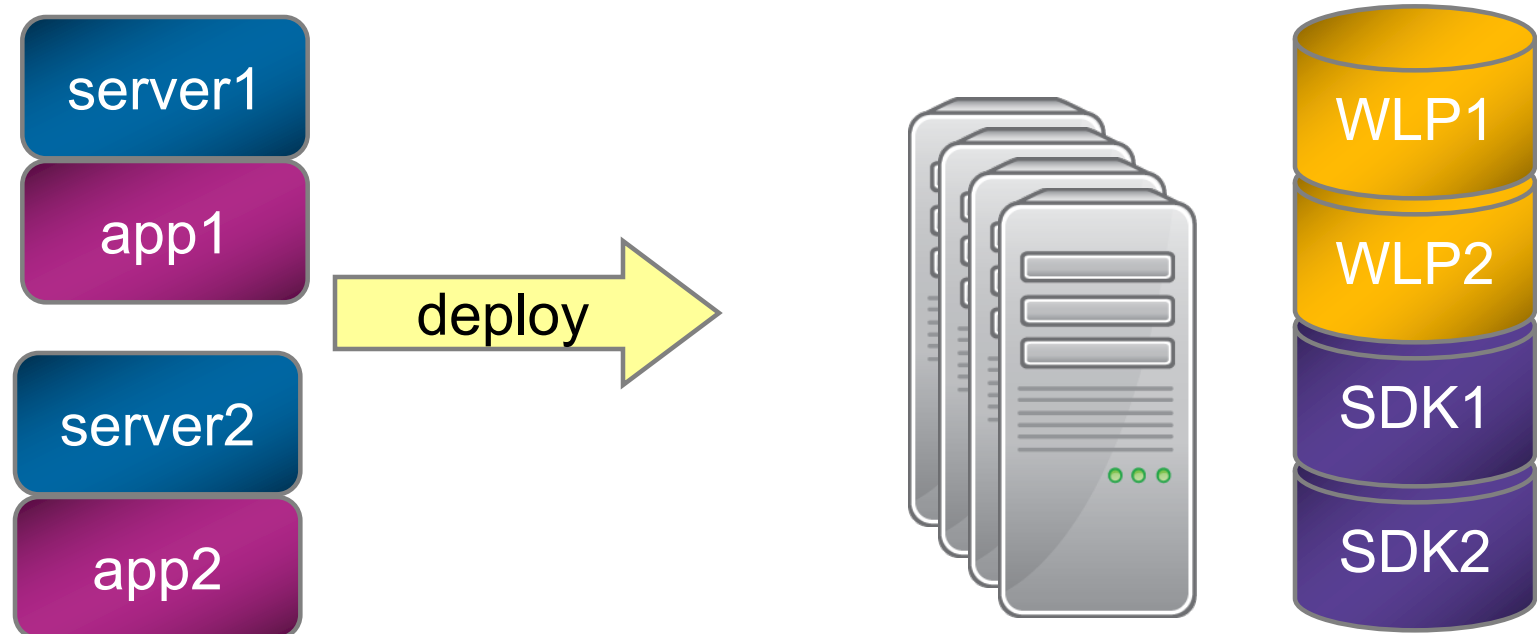


Each archive contains just server definition

Applications pre-deployed read-only, shared by different servers

WLP and SDK pre-installed and shared by different servers

