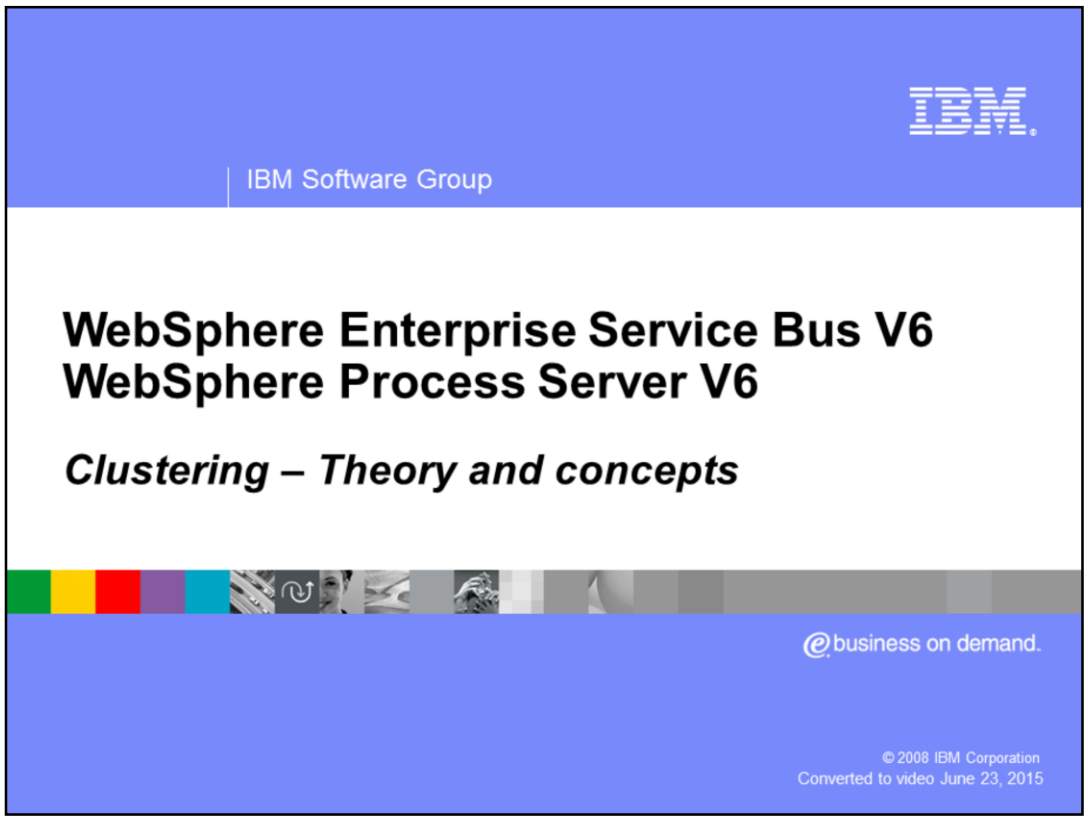
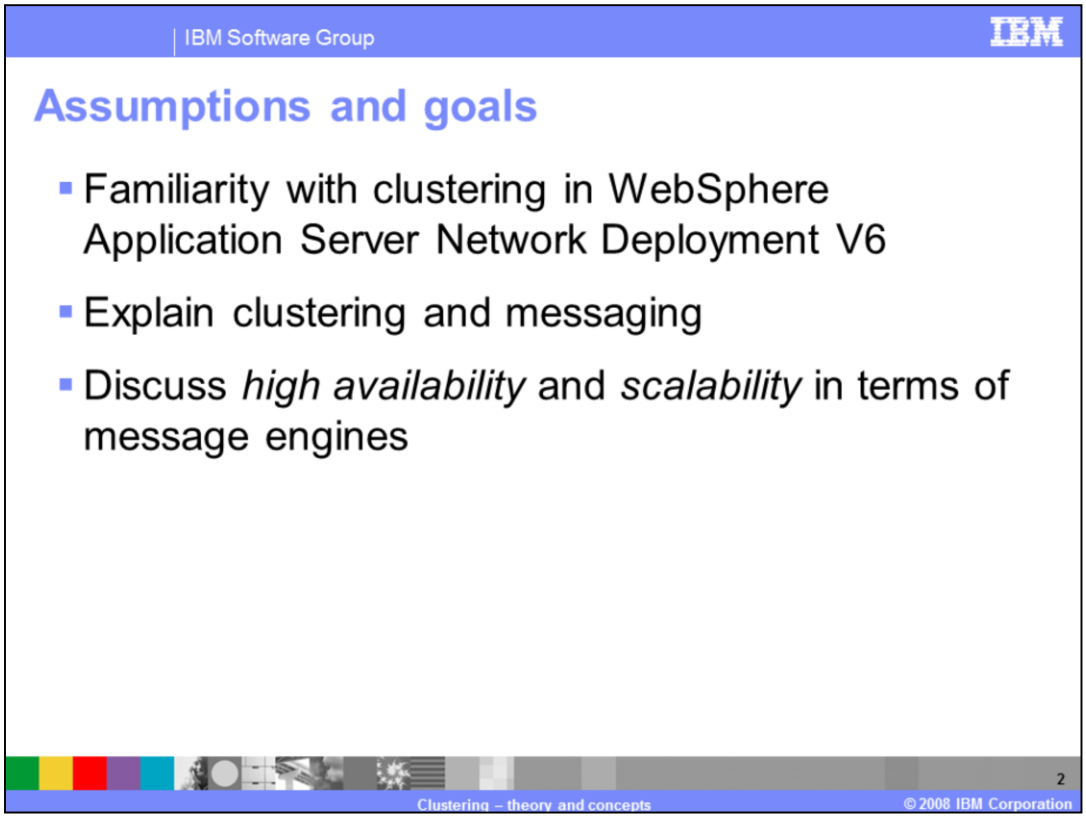
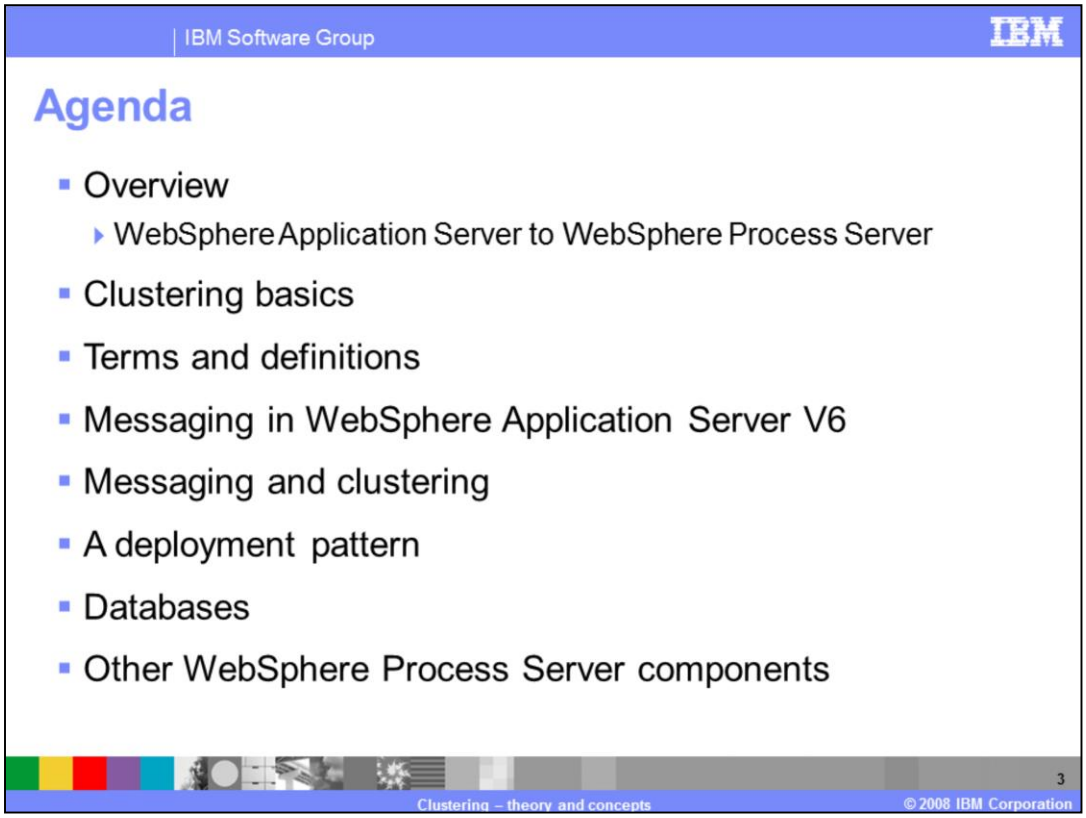


