SHARE Summer 2010
Boston, MA
Wednesday August 4, 2009

WebSphere 101 Application Server Intro & Concepts for Beginners

Michael Stephen
msteff@us.ibm.com
IBM WAS L2 Team Lead

Paul Houde
phoude@us.ibm.com
IBM Advanced Technical Skills

WebSphere Application Server Sessions

Room	Day	Time	Title	Speaker
312	Monday	12:15	Lab	Stephen
203	Monday	4:30	WebSphere: What's New	Follis
203	Wednesday	9:30	WebSphere 101	Houde/Stephen
201	Wednesday	1:30	Introduction to IBM Support Assistant (ISA)	Hutchinson
200	Wednesday	3:00	WebSphere Process Manager and Business Process Manager Configuration	Hutchinson
310	Wednesday	4:30	OSGi/JPA/Batch Feature Packs	Follis/Bagwell
203	Wednesday	6:00	WebSphere for z/OS: I'm no longer a dummy but…	Bagwell
310	Thursday	8:00	WOLA Application Designs	Bagwell
310	Thursday	9:30	Security Architecture: How does WebSphere Play ?	O'Donnell
310	Thursday	11:00	WAS on z/OS High Availability Considerations	Bagwell
200	Thursday	12:15	Staged Application Development in a WebSphere ND Cluster	Loos
310	Thursday	1:30	WAS on z/OS and WLM Interactions	Folis

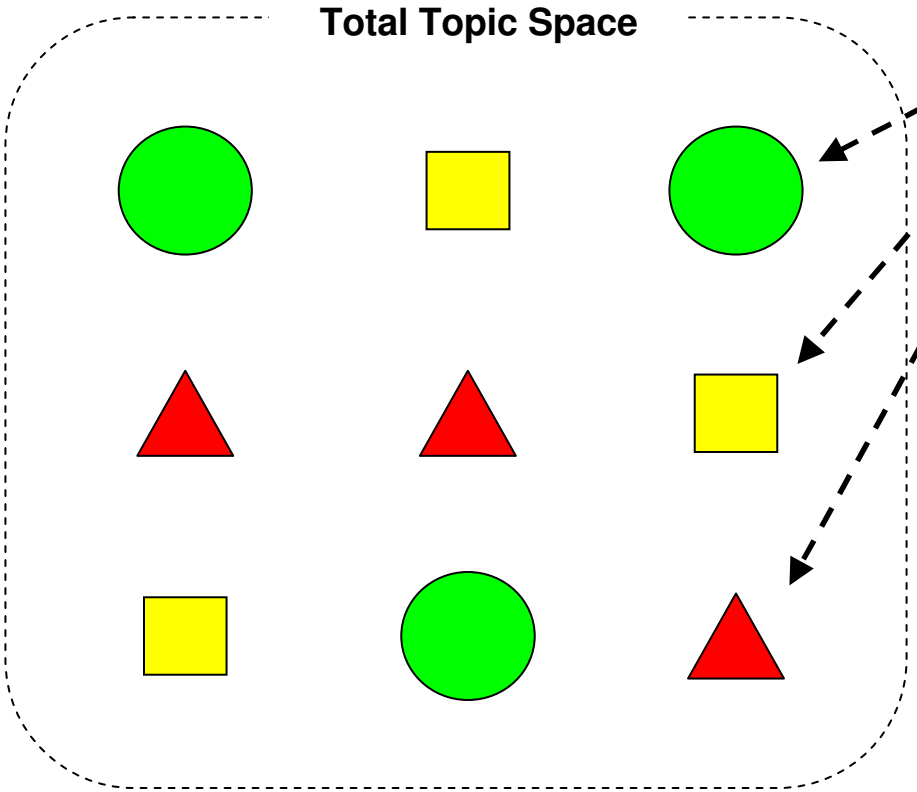
Session Objective and Agenda

Purpose is to explain in plain language what WebSphere Application Server is, what it does, how it works, and why one would consider using it.
Marketing jargon and techno-buzzwords left at the door.

- **Overview of key concepts, terminology and technologies**
  - **What an "Application Server" is**
  - **Java and the role of open standards**
  - **Review of the kinds of applications WebSphere can and can not run**
  - **Review of the notion of a "three tier" architecture -- very common with WebSphere**
- **High level of WebSphere Application Server z/OS**
  - **How it is implemented**
  - **What it looks like from a system perspective**
  - **Cells, nodes, servers, clusters**
- **Overview of installation and configuration**
  - **Understanding the process in simple terms**
  - **Understanding the keys to success -- in a word: planning**
- **Overview of how it's used:**
  - **How applications are installed**
  - **How people access the applications**
  - **How the applications access data**

A Note About Introductory Presentations

In many ways they're more challenging to write and deliver than detailed technical ones:



- **Essential main points**
- **Interesting but peripheral points**
- **Annoying but sometimes important things and exceptions**

Key is to strike the right balance

Be aware that I will sometimes deliberately not mention something because I'm seeking to inform without overwhelming you.

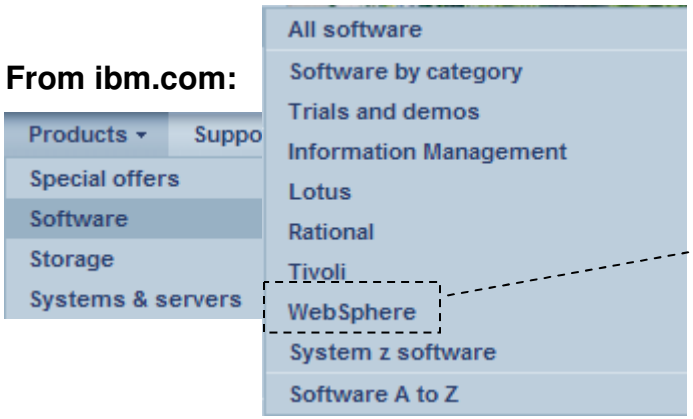
I may also ask that certain questions be taken "offline"

Key Concepts, Terminology and Technologies

"WebSphere" is a Brand; "WebSphere Application Server" a Product

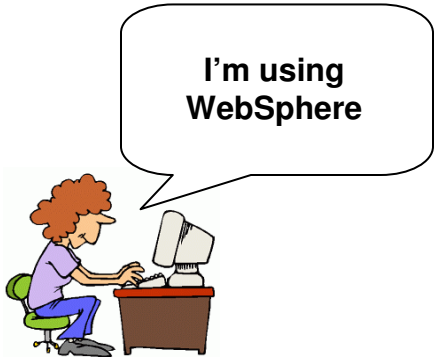
We'll start by clearing up a point of confusion about the term "WebSphere."

From ibm.com:



- **WebSphere Adapters**
- **WebSphere Application Server** — Our Focus
- **Branch Transformation Toolkit for WebSphere Studio**
- **WebSphere Business Events**
- **WebSphere Business Integration Server**
- :
- :
- **WebSphere Virtual Enterprise**
- **WebSphere Voice Response for AIX**
- **WebSphere Voice Server**

Probably close to 100 products that carry the "WebSphere" brand name



For many, the term "WebSphere" is meant to mean "WebSphere Application Server". Sometimes the acronym "WAS" is also used informally.

WebSphere Application Server is now +90%"common code" base for all platforms.

What An "Application Server" Provides

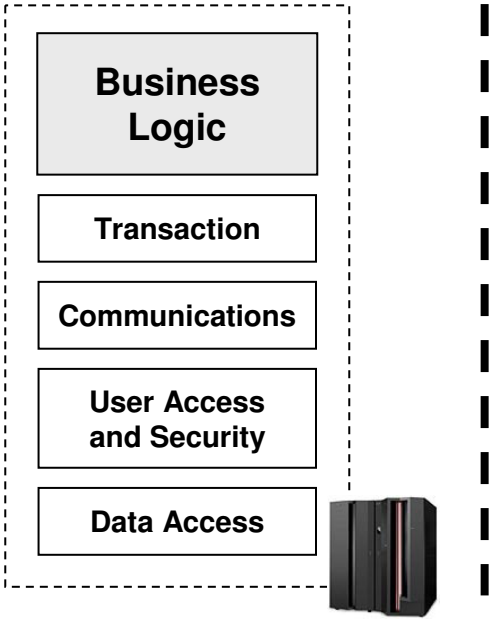
WebSphere Application Server is an "application server" … but what is that?

In the "Old Days"

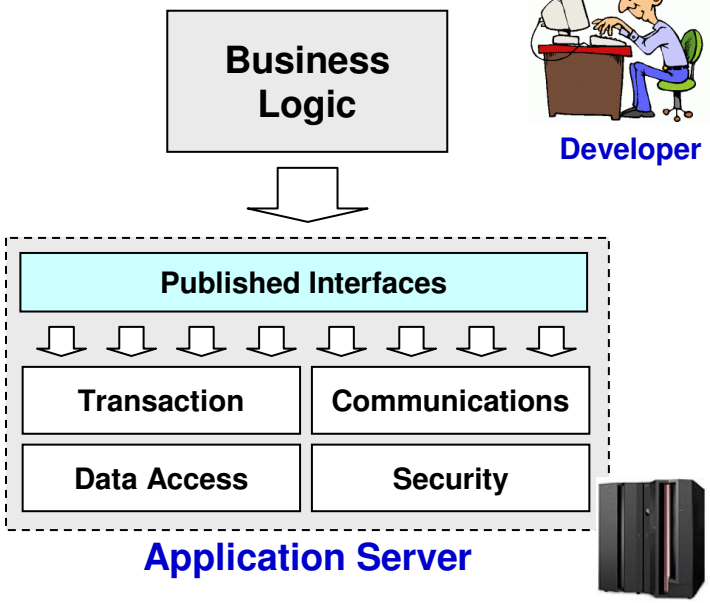
Developer

Had to write it all

Everyone was reinventing wheel over and over again



Nowadays



Purpose is to provide pre-packaged application support stuff so developers can focus on the main business task. No more re-inventing the wheel.

This is not new with WebSphere … IBM had an application server back in 1968!*

So what's the key difference between WebSphere and past application servers?

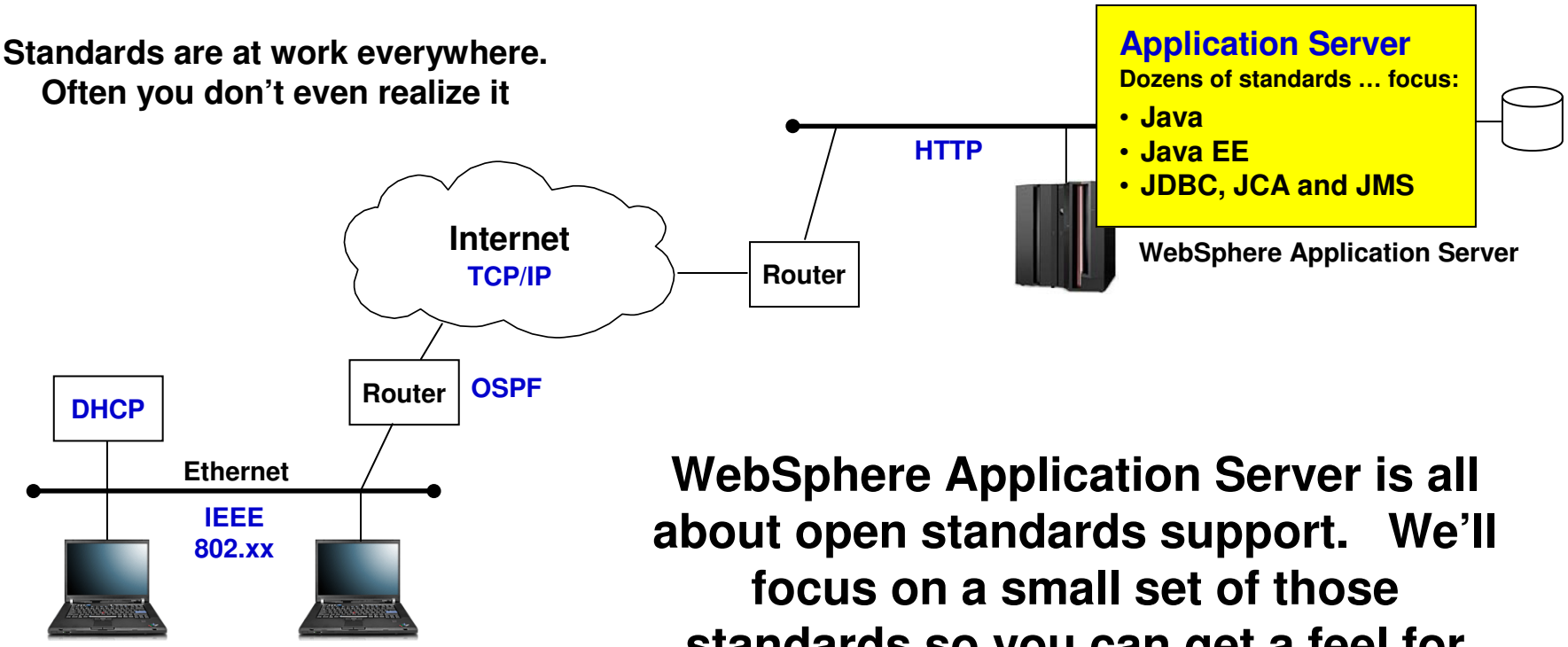
* CICS is an application server

Agreed-to Standards

Rules and regulations agreed to by the players in an industry that will allow the different offerings to work better together.