Shared Topology 3: using shared disk

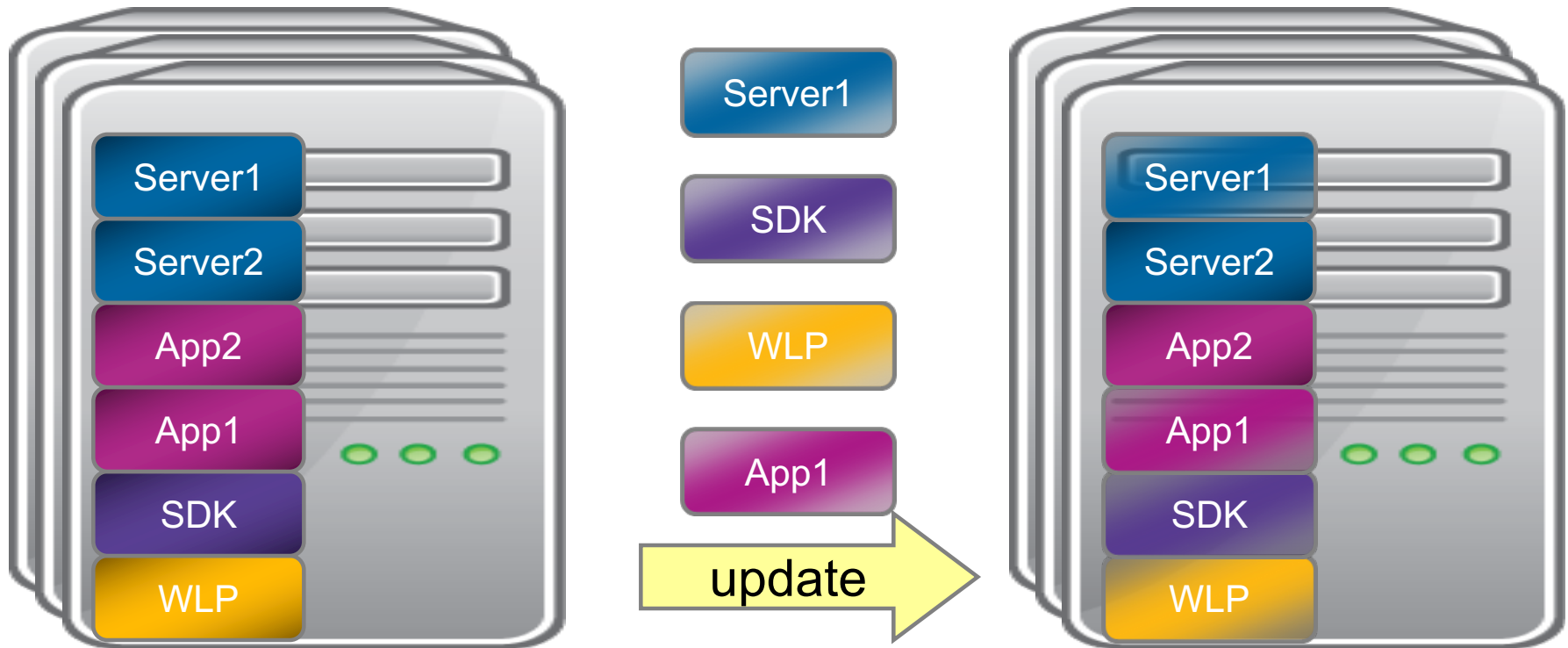


Place shared artifacts on shared disk: one time deployment
Not recommended for production: single point of failure