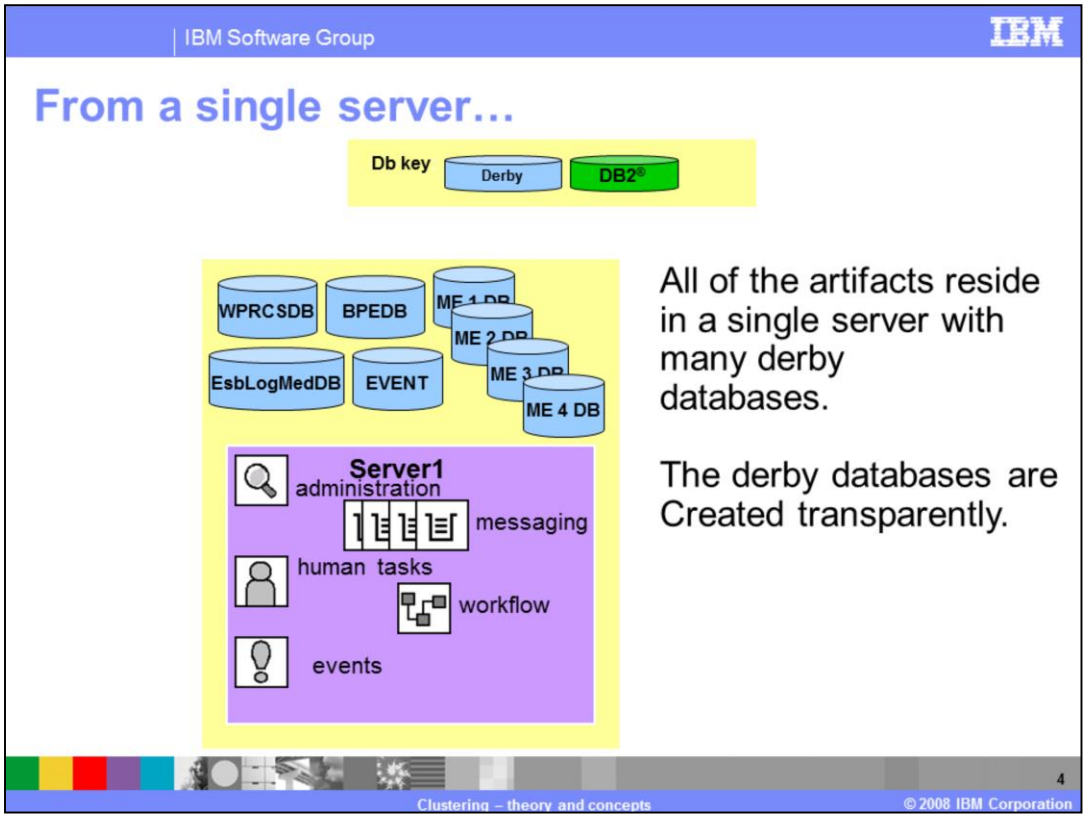
This presentation covers the theory and concepts of clustering for the WebSphere Enterprise Service Bus and the WebSphere Process Server V6. Throughout this presentation all references to the WebSphere Process Server also apply to the WebSphere Enterprise Service Bus.



This presentation assumes that you are familiar with clustering in WebSphere Application Server Network Deployment V6. The goal is to present the underlying and motivating issues involved with clustering and messaging in a WebSphere Process Server environment. The next goal is to discuss high availability and scalability in terms of Message Engines.