Otherwise we'd have islands of proprietary technology and difficulty interconnecting

Standards are at work everywhere.
Often you don't even realize it

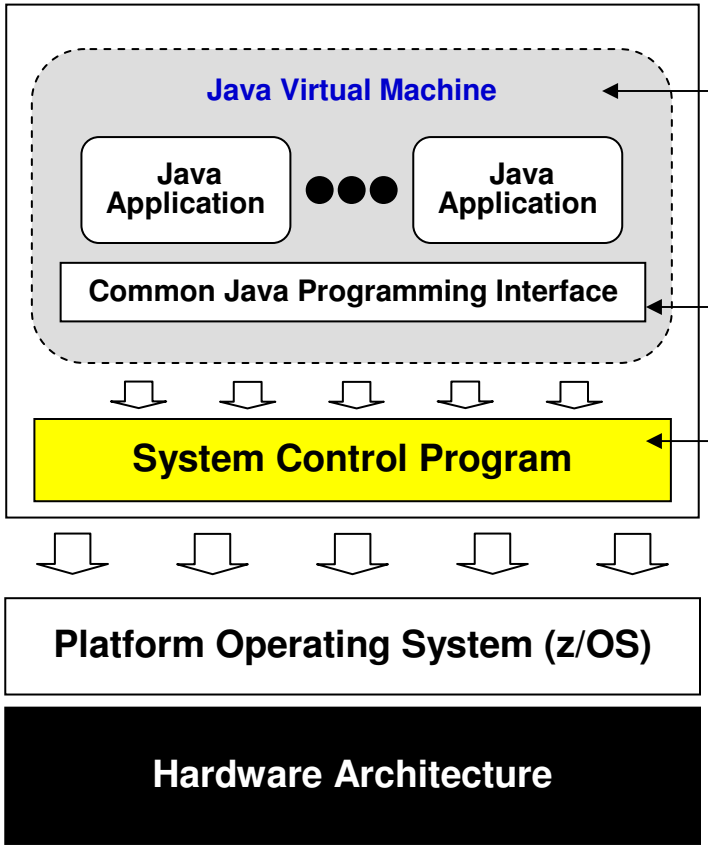


WebSphere Application Server is all about open standards support. We'll focus on a small set of those standards so you can get a feel for the benefit of standards

Java and the Java Virtual Machine (JVM)

Java is a programming language; the JVM is what the Java programs runs in. It's what shields the Java program from the specifics of the platform.

IBM writes and supplies this for its various platforms, z/OS included

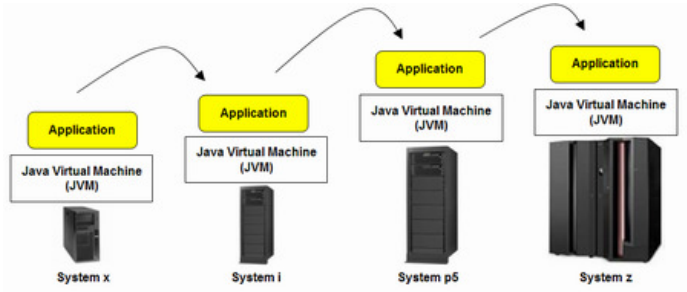


Java applications live in here, happily unaware they're on any specific platform

All JVMs must provide the same API* … that's what makes it "universal"

This provides the translation between Java and the instructions used by the native operating system

Value of all this:



Platform neutrality

WebSphere Application Server contains a JVM
(Several, as a matter of fact)

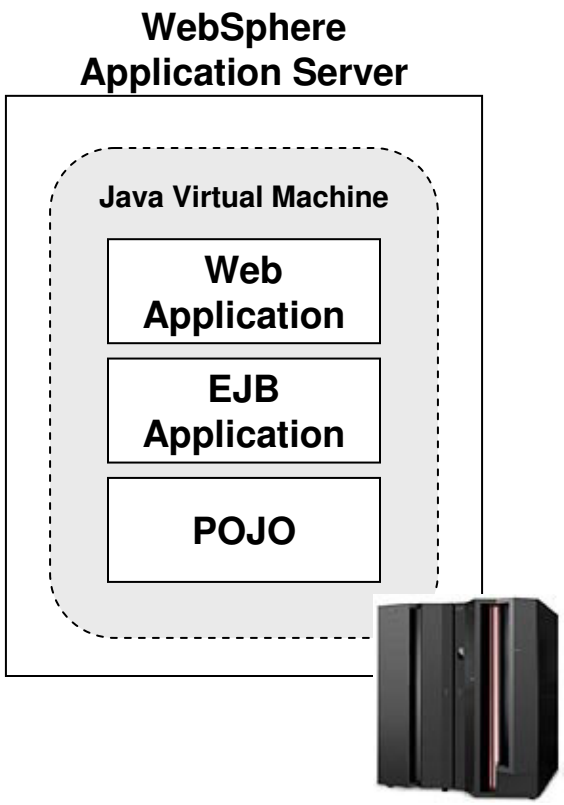
*** For a given specification level of Java**

Different kinds of Java programs …

Different Kinds of Java Programs

WebSphere Application Server can host -- or "run" or "support" -- several different kinds of application -- all written in Java:

It's okay not to understand the details of these things … better at this point just to understand that different kind of programs exist and listen for these terms when others talk about the WebSphere environment.



Web Application

An application that's accessed with a browser. This typically consists of static files (HTML, JPG/GIF), and Java programs that generate dynamic output:

- Servlets -- Java program that contains logic to do things like perform calculations, access data, and format a reply
- JSPs -- stands for Java Server Pages, it's a way to create a dynamic web page that can be populated with dynamic content

EJB Application

Stands for "Enterprise Java Bean," it's a more sophisticated application that's intended for high-end applications. Two flavors:

- Session Beans -- meant to hold the logic of the application
- Entity Beans -- meant to represent data as an "object"

Many EJB applications are made up of just session beans -- easier.

Java EE

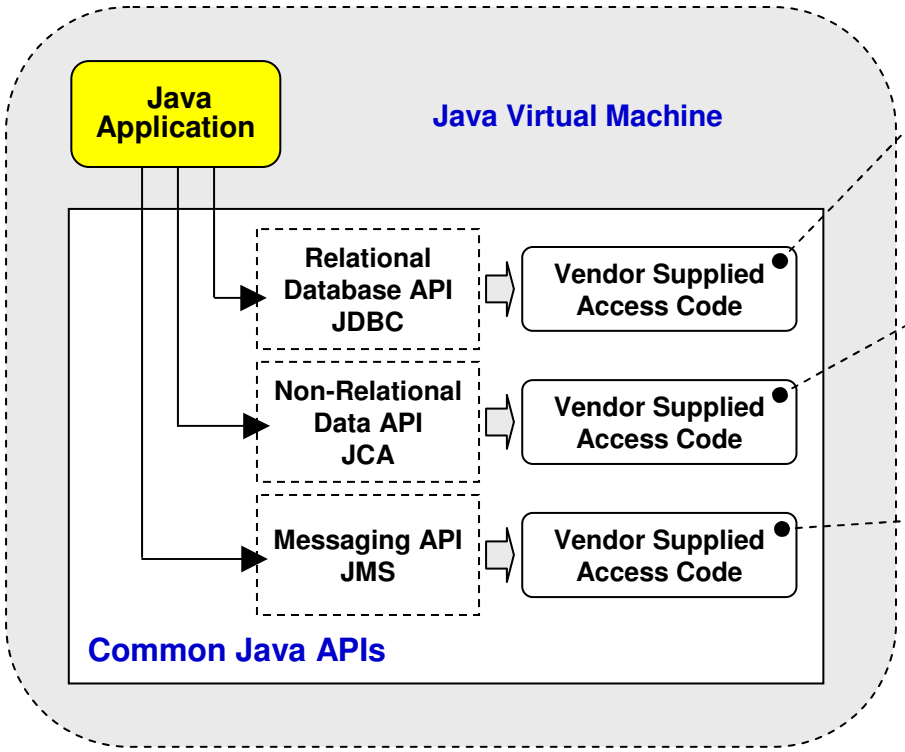
Too simple a categorization, but okay for now

POJO

Stands for "Plain Old Java Object." It is the simplest form of a Java program and lately more people are returning to simplicity.
(POJO commonly applies to the EJB 3.0 environment and Java Batch environment)

Data Connectivity -- JDBC, JCA and JMS

All three are open standard specifications for access different kinds of data. Key is that they provide a defined, standardized interface to "hide" the actual data system where data is held:



Relational Databases

- DB2, Oracle, Informix, Sybase, etc.
- JDBC = Java Data Base Connectivity
- Standard: a defined set of APIs that are converted to the specific commands needed for the backend DB by the vendor supplied code

Non-Relational Databases

- CICS, IMS, SAP, PeopleSoft, etc.
- JCA = Java Connector Architecture
- Standard: a defined set of APIs that are converted to the specific commands needed for the backend data store by the vendor supplied code

Messaging Providers

- MQ as provider, or WebSphere built-in messaging
- JMS = Java Messaging Service
- Standard: a defined set of APIs for applications to send and receive messaging. The actual carrier of the messages can be MQ or the built-in system of WebSphere.

There is more to this when configuring and using it … but this provides the essential point about "hiding" the vendor specifics

WOLA WebSphere Optimized Local Adapters

- The WebSphere Application Server for z/OS "Optimized Local Adapters" provides a high-speed cross memory exchange mechanism between Java applications in WAS and non-Java programs running outside WAS

- Service Level 7.0.0.4 provided support for external address spaces: Batch, Unix Systems Services, CICS and ALCS

- Service Level 7.0.0.12 (GA 7/30/10) added code to provide connectivity to IMS
  via the standard BBOA1* WOLA APIs, as well as the Java Connector Architecture (JCA) resource adapter archive (RAR) provided with WOLA.

- White Paper:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490

- Session: **WOLA Application Designs**

  Thursday 8:00 Room 310 WOLA Application Designs

Typical "Three-Tier" Application Design

We offer this just to put WebSphere Application Server into context with a design concept often talked about and used:

Web Page

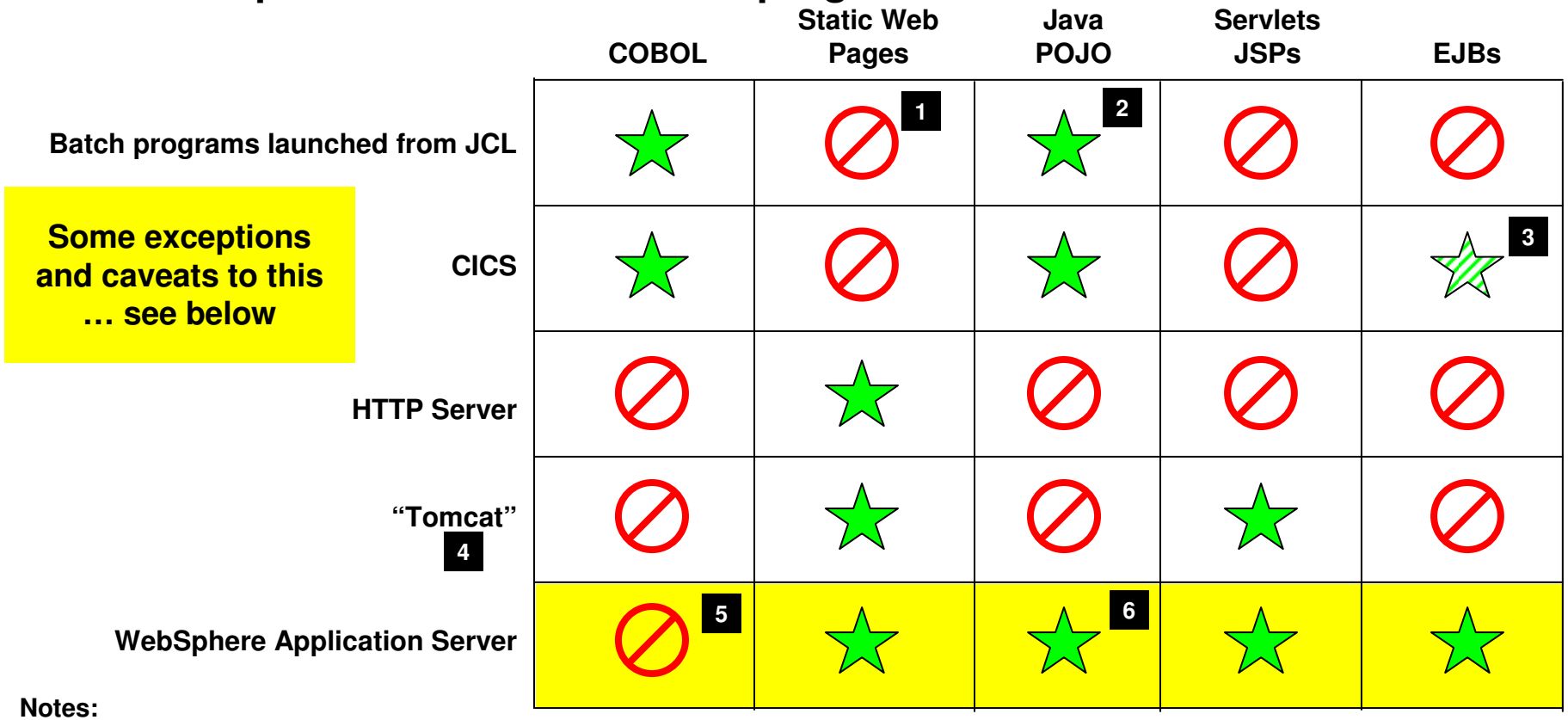
Presentation Tier	Application Logic Tier	Data Tier
Creation of the HTML used at the browser to display result to user	Where the business logic resides. This is what does the work.	Where the data resides -- DB2, CICS, IMS, MQ, etc.