Update Strategies

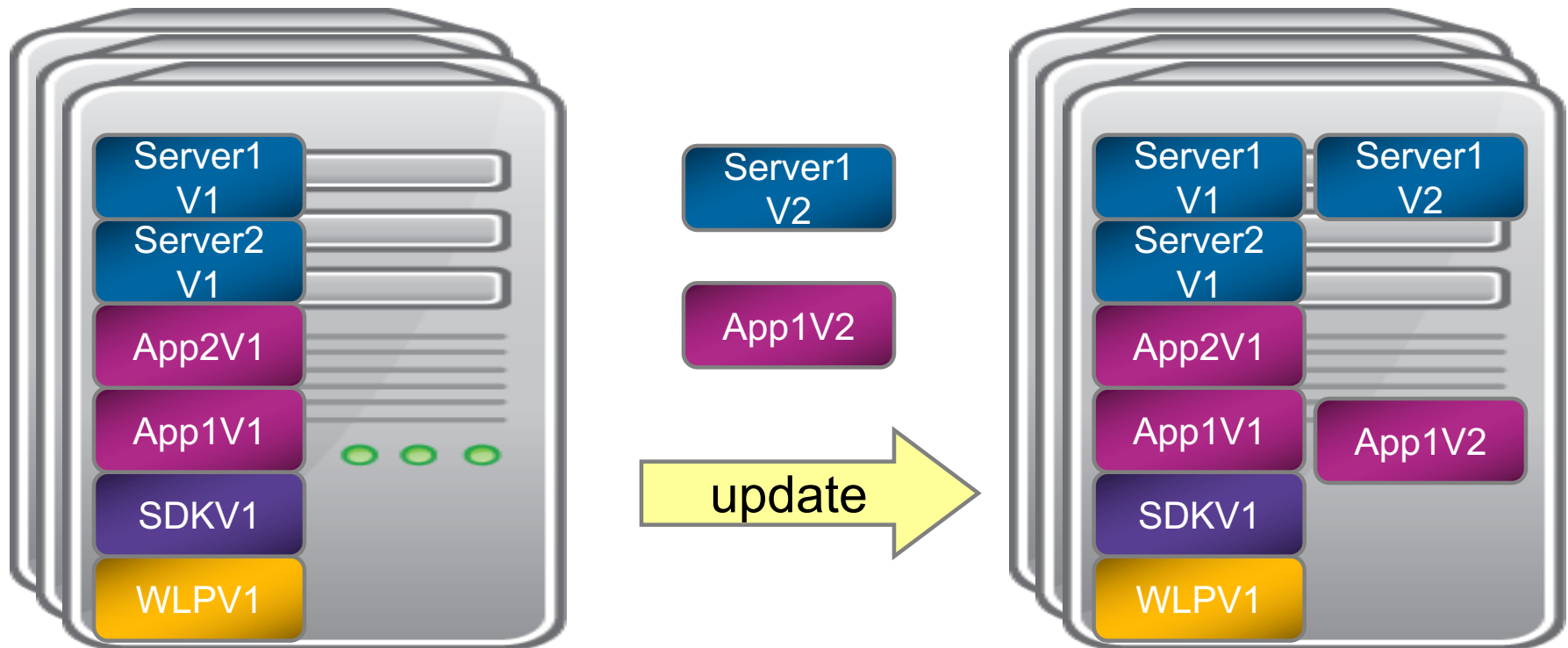
- **Destructive update:**
 - Update in place
 - Less configuration to track
 - Some updates may be global, e.g., updating SDK or WLP installation
- **Non-destructive update:**
 - Manually create new versions
 - Update only what needs to change
 - Can roll back easily

Destructive in-place Update



- Stop all affected servers
- Deploy new configuration with the same name
- Start affected servers
- To rollback, deploy the old configuration

Non-destructive update of WLP and SDK



- Deploy new server and application: Server1V2 and App1V2
- Stop Server1V1 and start Server1V2
- To roll back: stop Server1V2 and start Server1V1

Liberty Security Configuration

Features Available

- Basic, Form, Cert login
- EE Programmatic APIs
 - isUserInRole, getUserPrincipal
 - getRemoteUser, authenticate
 - login, logout
- ④ RunAsRole
- ④ Transport layer security (SSL)
- ④ Registry
 - Basic Registry
 - LDAP Registry (SSL, failOver, referrals)
 - SAF
- ④ WebSphere Authorization
- ④ SAF Authorization
- Basic Single Sign On – LTPA
- Authentication Aliases
- Session security
- JAAS
- TAI
- Relevant Public APIs
 - (wsspi, websphere packages)
- JMX security
- RestConnector security
- MBean security
- Only one administrator role
- Simple password encoding

Configuring Security Features

- **Secure by default in Liberty** (minor different from tWAS)