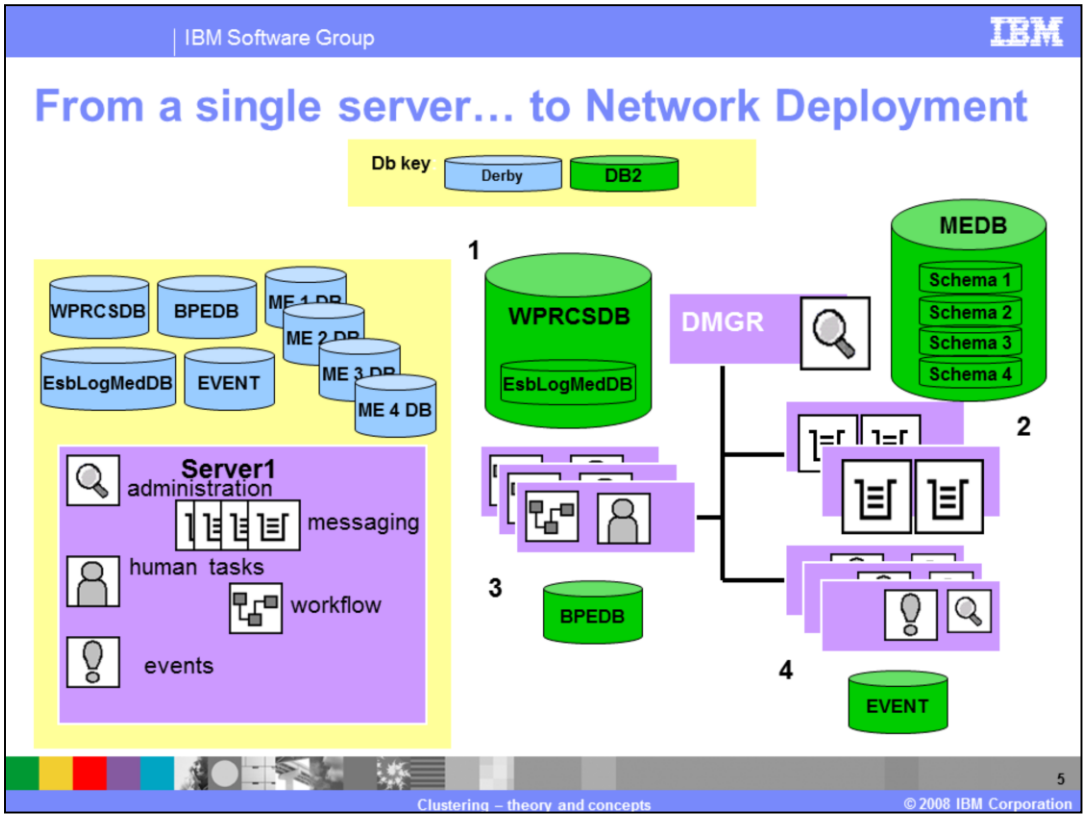


The agenda for this presentation begins with an overview of the differences between WebSphere Application Server and WebSphere Process Server. Then it will cover clustering basics, including some terms and definitions, before moving into Messaging topics. Finally, deployment options, database options, and other components are discussed.



This demonstrates the artifacts as they are created in the WebSphere Integration Developer environment before they get deployed to the WebSphere Process Server Network Deployment V6 runtime environment.

Initially everything is in the WebSphere Integration Developer, using Derby for the databases and everything is all in one server.



To get to the fully network deployed configuration….

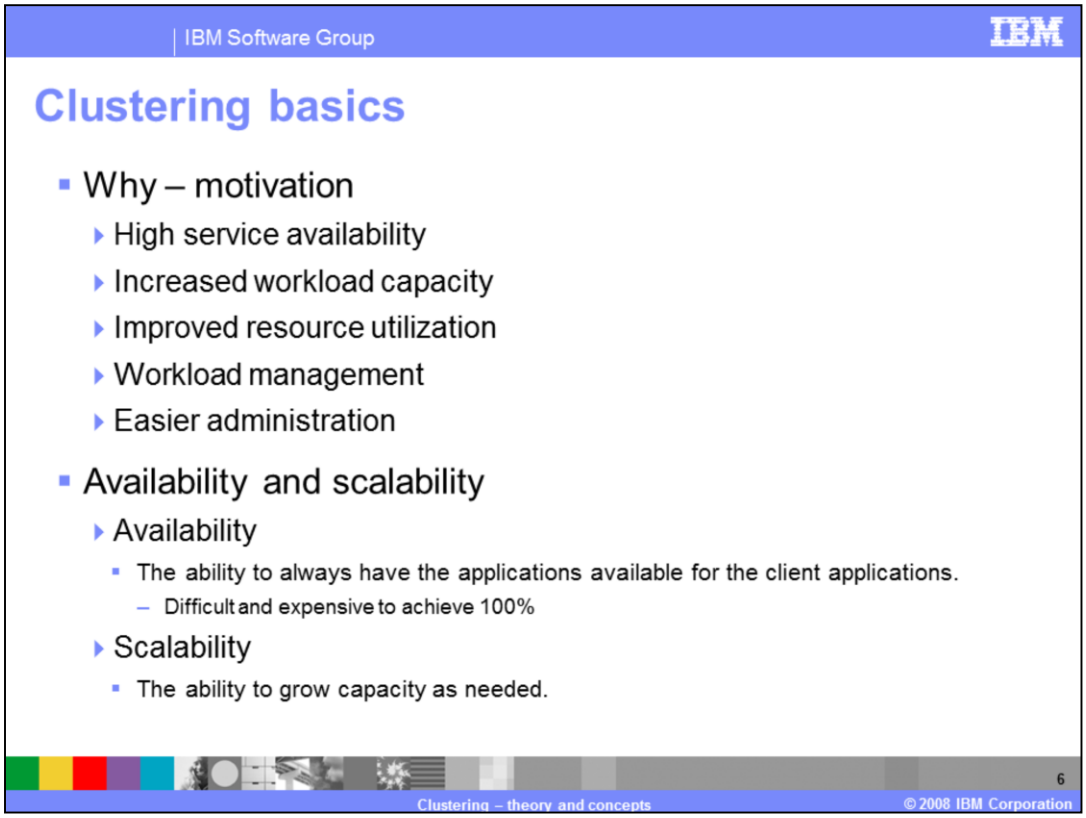
1. Create the WPRCSDB database and incorporate the tables required for the Enterprise Service Bus mediations.

2. As the messaging component moves to the runtime environment, the data stores for the Message Engines are consolidated to a centrally located, remote database and the JMS resources are established on numerous servers.

3. As the business processes move to the runtime, the BPEDB database is created and the applications are deployed on numerous servers.

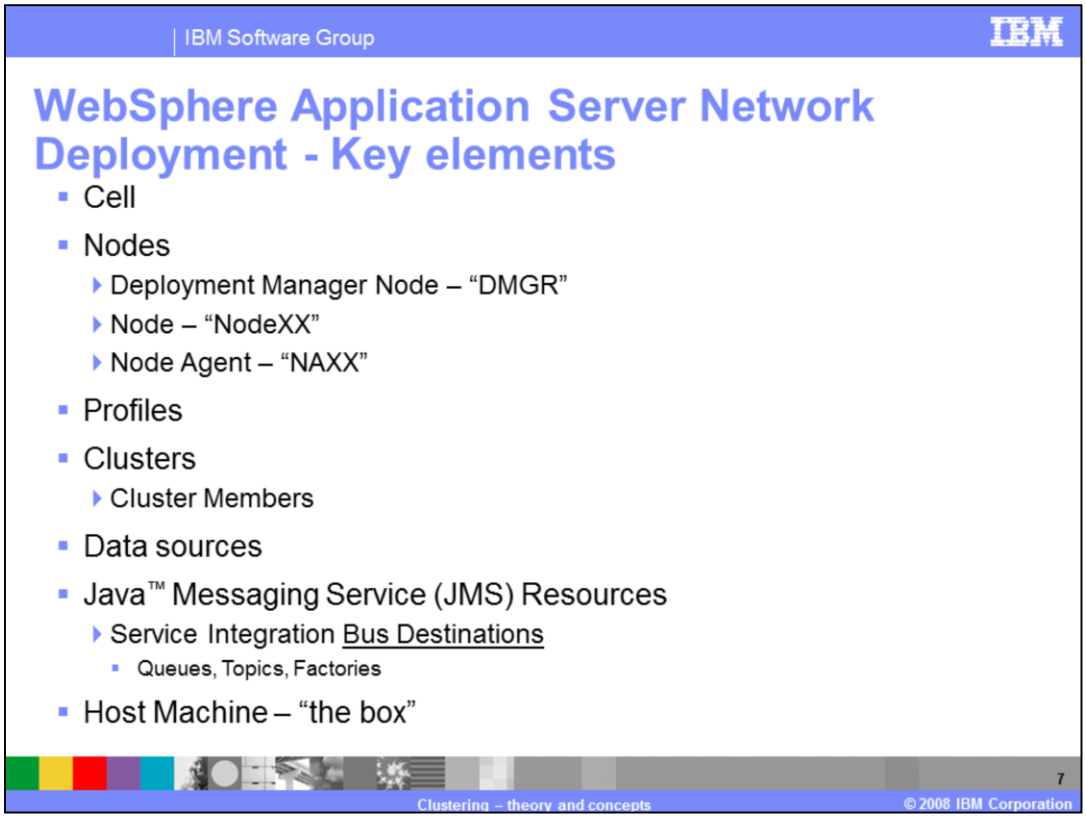
4. Moving the Common Event Infrastructure component to the runtime causes the creation of yet another database that is specifically for the Common Event Infrastructure events. These events can be generated from any of the servers, either from the Business Process Choreographer infrastructure or from applications directly.