All three *can* exist on the same platform -- logical three, physical one:

Much more to talk about … but this paints the basic picture

Comparing the Different Places Programs Can Run on z/OS

Let's take a quick look at what kind of programs can run:



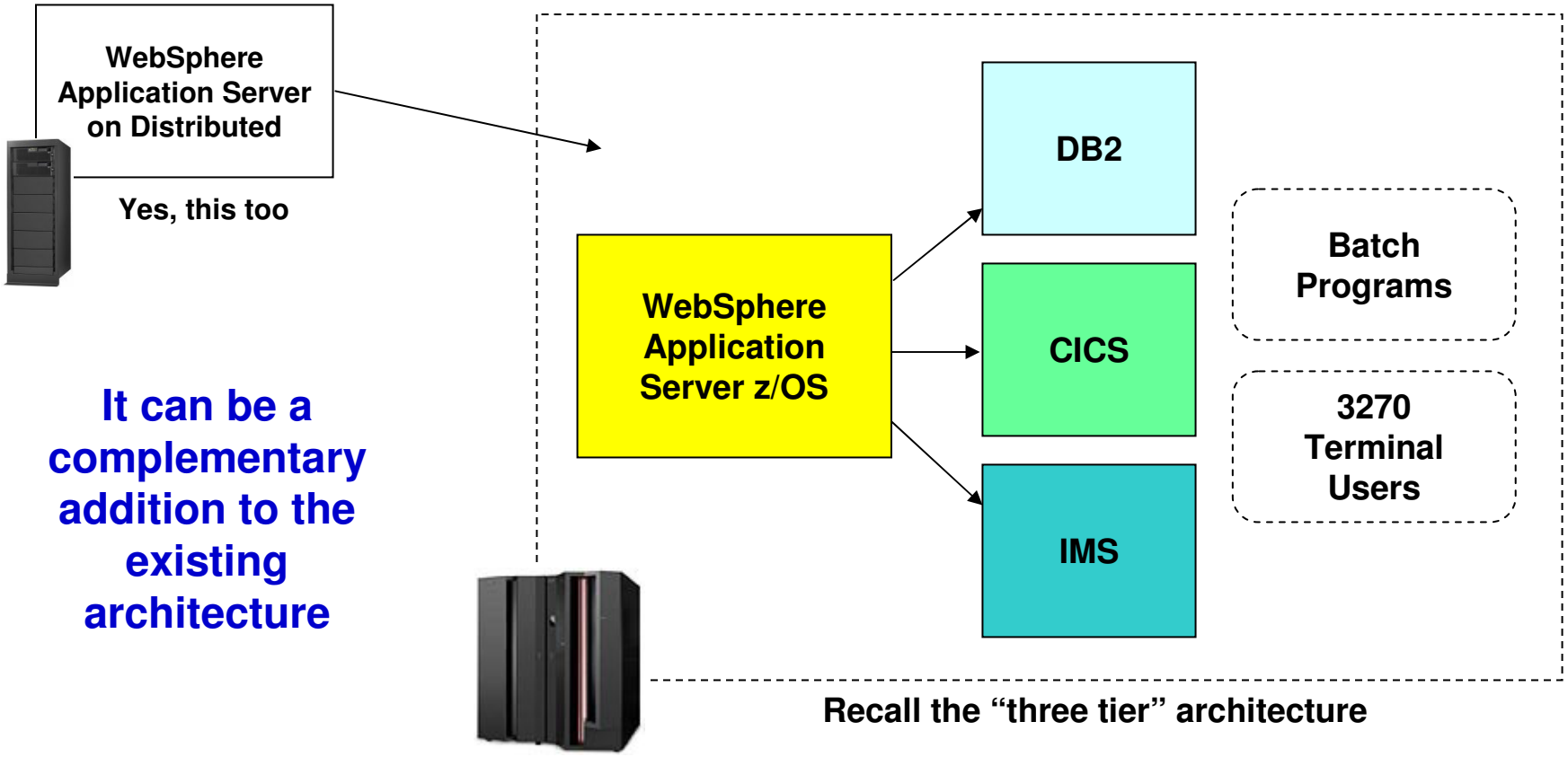
Notes:

1. Is possible to use JCL to launch a HTTP server, but the point here is that batch JCL can't *itself* provide a web server
2. Either BPXBATCH direct invocation of JVM, or the use of something like JZOS
3. CICS supports EJBs, but the specification level supported is quite back-level. In general CICS is not considered the preferred place for EJBs
4. Tomcat is an open-source servlet/JSP engine
5. It is possible to have native code -- C/C++ -- run "in" the WebSphere address space (JNI code). But in general WAS is a Java runtime environment
6. Relatively recent thing in WebSphere

Use in combination …

Use WebSphere in Combination with Other Solutions!

WebSphere Application Server works perfectly well in combination with other traditional systems such as CICS, IMS and DB2:

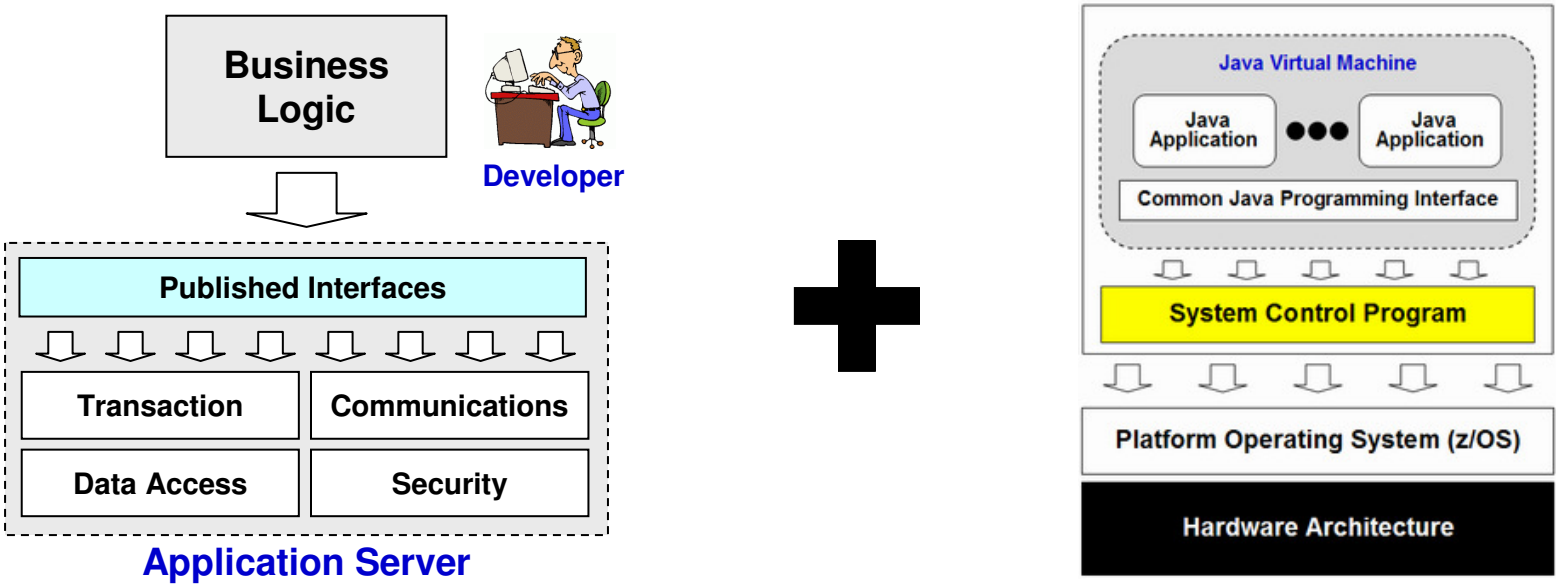


It can be a complementary addition to the existing architecture

High Level Overview of WebSphere Application Server

We Need to Marry Two Key Concepts Together

The idea of a framework that provides common services, and the notion of a Java Virtual Machine:

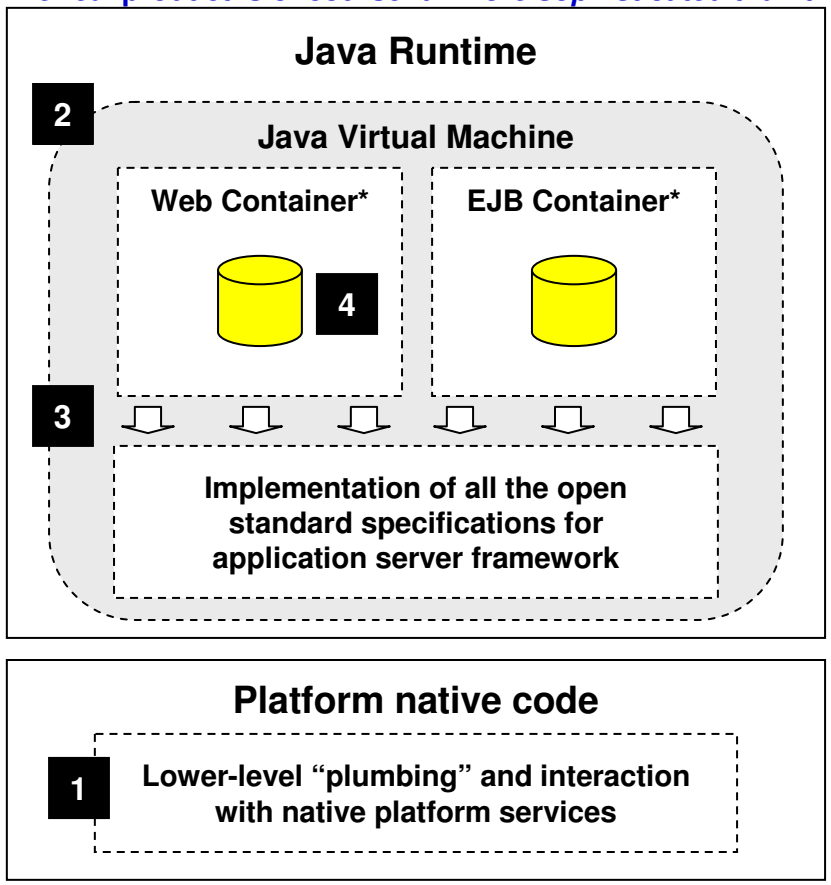


WebSphere Application Server

Schematic Diagram of WebSphere Application Server

Here's a semi-conceptual view of what WebSphere Application Server is:

The real product is of course *far more sophisticated* than this … but this gets the key points across



* "Containers" are just logical software constructs inside the JVM that provide services specific to the type of application that runs in them. "Web Container" is for web applications; "EJB Container" is for EJBs.

1. Server is Started

- On z/OS that's done with a START command (more later)
- This native code is what establishes the lower-level "plumbing" and allows for the invocation of the Java environment

2. Java Runtime Established, including JVM

- Once the native base is ready, it establishes Java runtime environment and launches the JVM

3. WebSphere Java Components Loaded into JVM

- With the JVM launched, WebSphere Application Server can now load the Java components that make up the Java EE environment
- This is the "framework" we mentioned earlier
- This is why WebSphere Application Server is more than just a JVM.

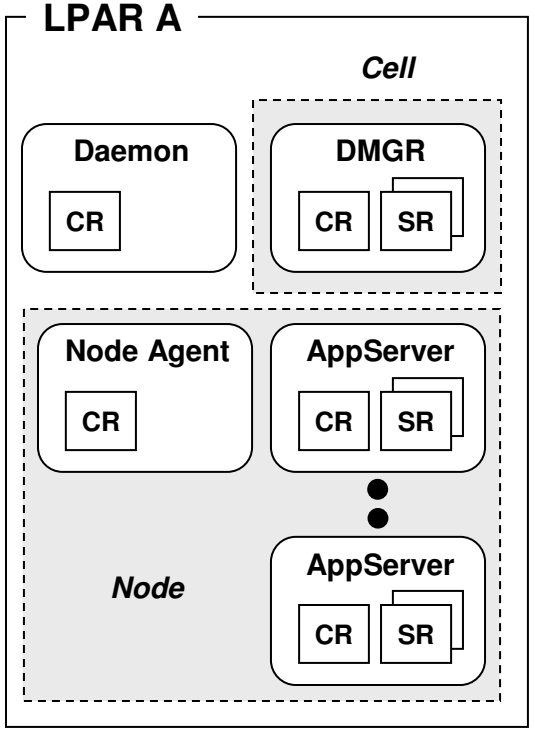
4. Your Applications Loaded and Started

- If they're deployed in the server and configured to start automatically, WebSphere will do that for you.