- **Security component is NOT loaded at startup**
- Security is on by default for administration task
- All security defaults are set appropriately
- Security for applications must be configured to enable

Includes all the security services (authentication, registry, authorization) and web specific security code

- **appSecurity-1.0**

- Includes all the security services (authentication, registry, authorization) and web specific security code

- **zosSecurity-1.0**

- Includes the SAF registry and authorization code

Includes the SAF registry and authorization code

- **ssl-1.0**

- Includes the SSL specific code

Includes the SSL specific code

Liberty administrative security

- One “administrator” role
- One user registry for apps and admin
- Simple configuration for a single admin user

UseCase: As a developer, all I want is one userID and password to test my applications and run admin task.

```
<featureManager>  
    <feature>appSecurity-1.0</feature>  
</featureManager>  
<quickStartSecurity userName="bob" userPassword="{xor}Lz4sLCgwLTs"/>
```

- But still easy for multiple users

```
<administrator-role>  
    <user>fred</user>  
    <group>administratorsGroup</group>  
</administrator-role>
```

User Registries

■ Three types of registry

- Basic XML based registry
- LDAP Registry (Microsoft Active Directory, IBM Lotus Domino, Novell eDirectory, IBM Tivoli Directory Server, Sun Java System Directory Server, Netscape Directory Server, IBM SecureWay Directory Server)
- SAF Registry

UseCase: As a developer, all I want to configure is a basic registry with one or more users and groups.

```
<server>
  <featureManager>
    <feature>appSecurity-1.0</feature>
  </featureManager>
  <basicRegistry realm="basicRealm">
    <user name="bob"
      password="{xor}CDo9Hgw=" />
    <group name="group1">
      <member name="bob"/>
    </group>
  </basicRegistry>
</server>
```

UseCase: I want to use LDAP as my user registry

```
<server>
  <featureManager>
    <feature>appSecurity-1.0</feature>
  </featureManager>

  <ldapRegistry host="ccwin12.austin.ibm.com"
    port="389" baseDN="o=ibm,c=us"
    ldapType="IBM Tivoli Directory Server" />

</server>
```

Enable SSL

- Simple as adding the ssl feature and providing the keyStore password.

```
<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>

<keyStore id="defaultKeyStore" password="{xor}DFoKyp="/>
```

- Certificate generated at startup
- securityUtility can be used to generate a self signed certificate
- Configure per endpoint SSL configuration (Advanced SSL configuration)

```
<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>

<keyStore id="myKeyStore" password="{xor}DFoKyp="
  location="${server.config.dir}/mykeystore.p12"
  type="PKCS12"/>
<keyStore id="myTrustStore" password="{xor}DFoKyp="
  location="${server.config.dir}/mytruststore.p12"
  type="PKCS12"/>
<ssl id="mySSLConfig" keystoreRef="myKeyStore"
  trustStoreRef="myTrustStore"/>

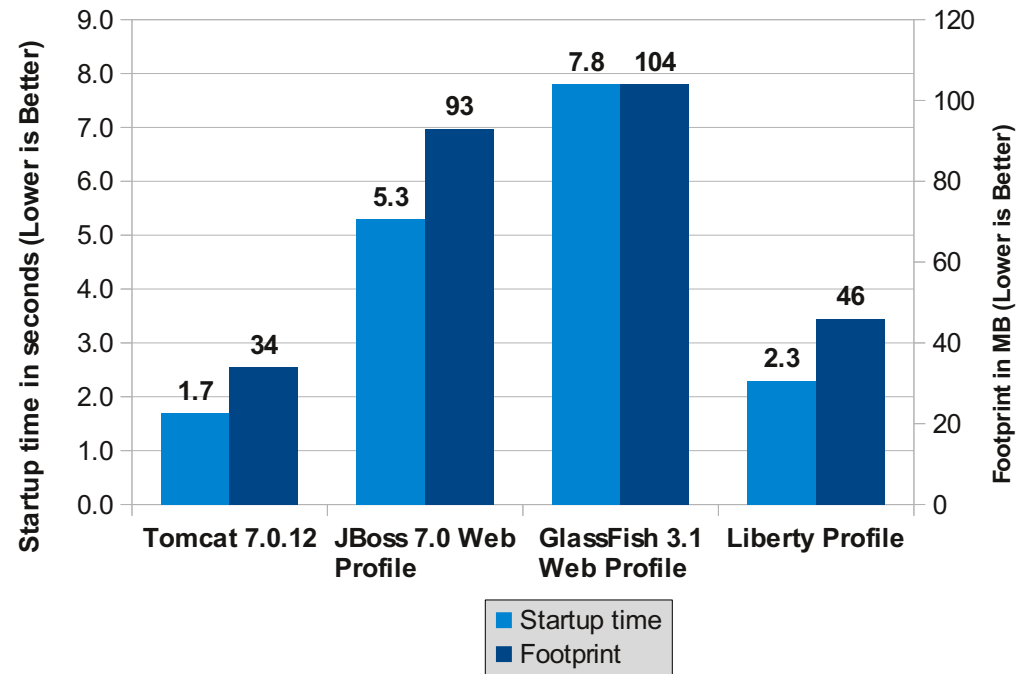
<httpEndpoint id="defaultHttpConfig">
  <sslOptions sslRef="mySSLConfig"/>
</httpEndpoint>
```