The picture here provides a sense of what it takes to move from the development environment, to a fully distributed Network Deployment environment for either test or production. There are several databases to be created and managed and decisions about how to distribute the infrastructure and application components.



As an architect or system administrator using WebSphere Process Server to implement your business processes, you expect to have a system that will tolerate failure and allow for maintenance without the loss of data or loss of service. You also need to be able to add processing capacity, to grow your systems to meet increased user demand.

WebSphere Network Deployment provides the ability to create logical groups of servers, with the servers being distributed across one or more machines. This capability provides a mechanism to tolerate failover, apply maintenance to some servers while others keep running, grow or shrink capacity by adding and removing servers from the group, all with a single point of administration.



Shown here are the key elements of a WebSphere Network Deployment cell that are necessary for a discussion on clustering.

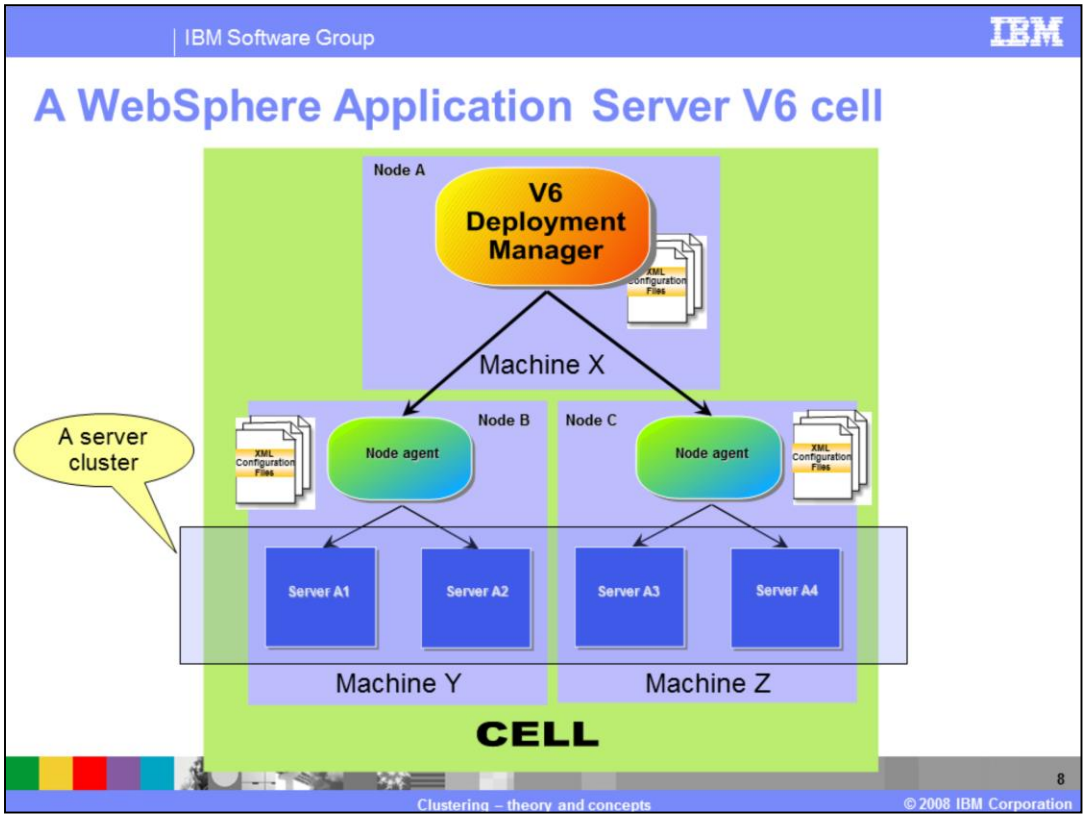
A **node** is a logical grouping of managed servers. A node typically corresponds to a logical or physical computer system with a distinct IP host address. Nodes cannot span multiple computers.

The WebSphere Application Server **profile** defines the runtime environment for a node. To create a node, the runtime environment to host an application server on a given host machine, you run the profile management tool. One of the primary purposes of the profile is to provide separation between the WebSphere runtime artifacts specific for a given configuration, and the WebSphere Application Server binaries, which are common to all configurations.

Clusters are sets of identical servers that are managed together and participate in workload management. The servers that are members of a cluster can be on different host machines.

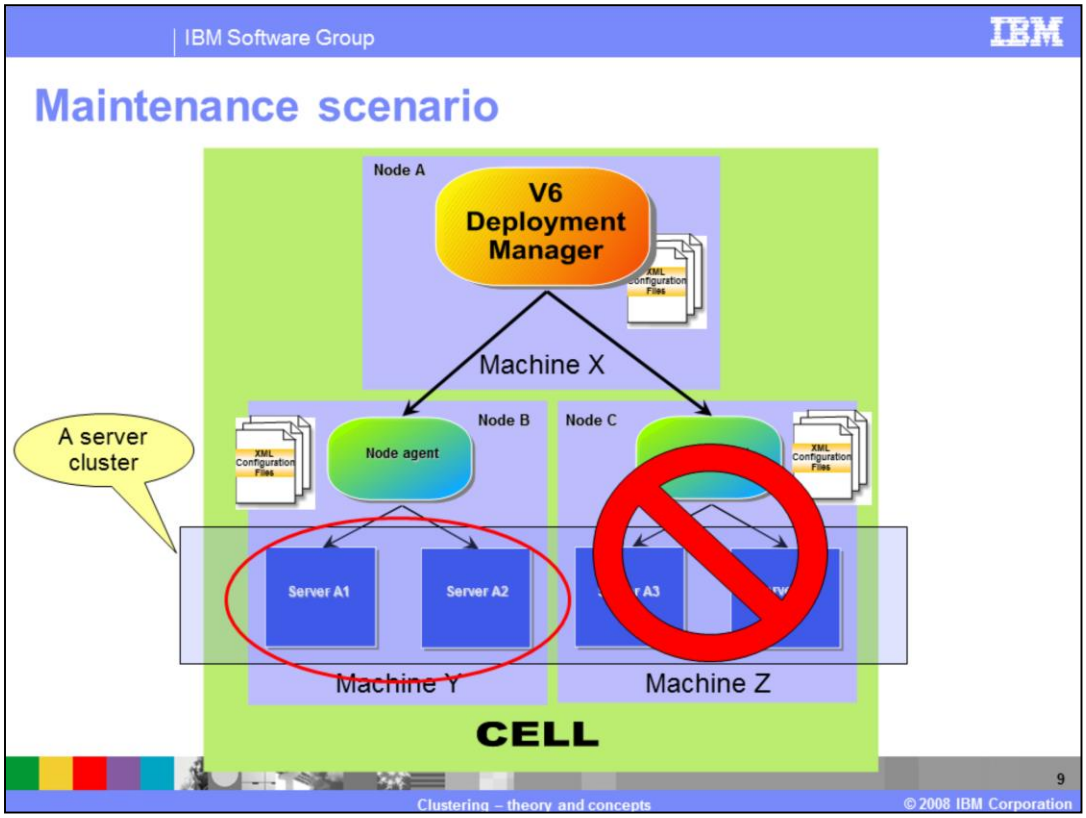
A **Bus destination** is a virtual location within a service integration bus, to which applications attach as producers, consumers, or both to exchange messages.