Now we're ready to see how this is implemented on z/OS

How It's Implemented on z/OS

The developers of WebSphere on z/OS chose to implement the function of WebSphere Application Server as a series of z/OS started tasks:



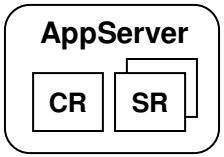
It provides an implementation very familiar to z/OS system administrators

It maps very well to z/OS utilities such as automation and monitoring

But what are those things in the picture?

- **The small boxes inside the curved boxes**
- **CR and SR**
- **DMGR? Node Agent? Node? Cell?**

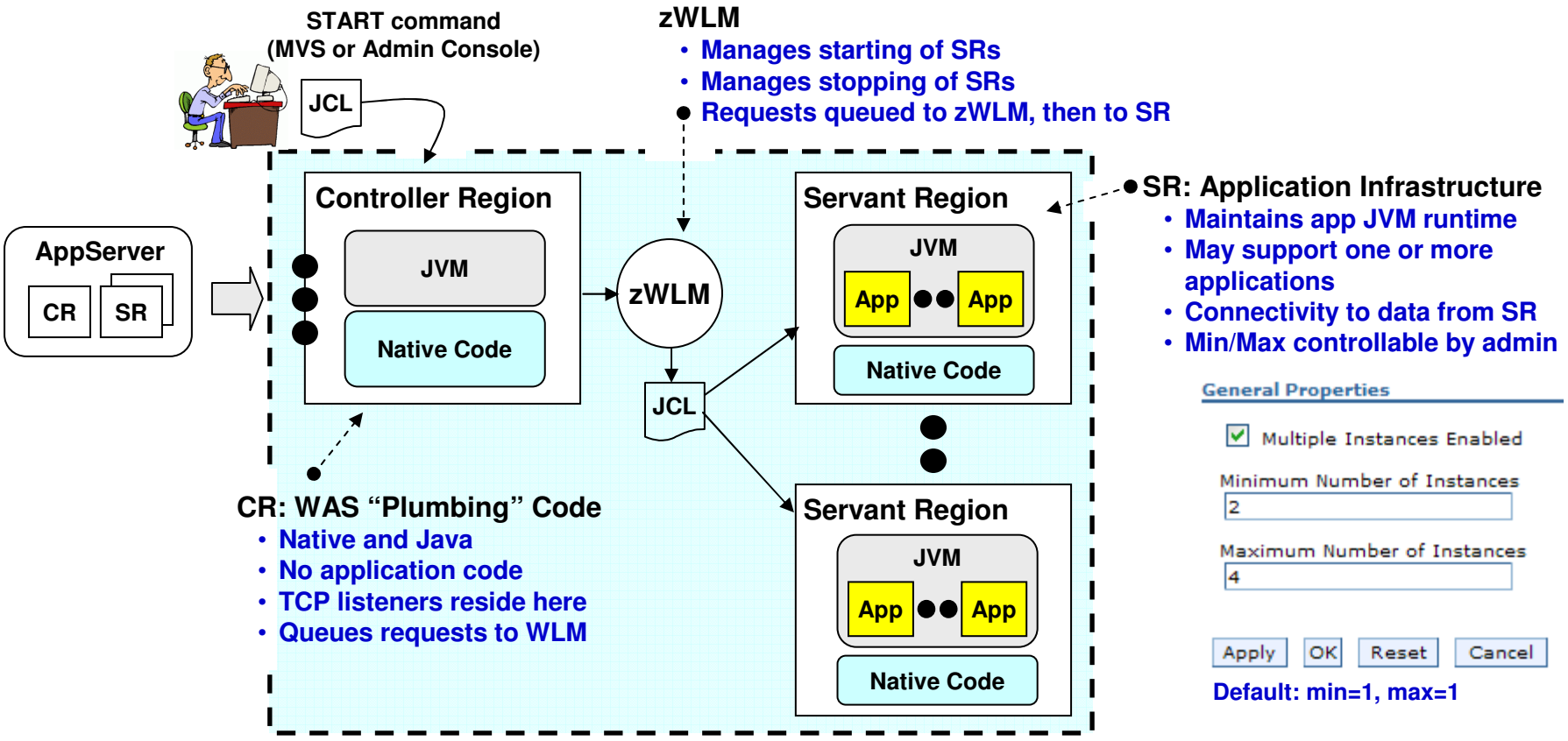
We'll start with the servers where your applications run:



This icon will be used throughout this presentation to represent the "application server," which is where applications run. The small boxes inside are a design unique to z/OS

A Peek Inside the Application Server Architecture

We see that inside our little curved-box picture of the Application Server resides two or more address spaces as well as integration with zWLM:



This is a built-in "vertical scaling" mechanism. Also allows for redundancy of application JVM to prevent single point of failure

Exploits the Strengths of the Platform

The WebSphere code base is not completely the same across all platforms. There is a a degree of unique code … to exploit the underlying platform:

Direct Exploitation

Code is written to realize it's on a given platform and unique code is then invoked.

- **zWLM -- managing servant regions to defined goals; internal routing of IIOP based on environment awareness; classification of workload**
- **RMF and SMF -- reporting on transactions and server components**
- **SAF -- security profile repository, including things such as keyrings and digital certificates**

  -→Thursday 9:30 Room 310 "Security Architecture: How does WebSphere Play"
- **Specialty Engines -- zAAPs and zIIPs / Coupling Facility -- for logging**
- **"Type 2" connectors -- native code implementation; true cross-memory communications**

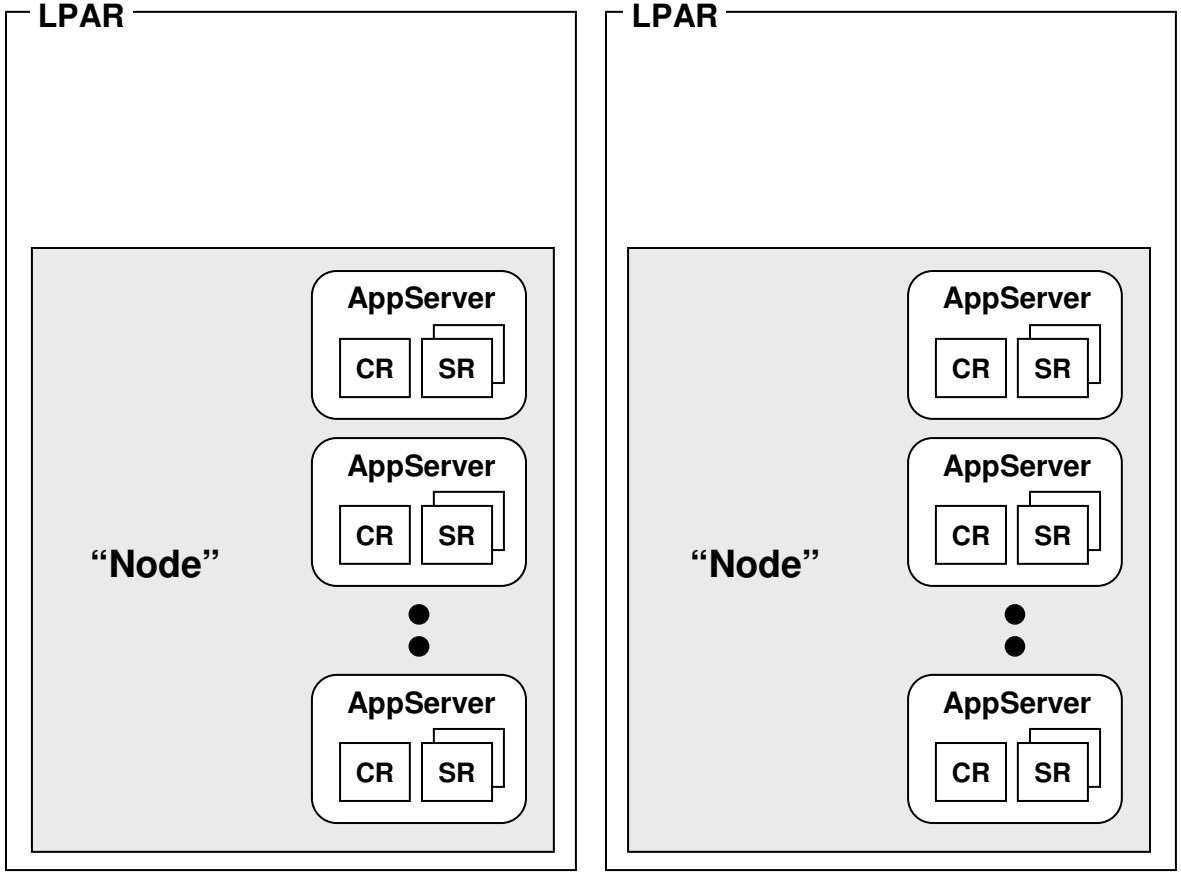
Indirect - Value of "Just Showing Up"

Common code employed, but because it rides on System z and z/OS it benefits

- **Sysplex Distributor and DVIPA support -- for intelligent balancing of traffic (Distributor) and the protection against adapter or TCP stack outage (DVIPA)**
- **Intra-Sysplex or inter-LPAR communications -- XCF (cross coupling facility) or Hypersockets**
- **Sysplex data sharing -- for common data access: DB2, MQ, CICS, IMS**
- **Hardware design -- fault tolerance; redundancy; hot-pluggable, etc.**

Multiple Application Servers and the Concept of a "Node"

There are many reasons* for creating multiple application servers. A "node" is simply the logical collection of applications servers on an LPAR:



Key Points:

- Nodes are a logical thing … it's not a started task
- They logically organize application servers on an LPAR
- No architectural limit to the number of application servers in a node; limited only by system resources
- Rule: node must stay on an LPAR; it can't span LPARs in a Sysplex

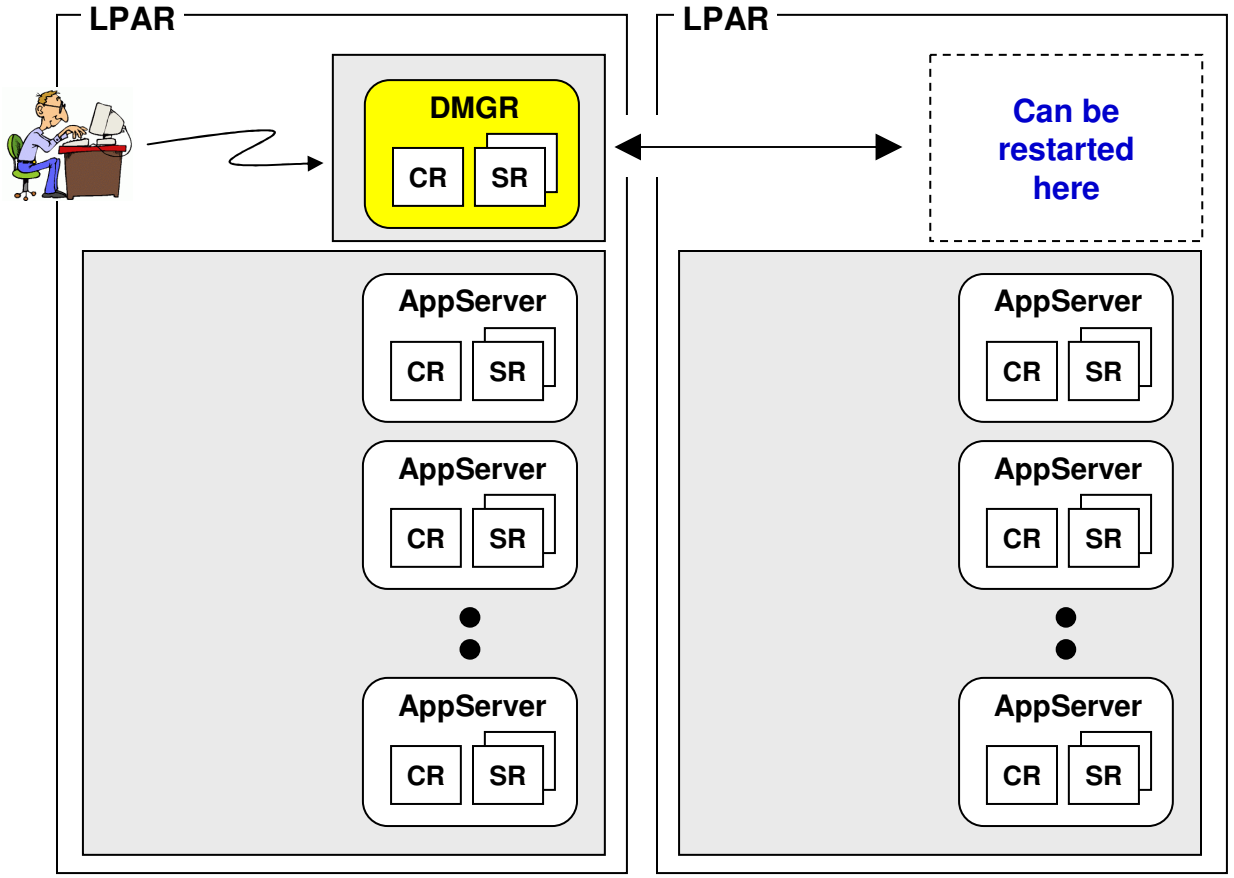
What's the point?