Performance

Liberty Profile – Startup & Footprint

- **The problem of a lightweight development environment in WebSphere has been solved!**
 - Liberty Profile startup & footprint are on par with Tomcat.
 - Liberty Profile starts up in less than half the time of JBoss Web profile.

Startup & Footprint Comparison of various lightweight servers



System Info:

Lenovo T60p - 2 x 2.16 GHz Intel Core Duo T2600
2GB RAM, Windows XP 32-bit

Apache Tomcat 7.0.12

JBoss Community Edition 7.0 Web Profile server

GlassFish Server 3.1 Open Source Edition Web Profile

WAS V8.5 Liberty Profile

(All servers had the TradeLite benchmark application installed)

Note: Tomcat , JBoss, and GlassFish were measured with the HotSpot JDK, while Liberty was measured with the IBM JDK.

Liberty Profile – Throughput

- **A lightweight server that can service requests with the speed of a full production server!**
 - Liberty Profile provides up to 20% better runtime performance than JBoss and 25% better than Tomcat.

System Info:

IBM x3550 – 4 x 1.86 GHz Intel Xeon E5320, 8 GB RAM
RedHat Linux 5.3 32-bit

Apache Tomcat 7.0.12

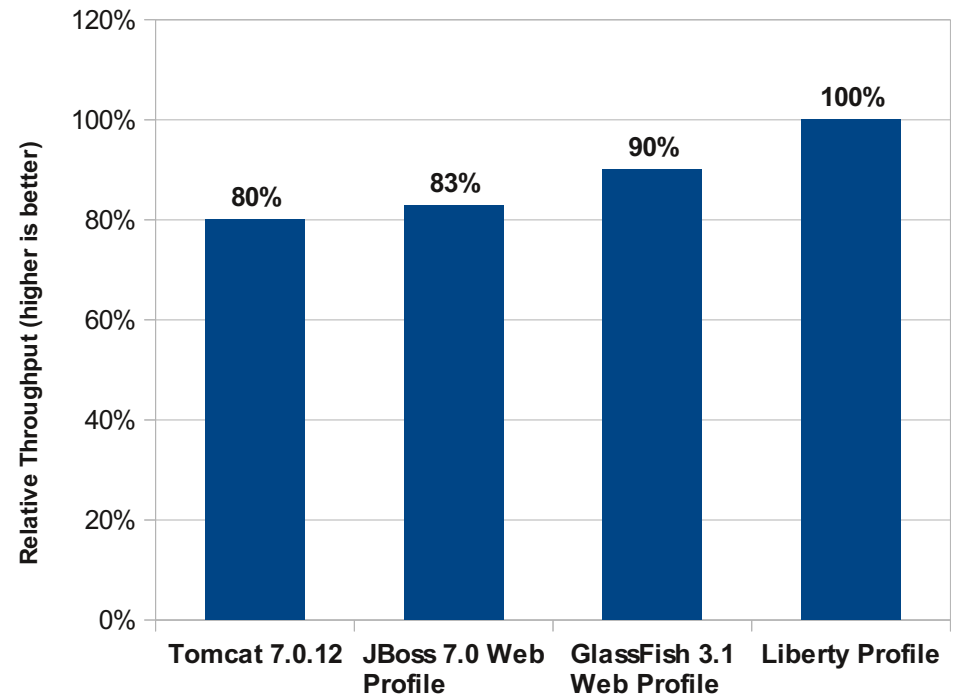
JBoss Community Edition 7.0 Web Profile server