There is "the box" or "the host machine" which often gets confused with the server. A server is the Application Server, which is a software entity that hosts applications, whereas the host machine is the physical hardware that the server or servers are running on.

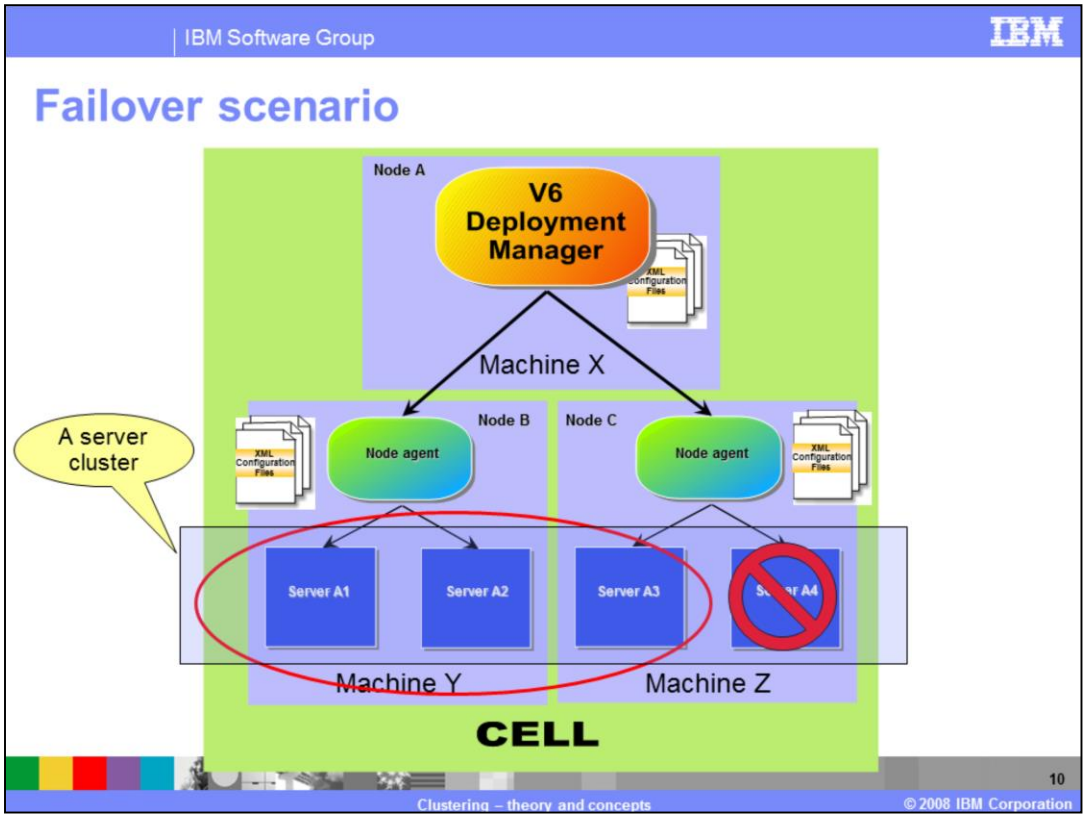


The nodes can be on separate boxes or they can be on the same box. They are shown here on separate boxes and the deployment manager is on a separate machine as well. To host more than one node, the host machine must have ample memory, disk space, and processor capacity.

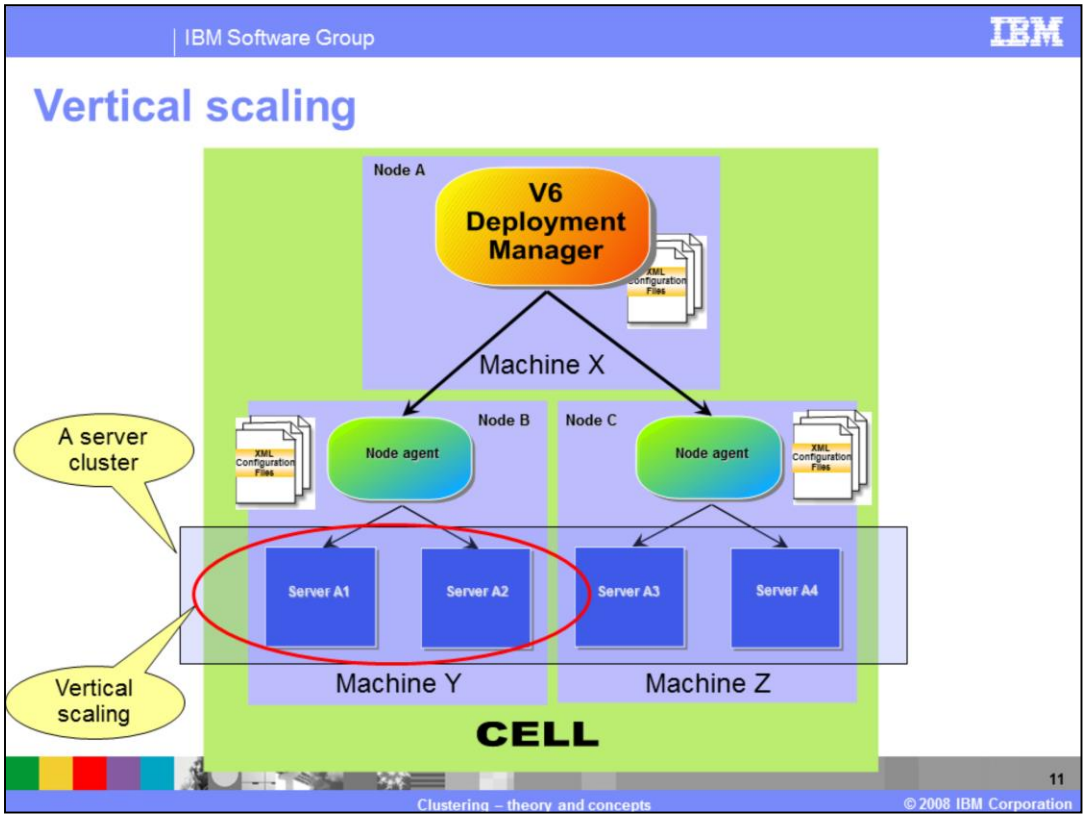
All four servers in the **server cluster** are identical.