(We'll see in a moment)

* Requirement for separation of application. Applications have different custom JVM settings. Different performance requirements

First -- The Administrative Application Server

There is a special purpose server called the "Deployment Manager" that runs the Administrative Console:



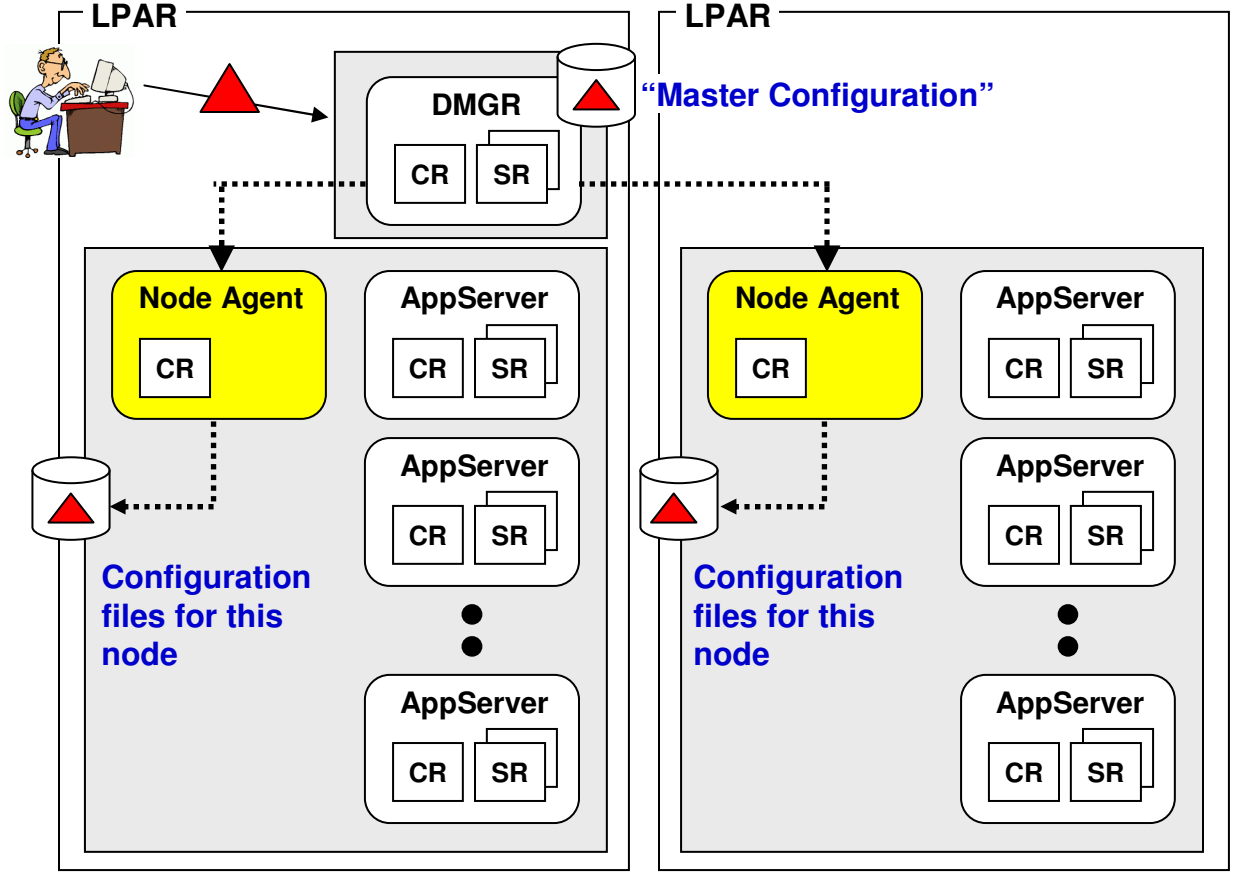
Key Points:

- DMGR structure like application server -- one CR and one or more SRs.
- Only the Administrative Console is allowed to run in this special purpose server.
- The Administrative Console is really just a very smart web app that knows how to translate your configuration mouse clicks into updates to XML configuration docs.
- Properly configured, the DMGR can be started on other LPARs
- Only one DMGR is allowed per "Cell" (which we'll describe soon)

Something is missing …

Node Agents -- Act on Behalf of DMGR in the Node

Node Agents are single-CR structures that update the node's configuration on behalf of the DMGR, which sends updates to the Node Agent:

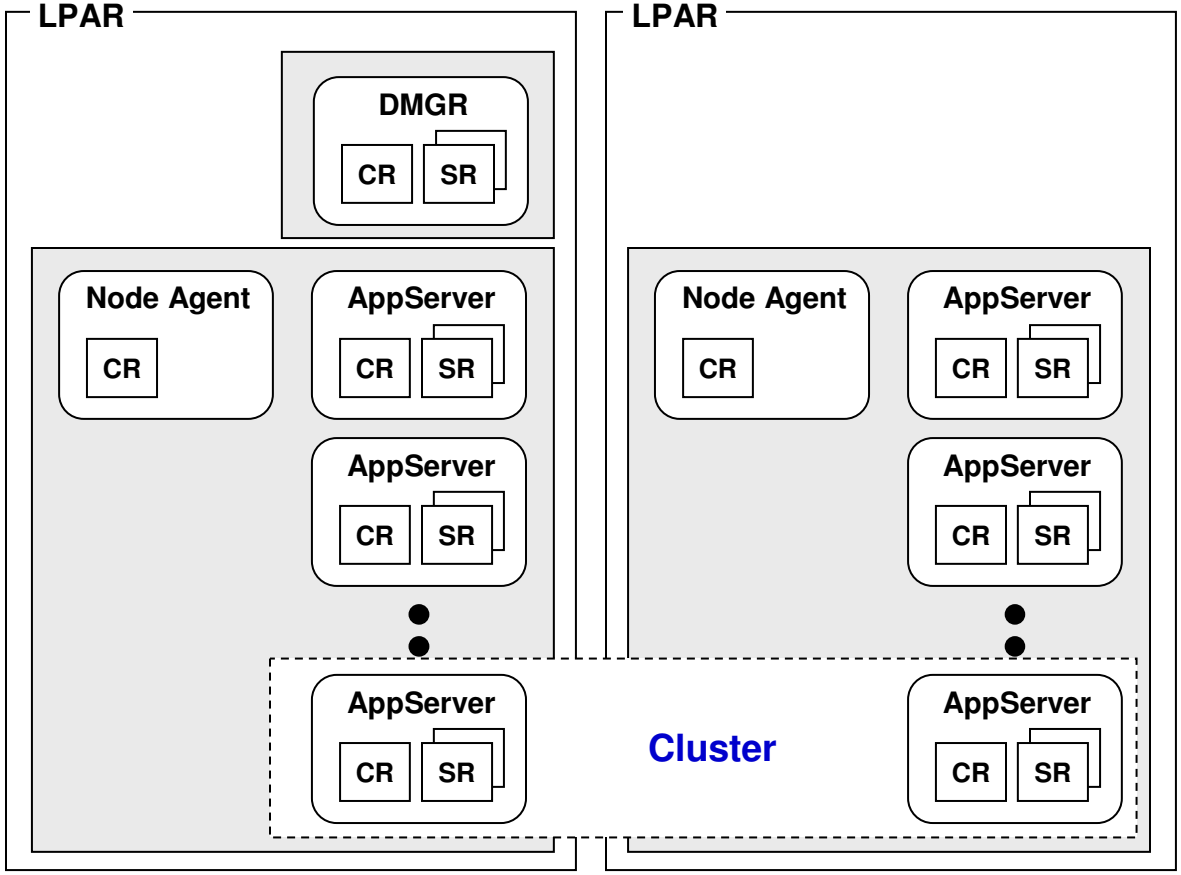


Key Points:

- WebSphere is a distributed architecture -- this allows the configuration to be on separate machines and still work.
- This design frees the DMGR from requiring write access to each node's configuration file system.
- Node Agents are just that -- agents that work on behalf of the DMGR to make the changes in the node.
- Act of copying down changes is called "synchronization"
- Trivia - DMGR maintains master copy of configuration, changes made there first, then copied out to the nodes.

Clusters -- Grouping of Servers to form a Logical One

WebSphere allows you to define multiple servers that acts as a kind of "single logical server". These are clusters:

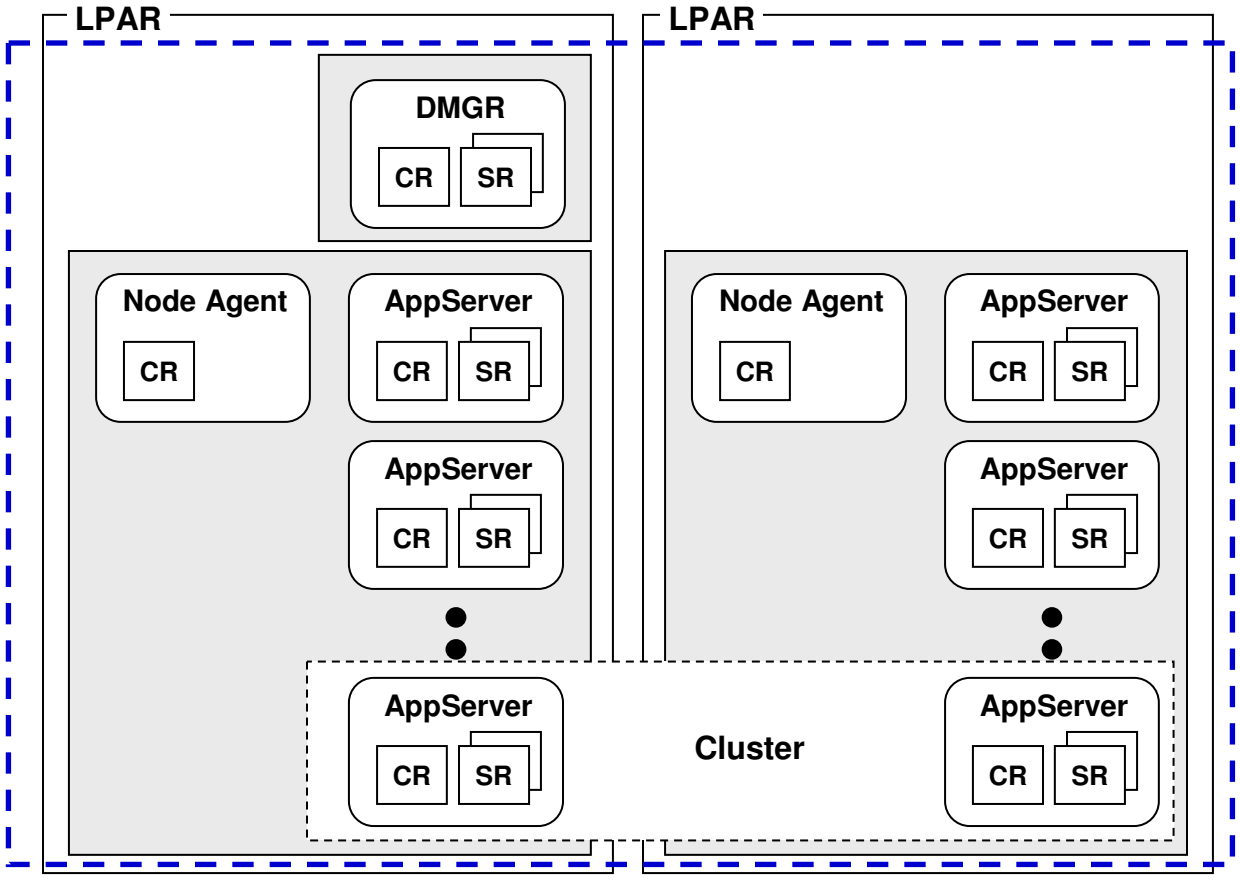


Key Points:

- The application servers are in fact separate servers, but WebSphere treats them as one for application deployment
- These are used in HA configurations when mutliple concurrent copies of an application is desired.
- We are intentionally skipping the topic of "front end load balancing" … interesting topic but too much for this session.

Now We Can Introduce Concept of the "Cell"

The Cell is really nothing more than the extent of administrative control a DMGR has. In this example it controls two nodes on two LPARs … that's the cell.