GlassFish Server 3.1 Open Source Edition Web Profile

WAS V8.5 Liberty Profile

(All servers had the TradeLite benchmark application installed)

Throughput Comparison of various lightweight servers




Note: Tomcat , JBoss, and GlassFish were measured with the HotSpot JDK, while Liberty was measured with the IBM JDK.

Elastic Caching with Liberty

- **WebSphere eXtreme Scale and WebSphere DataPower XC10**
 - Integrates seamlessly with Liberty Runtime and its dynamic feature model
 - WXS Container servers can now run on Liberty profile
 - Standalone Liberty Servers can maintain HTTP Session failover and high availability by leveraging an WXS Grid
 - Simple for customer to make use of WXS Use cases!
- **For Developers, easy to develop WXS applications using Liberty within Eclipse**
 - Start up a WXS grid, start up a WXS client, & start up the Liberty server all within a single runtime environment!
- **Additional Tooling for WXS supported for Liberty**
 - Very easy for customers to develop & configure WXS applications right in the Eclipse tool!

Elastic Cache



1 DataPower XC10 for simple data oriented scenarios

2 **WebSphere** software

eXtreme Scale for maximum flexibility



Integration with WebSphere eXtremeScale

```
<server description="catalog server" >
  <featureManager>
    <feature>eXtremeScale.server-1.0</feature>
  </featureManager>
  <xsserver catalogServer="true" />
</server>
```

```
<server description="grid container" >
  <featureManager>
    <feature>eXtremeScale.server-1.0</feature>
  </featureManager>
  <xsserver catalogServerBootstrap="xdblade40:2809,..."/>
</server>
```

```
<server description="web app client" >
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>eXtremeScale.web-1.0</feature>
  </featureManager>
  <xsWebApp catalogHostPort="xdblade40:2809,..."/>
</server>
```

Normal WXS **deployment.xml** and **objectgrid.xml** define the properties of the elastic data grid: in the "grids" subdirectory of each grid container server

HttpSession data is stored in an elastic data grid

Summary: Flexible, Lightweight Profile

- An extremely lightweight WAS profile for web apps
 - Incredibly fast (re)start times: <5 seconds
- Flexible configuration to load only what the application needs
 - Memory footprint (web feature): <60 MB
- Simplified configuration for quick time to productivity; single config file across dev environments (if desired)
- Free & frictionless download for development: including ZIP developer install from WAS for Dev download site
- WebSphere Developer Tools for Eclipse deliver tools for the most popular WAS programming models as Eclipse features.
- Fidelity with full profile WAS server
- Flexibility of development platforms and JDKs
- Simplified configuration will ease migration
- Additional “Packaged Server” deployment model
 - Lightweight production profile based on the needs of the application
 - Smooths transition from development to production
- Unmanaged or managed server instances
 - Unzip and go
 - Centralized management using ND Job Manager; agentless and asynchronous