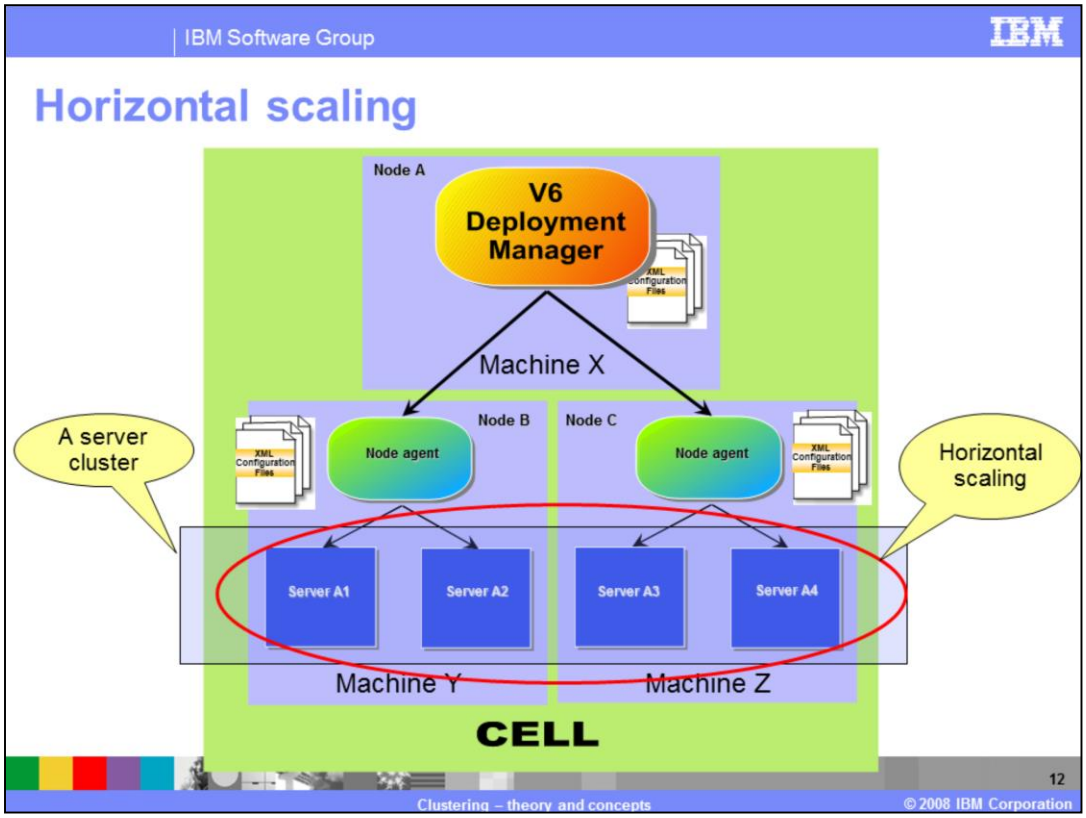
If Machine Z needs to be taken off-line for maintenance then Machine Y will still have two servers available to service requests.



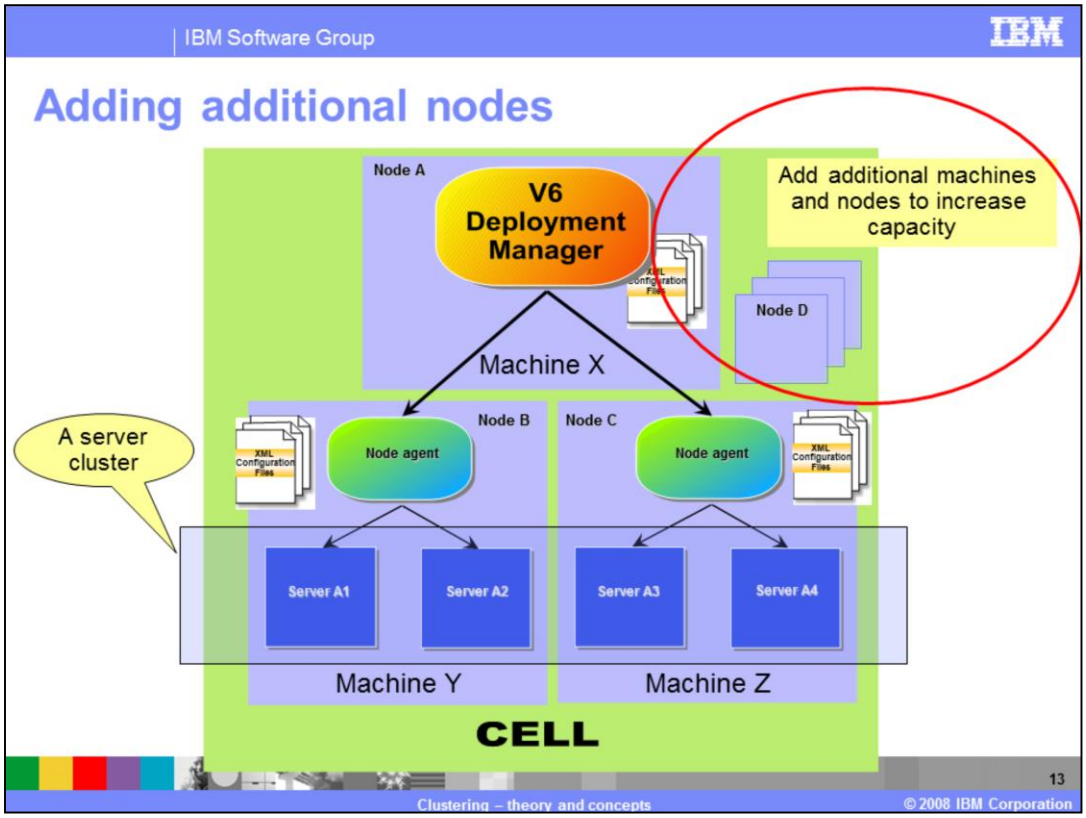
If any one of the servers experiences a problem and depending on how the system is configured, it is possible for one of the remaining three to recover the work in progress. That is, **failover** to one of the remaining servers in the server cluster.