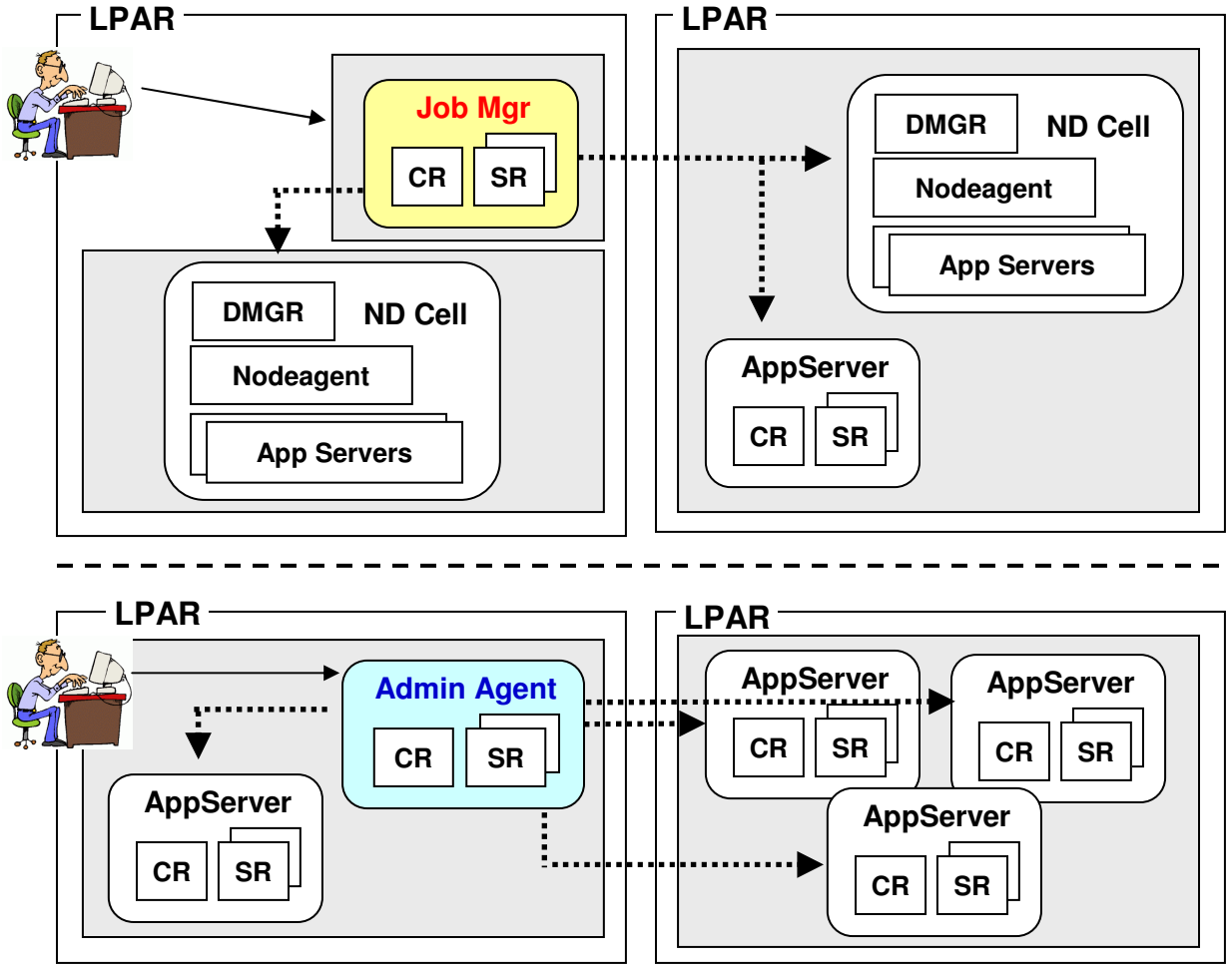


Key Points:

- The Cell is a logical thing … it is not a started task or address space.
- The Cell marks the boundary for administrative isolation … you can limit who has access to modifications to the Cell. This is how QA, Test and Production is best kept separate.

Flexible Management

Additional architectures were introduced in WAS V7 to deal with the administration of large groups of WAS cells



Job Manager:

- Job Manager is a central console for the administration of many cells (ND or stand alone)
- Job Manager differs from DMGR since it does not centrally store configuration and can manage multiple cells
- Asynchronously submit jobs at scheduled times
- Each cell maintains autonomy

Administrative Agent:

- Replaces the administration console for a group of stand alone application servers
- Reduces the footprint of each application server by removing administrative overhead
- Each server maintains autonomy

Answering a few Q's …

Anticipating Some Questions

May I have more than one Cell?

Yes … no limit to the number of cells you can create.

May I have a cell that spans z/OS and distributed servers?

Yes … but start out with z/OS-only until you gain experience. Then move to the more complicated topic of "heterogeneous cells"

Complication comes chiefly from security issues and the coordination of digital certificates, and the creation of an external userid repository such as LDAP.

What about the Daemon Server?

We intentionally skipped over that to keep things simple ☺

Can an application server belong to two cells at the same time?

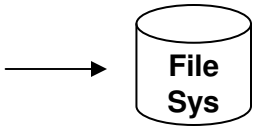
No … overlapping of resources like that is not allowed.

Installation and Configuration

SMP/E Installation of WebSphere Application Server for z/OS 7.0

Relatively straight-forward SMP/E installation:

```
WAS700.WAS.SBBOEXEC
WAS700.WAS.SBBOHFS
WAS700.WAS.SBBOHFS.DATA
WAS700.WAS.SBBOJCL
WAS700.WAS.SBBOMAC
WAS700.WAS.SBBOMSG
```



`/usr/lpp/zWebSphere/V7R0`

Unix Systems Services
File System

And a short list of relatively simple system programmer steps, all well documented

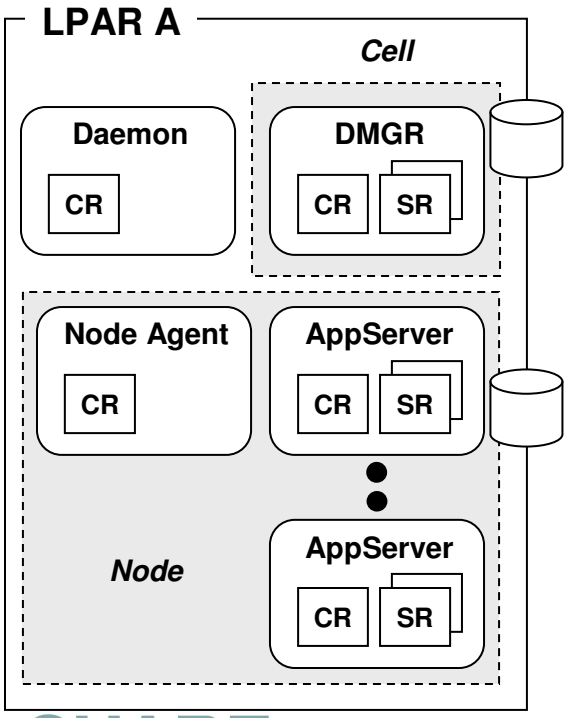
No module libraries … that's different from in the past. Now the entire product is contained within a file system (HFS or zFS)

This is just the product itself … this is *not* your customized configuration. That's a separate set of sets which we'll cover next.

Customization at a Very High Level

The whole objective is to create the configuration information, which is kept in an HFS or ZFS:

Supply names and values so your configuration is customized to your needs ⇨ **Upload generated batch JCL jobs to z/OS and run** ⇨ **Start Servers**




```
/wasv7config/mycell/mydmnode
└─ /DeploymentManager
   └─ /profiles/default
```

Whole bunch of customized XML files

```
/wasv7config/mycell/mynodea
└─ /AppServer
   └─ /profiles/default
```

Whole bunch of customized XML files

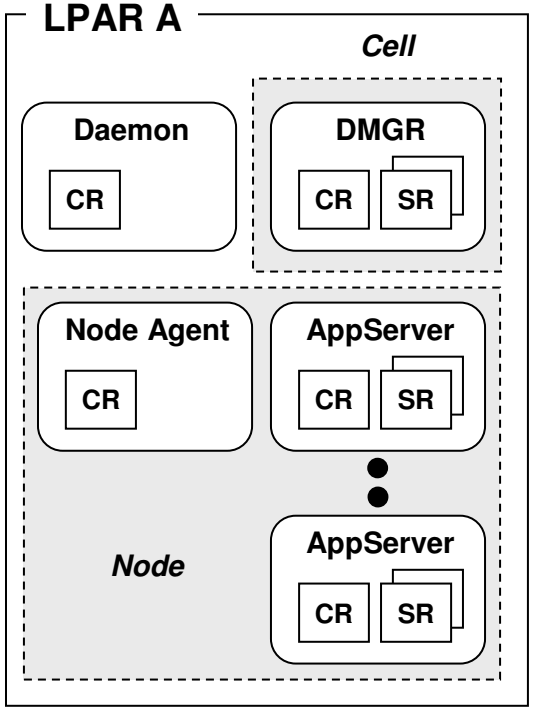
Don't worry what's in those XML files … detail you don't need to know right away

The key is the configuration "tree" needs to get built and populated with your specific information.

That's one piece of the configuration puzzle … what are the other pieces?

Fundamental Pieces of a Customized Configuration

It's helpful to focus on three fundamental things that make up a customized configuration of servers, nodes and a cell:



Configuration File System

- HFS or ZFS, this contains all the XML files that make up the configuration.
- Each node has its own configuration file system

JCL Start Procedures

- This is what is used when starting the servers, Node Agents and Deployment Manager.

SAF Profiles

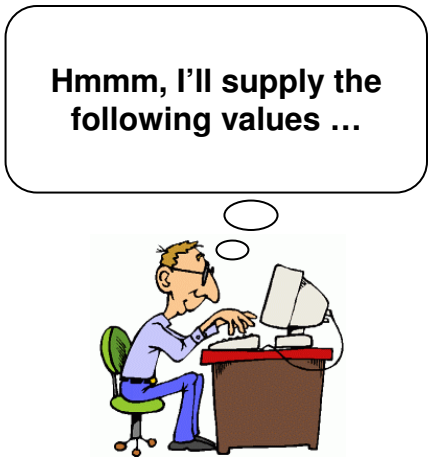
- They are what provides the essential z/OS security for the started tasks -- Userids and Groups for file ownership and administrative access; STARTED profiles for assignment of IDs, etc.

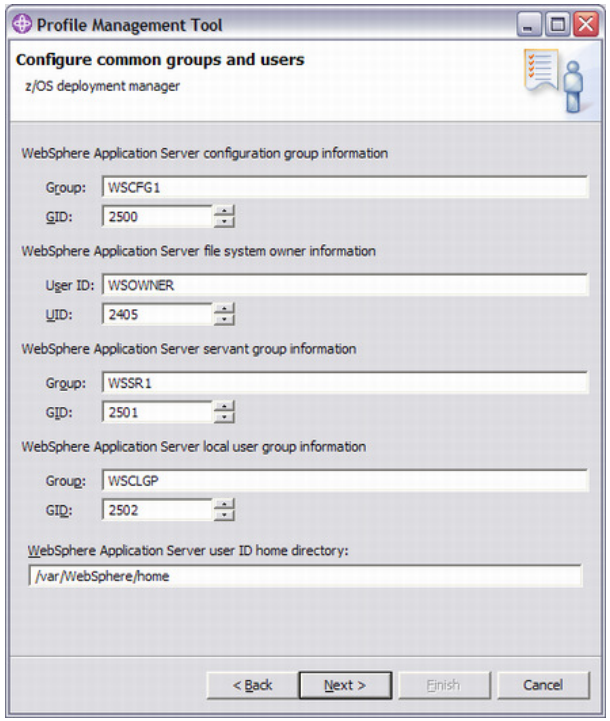
Two key points:

1. To discard a configuration you don't like, all you need do is clean up these three things (SAF being the most complicated)
2. These things are created by the configuration tool called the "WCT" and are customized with your specific names and values

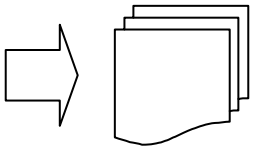
The WCT Configuration Tool

Is a workstation graphical tool that captures key names, values and input from you and consistently imbeds those values in customized batch jobs.





Made up of two tools…zPMT and zMMT



Customized Batch JCL jobs

These get uploaded to z/OS where they're submitted, one after another, to create the configuration runtime.

Uploading and running the jobs is the easy part.