धन्यवाद
Hindi

多謝
Traditional Chinese

භවතුඹ
Tamil

Dankschen
Austria

Gracias
Spanish

Спасибо
Russian

Köszönöm
Hungarian

Dziekuje
Polish

Engraziel
Switzerland

Thank You
English

Dekuju
Czech

Bedankt
Dutch

Obrigado
Brazilian Portuguese

شكراً
Arabic

Merci
French

Tack
Swedish

Tesekkür ederim
Turkish

Grazie
Italian

多谢
Simplified Chinese

Dakujem
Slovak

Danke
German

תודה
Hebrew

நன்றி
Tamil

ありがとうございました
Japanese

감사합니다
Korean

Reference

- WAS V8.5 information center <http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp>
- WAS DEV website for study and download Liberty <http://wasdev.net>

App-Development Discussion

- 客户在实际开发环境中采用哪些工具？
 - 包括开发工具和应用服务器类型
- 客户中比较典型的开发场景或者拓扑结构有哪些？
- 客户在开发基于WAS的J2EE应用的过程中碰到的主要问题有哪些？
 - 包括开发、测试、部署和迁移时碰到的和WAS相关的问题
- 从Alpha到Beta再到GA，客户是否已经知道Liberty和Websphere Developer Tools V8.5？
- 客户大多采用哪种web service编程模型，jax-ws, jax-rs,还是jax-rpc？

Backup

Centralized Management via Job Manager

- Optionally manage server life cycle through WAS ND Job Manager
- Acts as a single point of management for
 - **Agentless** install using operating system user id
 - Start/stop server instance
- Each instance is a “standalone” server
 - Lifecycle management can be targeted at groups
 - HTTP Session failover
 - DB persistence
 - WXS session cache (replicated)

Liberty VS Tomcat

对比项	Tomcat	Liberty
大小	<20M	<50MB
启动速度	1~3秒	<5秒
Java EE标准支持	JSP 2.2, Servlet 3.0, JSTL 1.2 用户需要添加相关的jar包来实现对其它标准的支持	JSP 2.2, Servlet 3.0, JSF 2.0, JSTL 1.2, JPA 2.0, JDBC 4.0, JAX-RS 1.0, JNDI 1.0, Bean Validation 1.0, SSL 1.0, Security 1.0, Web Security 1.0, JMX 1.0 可动态加载或卸载对应的feature包
OSGi编程模型支持	否	是, 支持Blue Print 1.0, WAB 1.0
开发工具	支持eclipse	支持eclipse, IBM Rational Application Developer
支持的操作系统	Linux, Windows	Linux, Windows, AIX, Mac, HP-UX, Solaris, iSeries, zSeries
性能	一般	与传统WAS具有相同的核心代码, 性能好
应用程序热部署	支持	支持
配置文件	多个配置文件, 更新后需重启服务器	一个配置文件, 更新后及时生效, 无需重启
可移植性	您可以利用IBM Application Migration Toolkit将Tomcat上开发的应用快速移植到WebSphere应用服务器。反之, 则不然。	Liberty和传统WAS对编程模型和标准的支持一致。在Liberty上开发的应用可以直接运行在传统WAS之上。
文档	官方文档涵盖内容有限, 但网络上相关内容(包括: blog, 技术文章等)广泛。	http://wasdev.net 上提供下载地址、文档、视频、音频、样本引用等。
社区	http://tomcat.apache.org 有dev和user mailing list, 没有官方支持, 有时回复不够及时。	http://wasdev.net , 可以与开发人员直接交流
长期战略优势	一般用于开发环境, 生产环境用的比较少。 支持的并发度不高, 可扩展性不强。	Liberty与传统WAS共享同一代码, 天生具有其高效稳定的特点。 .开发-测试-生产切换零代价。 可扩展性高, 支持多个server间的负载均衡和故障恢复。