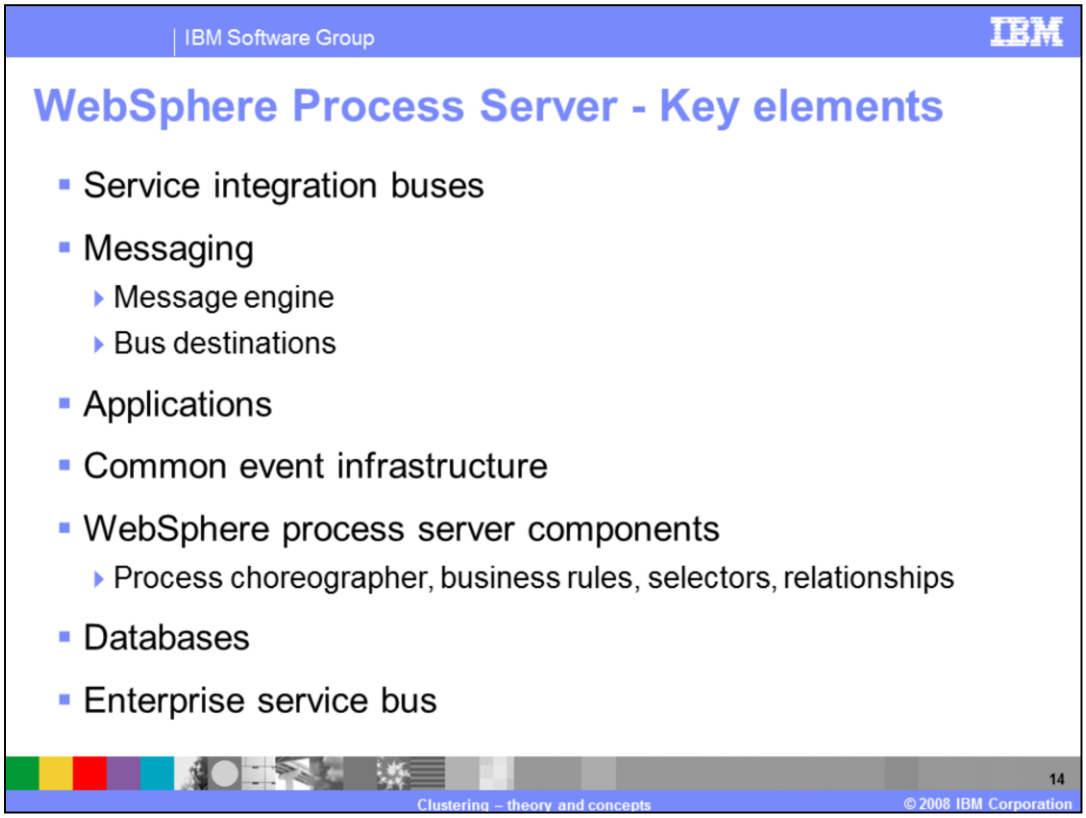
Adding additional application servers to a server cluster on the same machine is called ***vertical*** scaling



Adding additional application servers to the server cluster on a different machine is called ***horizontal*** scaling.



To **scale up** the capacity, one or more machines can be brought online and the new nodes can be federated into the cell and more servers can be added to the cluster



Although service integration buses and messaging are part of WebSphere Application Server Network Deployment V6, they are listed here because they are key elements for WebSphere Process Server solutions.

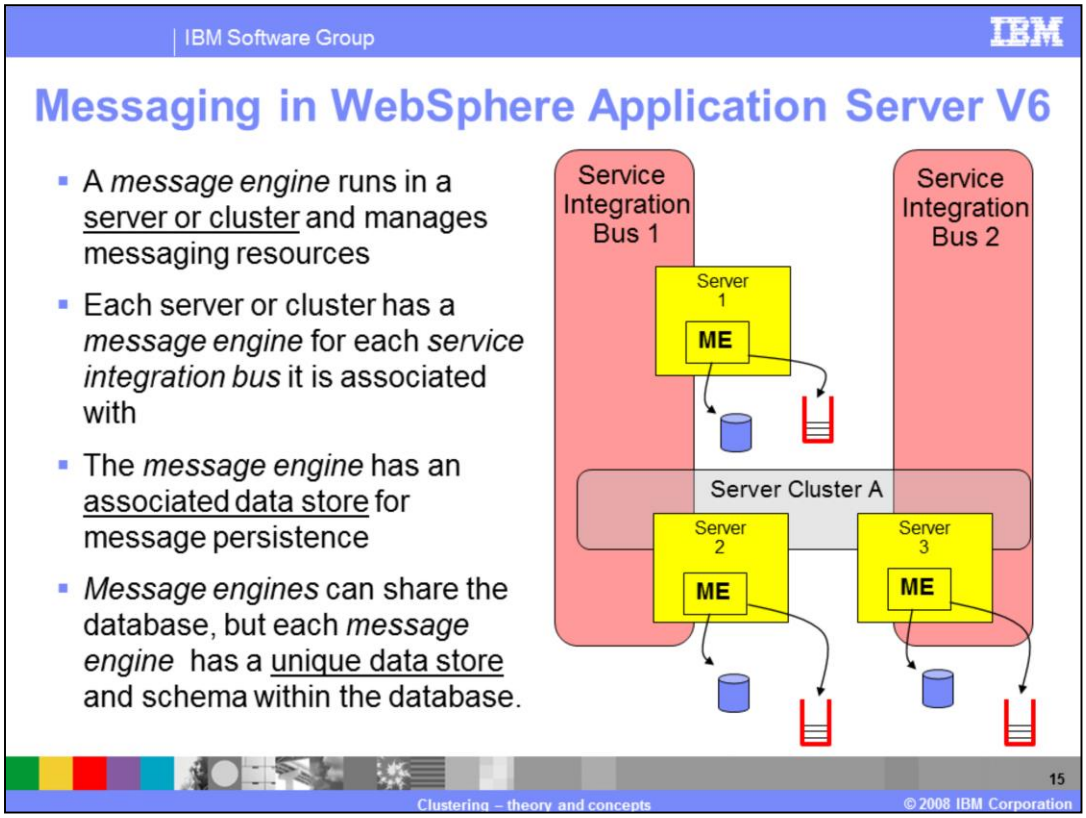
A ***service integration bus*** supports applications using message-based and service-oriented architectures. A bus is a group of one or more interconnected servers or server clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

A ***messaging engine*** is a server component that provides the core messaging functionality of a service integration bus. A messaging engine manages bus resources and provides a connection point for applications.

The JMS destination is the queue.

The ***Enterprise Service Bus*** is a feature provided with WebSphere Process Server V6 that adds additional messaging and service oriented features such as complex transformations and mediations.

Messaging is at the heart of the WebSphere Process Server functionality. Its used for the Common Event Infrastructure, Business Process Choreography and for the asynchronous SCA invocations. This is why you need a thorough understanding of WebSphere Application Server Network Deployment V6 messaging and clustering in order to understand clustering in WebSphere Process Server. The next few slides will discuss the fundamentals of WebSphere Application Server Network Deployment V6 messaging and clustering.