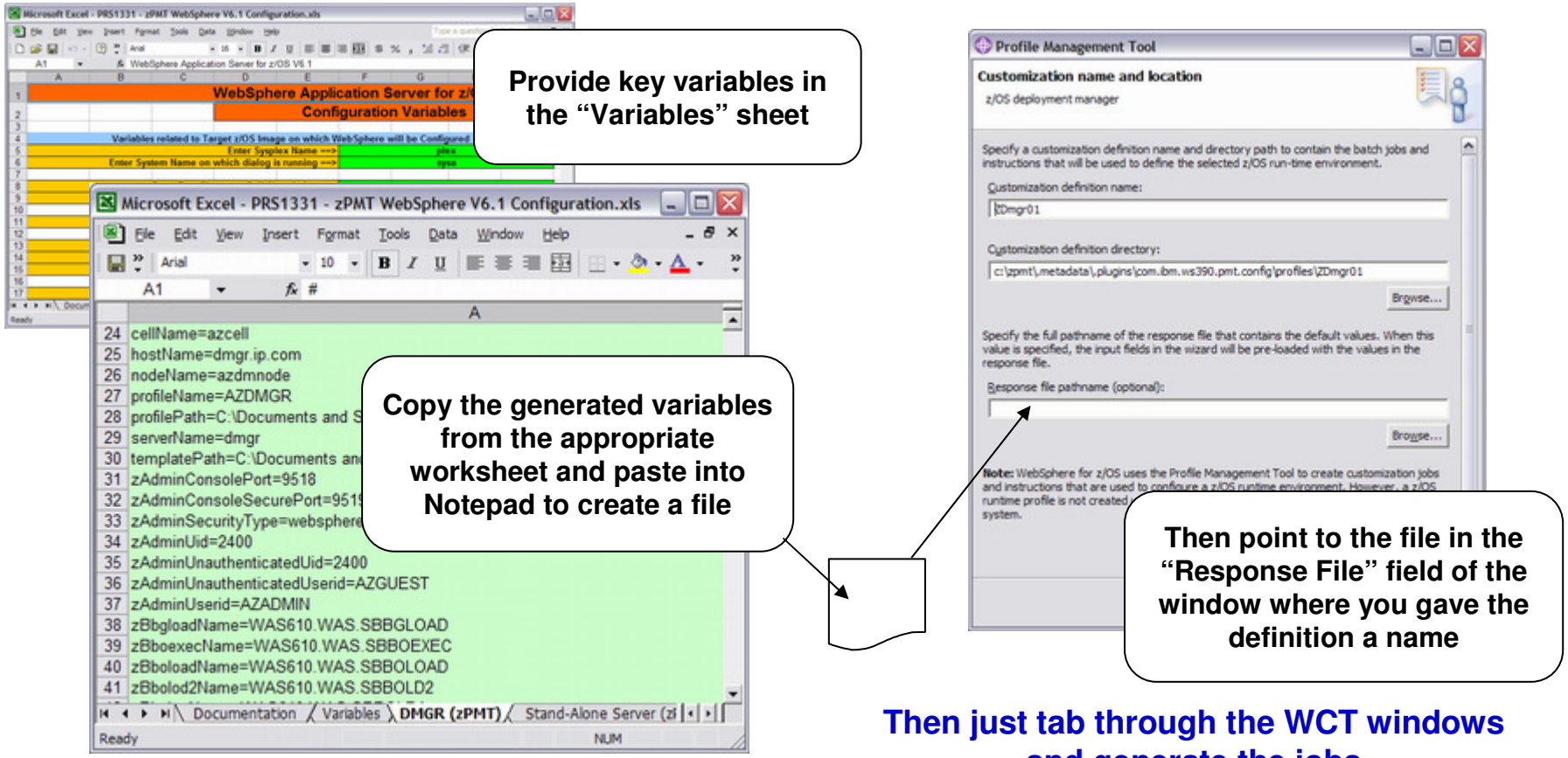
The real challenge is coming up with all the names and values and ports the WCT is going to ask for. Without a plan for those names you'll very quickly get confused.

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3357

Spreadsheet …

The PRS3341 Planning Spreadsheet

An Excel spreadsheet that makes planning values and using the WCT much easier … it helps enforce a disciplined "top down" design:



Then just tab through the WCT windows and generate the jobs

The Generated Jobs and Running Them

Let's look at example of generated job -- this will help "demystify" this:

`BBOCCINS`	Instruction checklist

`BBOSBRAK`
`BBOSBRAM`
`BBODBRAK`
	Creates the RACF profiles

`BBODCPY1`	Copies customized JCL procs into PROCLIB.

`BBODCHFS`
`BBODHFSA`
`BBOWWPFD`
	Allocates HFS, creates directory structure, copies in customized XML, and does final build of configuration "profile"

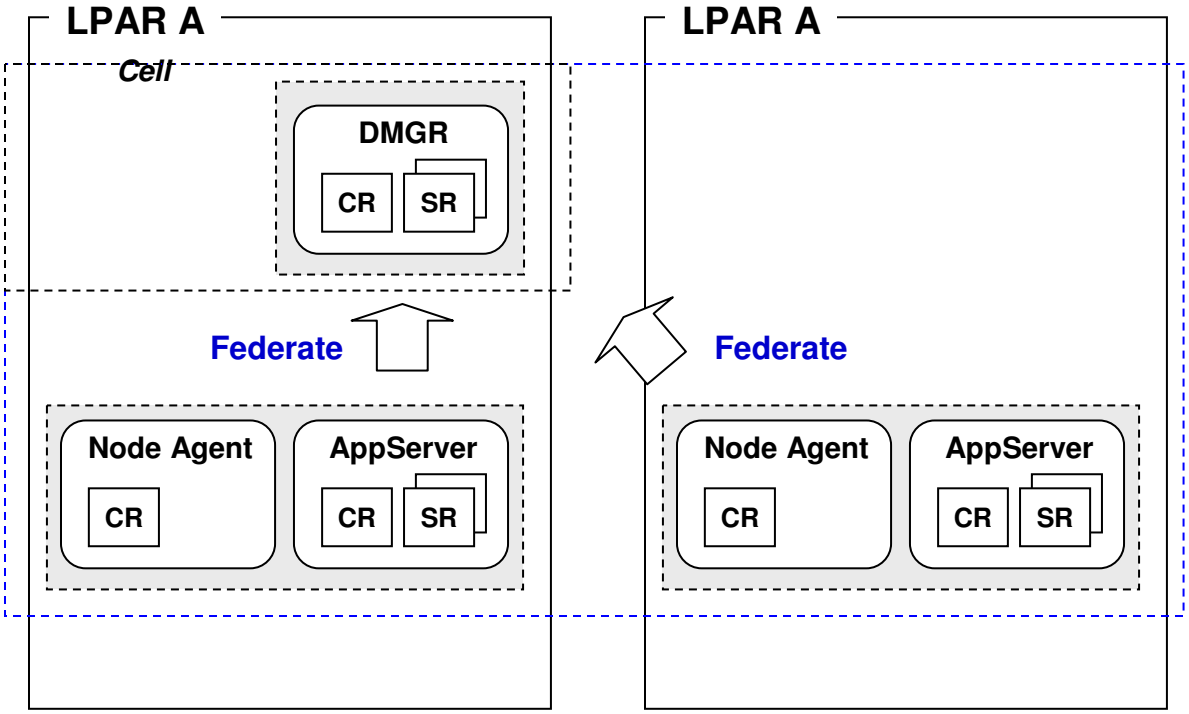
Key Points:

- No real magic to this … just batch jobs that do mundane things
- Key is how the jobs are customized, and that's where the spreadsheet/zPMT comes in
- Running the jobs is easy … making sure jobs have right information is the key

Typical problems are -- typos in the input data (spreadsheet helps avoid this) and insufficient authority (what you need is well documented)

Build Nodes and Federate

The jobs build a node. To build a bigger cell you do what's called "Federate." This involves running a batch job to join one node into the DMGR's cell:



Key Points:

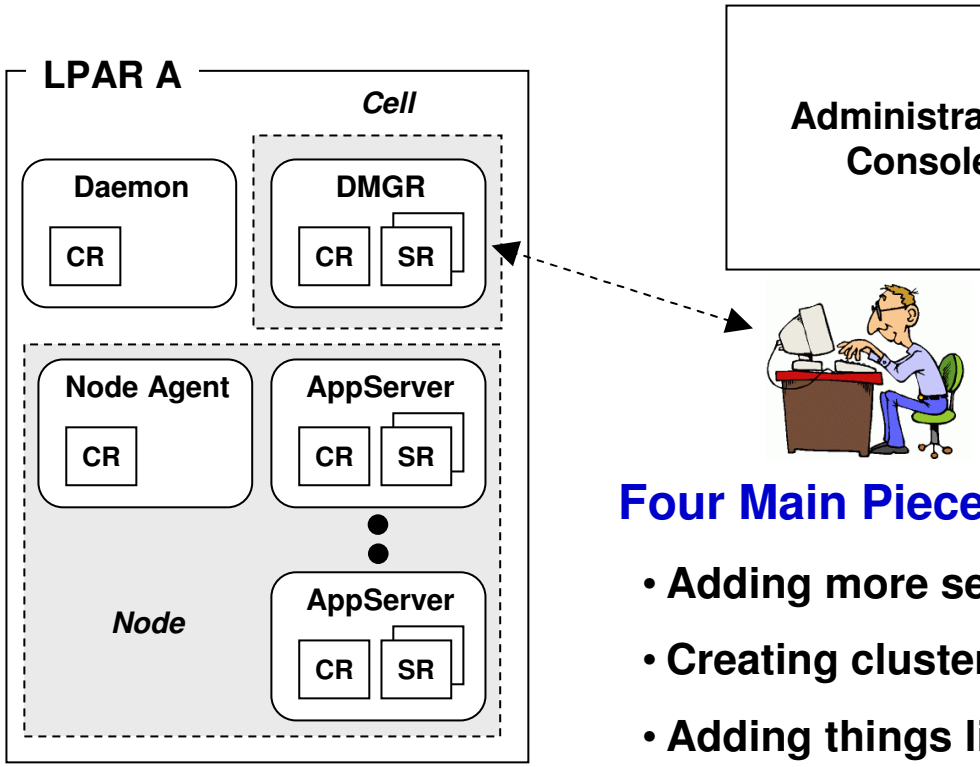
- This is a "building block" approach
- Build as big a cell as you want using this technique
- The DMGR's cell grows to pick up the nodes being federated

Details of this deliberately left out … don't worry about those right now. Key is the concept of joining a node into the DMGR's cell to make it grow. Get that concept and you're half-way home.

Post-Creation Customization and Using the WebSphere Runtime

What You Have After You've Built Your Cell

After you've done all that you have a configuration that is capable of accepting applications to run:



But your cell will no doubt require more post-creation customization

Four Main Pieces to this:

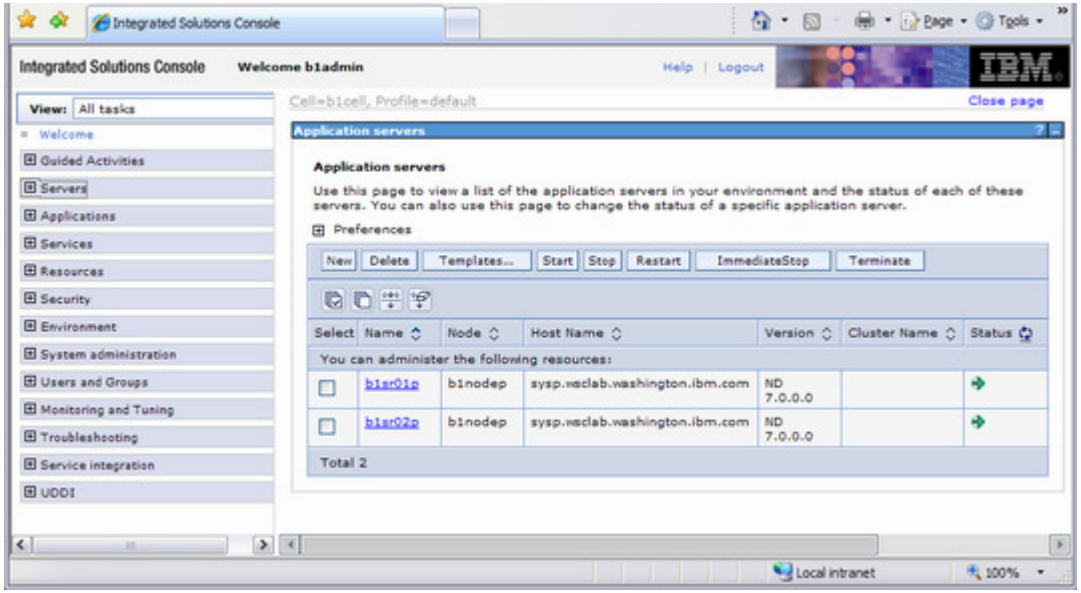
- **Adding more servers if you see the need for them**
- **Creating clusters**
- **Adding things like JDBC, JCA and MQ**
- **Deploying applications and starting them**

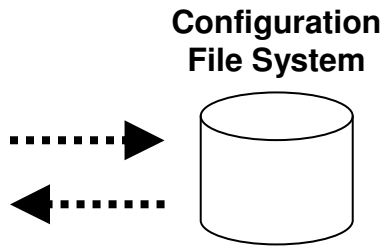
This post-creation customization is common across all platforms … it's not just a z/OS thing.

In fact, it's common across all middleware -- DB2, CICS, MQ … all require some customization

The Administrative Console

As we said, this is a very smart web application that knows how to update XML files in the configuration based on the point-and-click actions you do



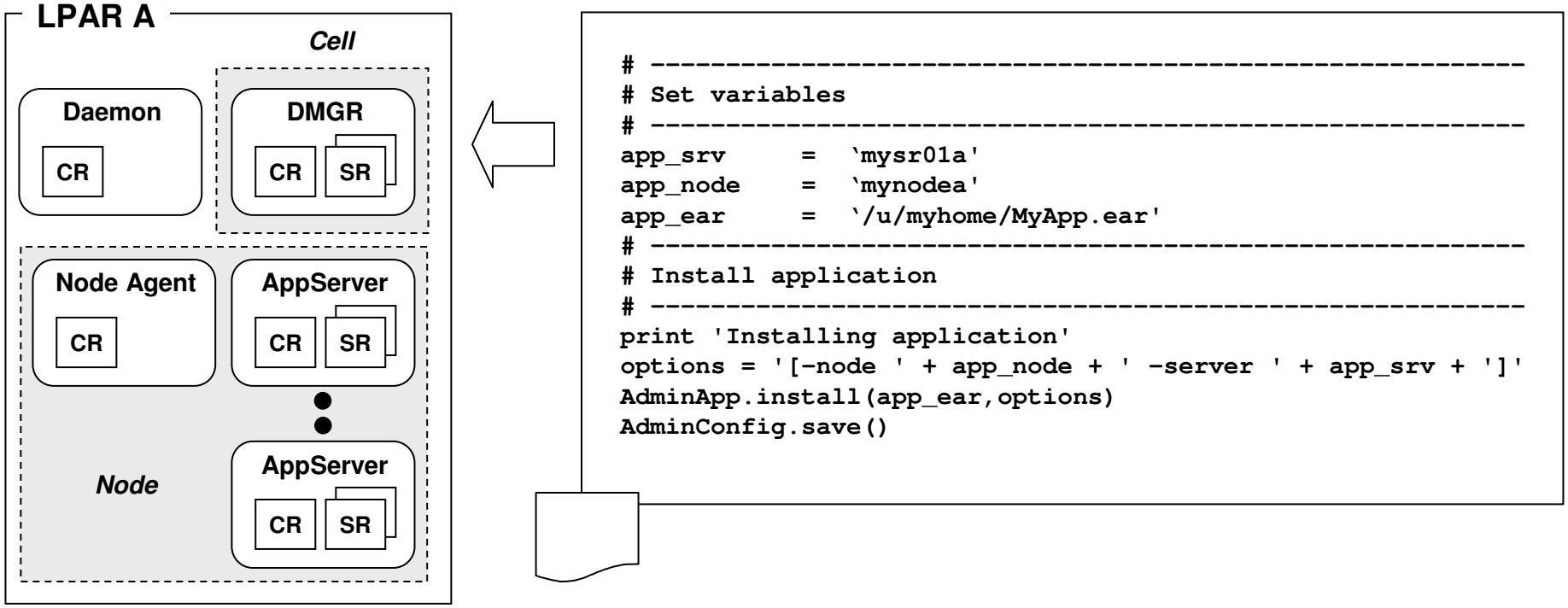


Key Messages:

- There are a lot of options within the Administrative Console
- You learn this over time … you can't master this right away
- Always remember what it's doing -- updating configuration XML with your new information, such as JDBC, MQ, applications, etc.

WSADMIN Scripting Interface

There's also a programmatic scripting interface that allows you to automate tasks. You can do nearly anything with WSADMIN you can with Admin Console.



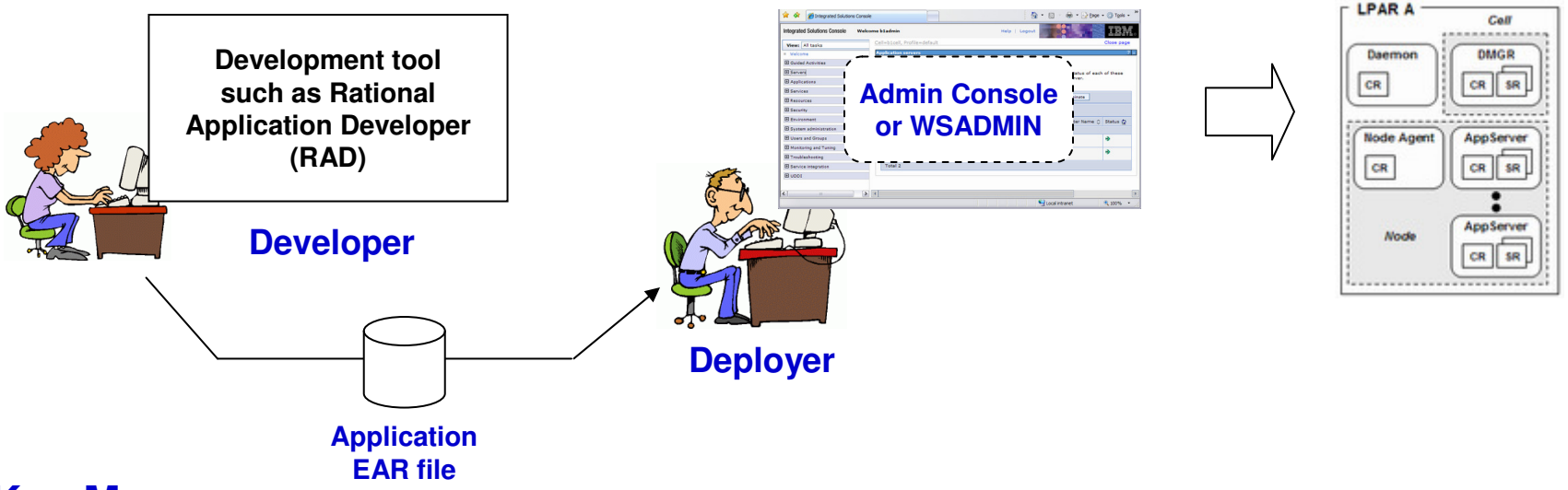

```
# ------------------------------------------------------------
# Set variables
# ------------------------------------------------------------
app_srv      =   'mysr01a'
app_node     =   'mynodea'
app_ear      =   '/u/myhome/MyApp.ear'
# ------------------------------------------------------------
# Install application
# ------------------------------------------------------------
print 'Installing application'
options = '[-node ' + app_node + ' -server ' + app_srv + ']'
AdminApp.install(app_ear,options)
AdminConfig.save()
```

Key Messages:

- Very handy for repetitive tasks
- Useful to insure consistent deployments across QA, Test, Production
- Does take some getting used to … like any programming language

Overview -- Deploying Applications

In WebSphere, an application is typically packaged as an "EAR" file -- a zip format file that contains all the piece-parts of the application:

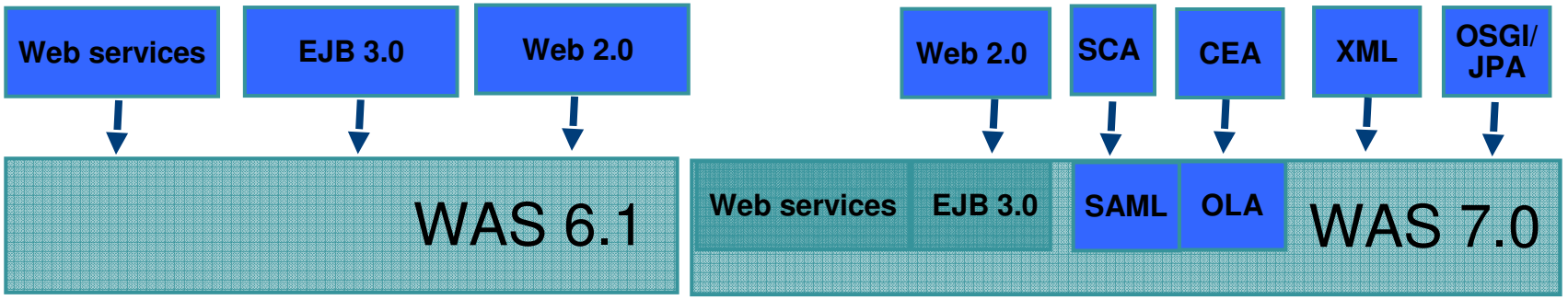


Key Messages:

- **The Admin function will "break open" the zip-format EAR file and put the individual files in the proper places in the configuration file structure**
- **Who performs role of "deployer" is different in each customer … sometimes a separate group; sometimes the z/OS system programmers.**
- **Some knowledge of the application and what's it's designed to do is necessary. You can't deploy applications blindly.**
  Talk to your developers and understand what's going on in the application!

WebSphere Feature Packs

Revolutionizing the way customers consume application server technology now and in the future

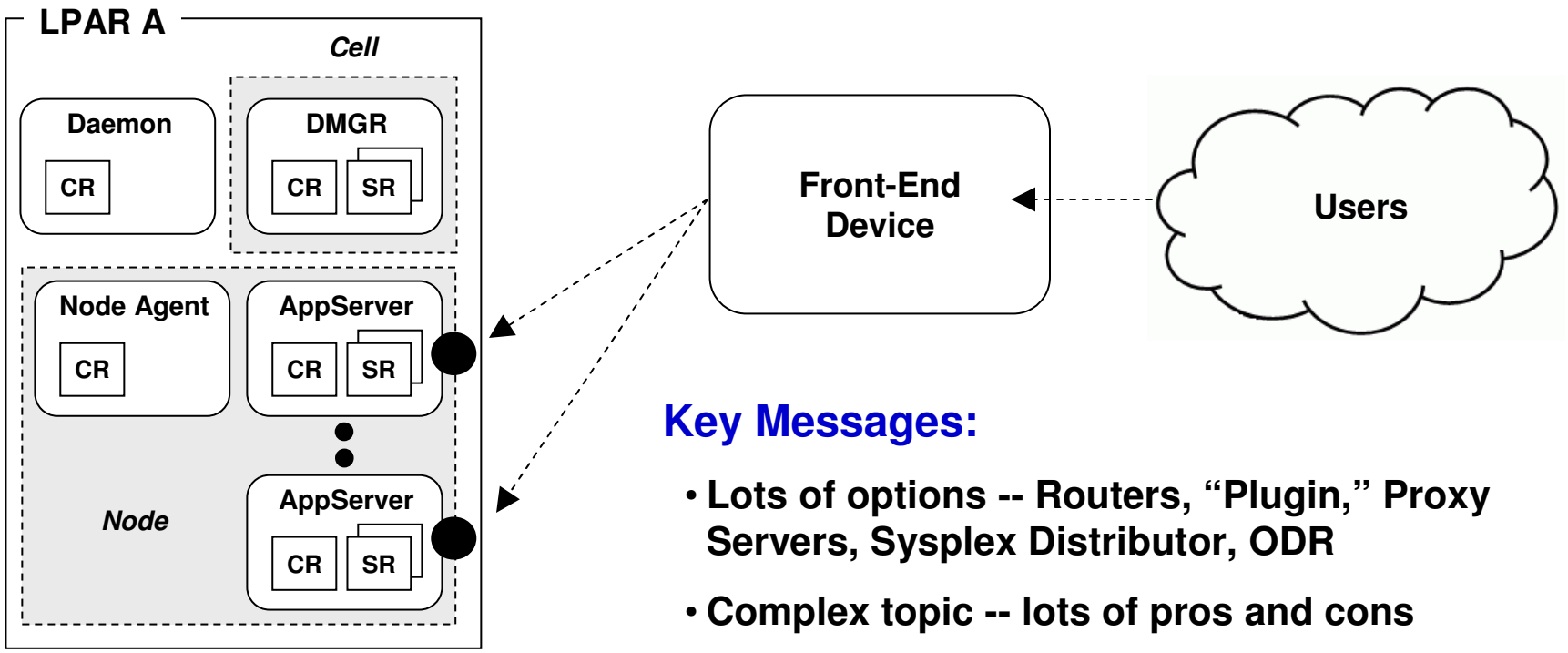


1. Choose the application server technology you need.
2. Install additional functionality on core WAS Application Server
3. Build the Application Server you want without waiting for new releases.

As new technology evolves, so does WebSphere –
Get the technology you need now without waiting for a new release!

Front-End Load Balancing Devices

Each WebSphere application server has its own HTTP listeners … so something "out front" is typically required.



Key Messages:

- Lots of options -- Routers, "Plugin," Proxy Servers, Sysplex Distributor, ODR
- Complex topic -- lots of pros and cons

Grand Summary

And we come to the end … with a single summary chart:

- **"Application Server"**
  Provides a common set of functions and services so developers can focus on business-value, not plumbing code.
- **"Open Standards"**
  Is what allows the industry to settle on things that allow interoperability.
- **Common at JVM and above; platform specific below**
  This provides the ability to move applications from platform to platform, allowing you to choose the platform based on the strengths of that platform.
- **z/OS Implemented as Started Tasks -- Exploits the Platform**
  Implemented in a way that's familiar to z/OS system programmers. Lower plumbing code takes advantage of the platform while the "JVM layer" shields the applications from having to have direct knowledge of the platform
- **Blended solution -- WebSphere + CICS, DB2, MQ, etc.**
  WebSphere Application Server complements the others, allowing existing applications to be maintained and allowing you to choose the system that best suits your needs.