Server 1 is a member of the Service Integration Bus **Service Integration Bus 1.**

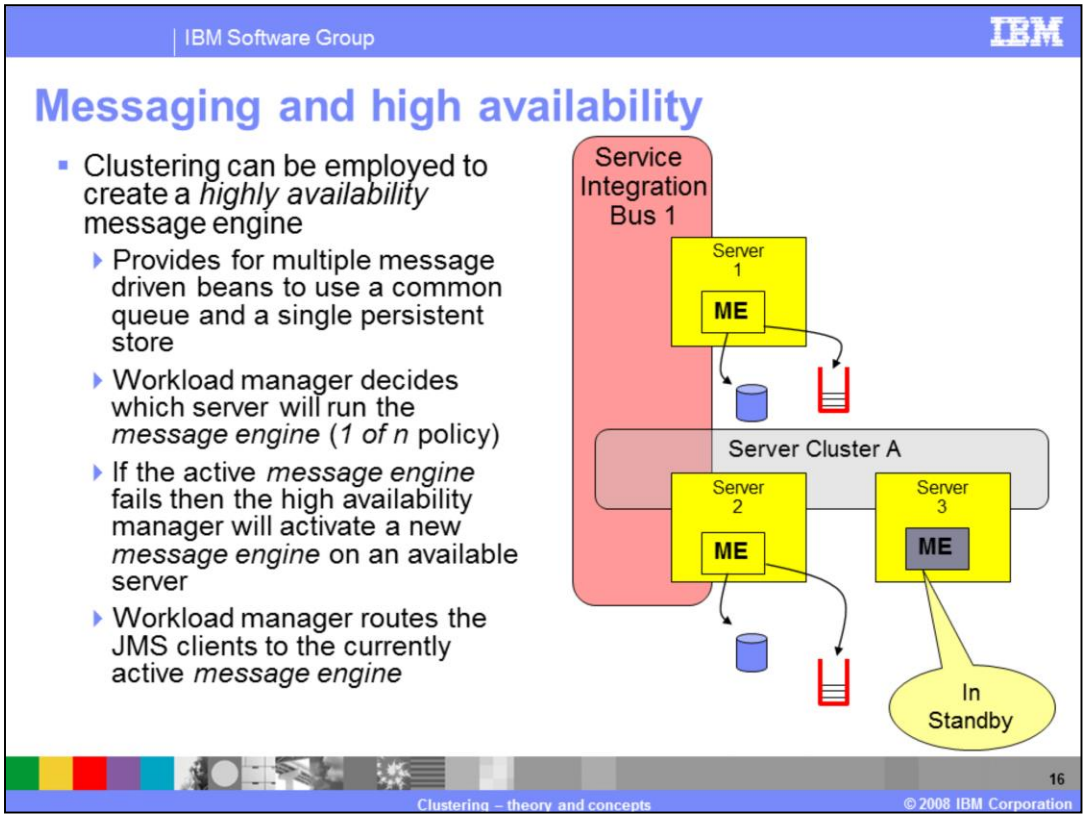
Server 2 and 3 are both members Server Cluster A which is also a member of the **Service Integration Bus 1** and **Service Integration Bus 2**.

A ***service integration bus*** supports applications using message-based and service-oriented architectures. A bus is a group of one or more interconnected servers or server clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Each ***messaging engine*** is associated with a server or a server cluster that has been added as a member of a bus. When you add an application server or a server cluster as a bus member, a messaging engine is automatically created for this new member. If you add the same server as a member of multiple buses, the server is associated with multiple messaging engines (one messaging engine for each bus).

The **bus members** of a service integration bus are the application servers and server clusters within which messaging engines for that bus can run.

Note that there are two different kinds of aggregations, the collection of servers (and server-clusters) that are part of (members) the Service Integration Bus and then there is another collection of servers that comprise the server cluster. A ***cluster member*** refers to the relationship between a server and the server cluster and a ***bus member*** refers to the relationship between the server or server-cluster and the service integration bus.



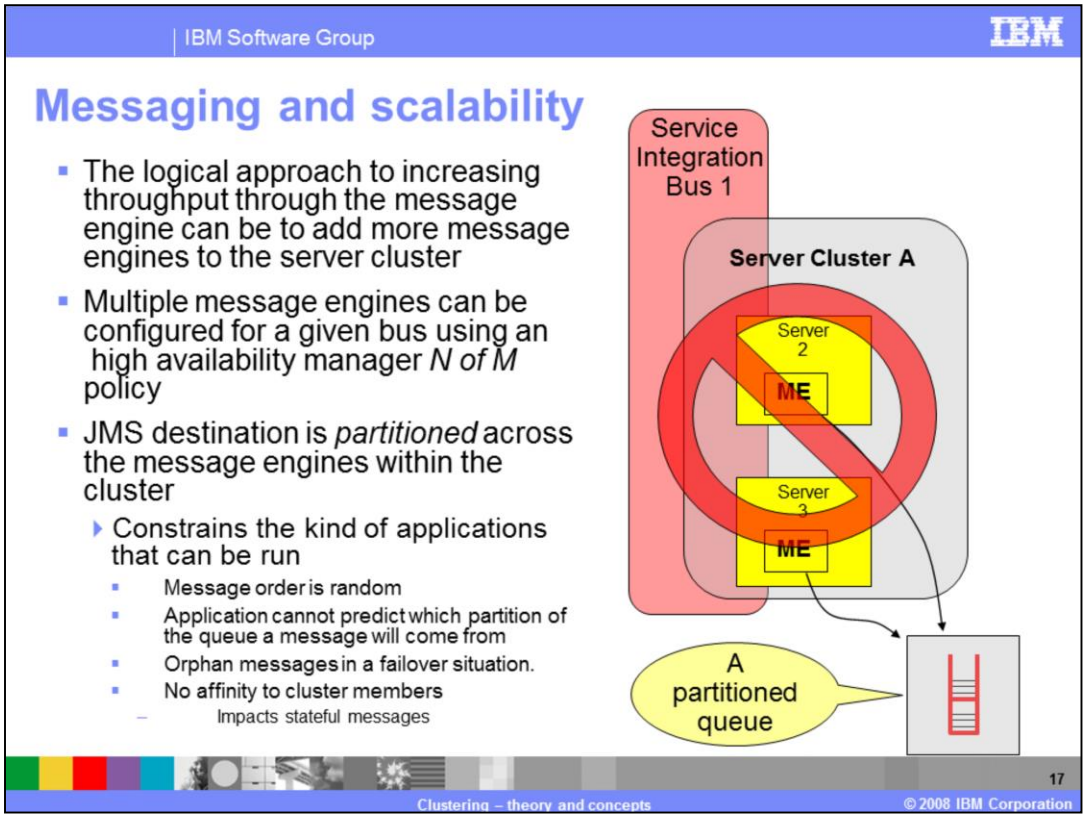
When a application server cluster is added as a member of the service integration bus the message engine is created for the cluster using the active/standby pattern by default.

This configuration is appropriate when the goal is *high availability*, and when there must always be a messaging engine available. This capability is also referred to as *failover*.

The drawback with this configuration is that there is only one queue and database table for the cluster. This can become a bottleneck. If this happens, the only way to get more throughput is to add a server on a faster computer with more memory.

It is possible to create a server cluster that has computers with different levels of service, some can be faster than others. In this situation the server with the highest capacity can be designated as the primary and the other servers can be the standbys.

When a cluster has servers that have different capacities, this is referred to as a mixed configuration. The applications running on the servers must all be the same but the configuration information can be unique for each server.



When you consider how to increase messaging throughput, it might seem that you should add more Message Engines to the server cluster.

The drawback with this approach is that it places constraints on the kind of applications that you can run.

When a user of the queue reconnects to the queue it can be routed to a different partition, so messages can not be processed in the order that they were placed on the queue.

The message consumer is not able to determine which partition a message is coming from, therefore it needs to retrieve the messages from all of the servers.

The order of the messages is indeterminant.

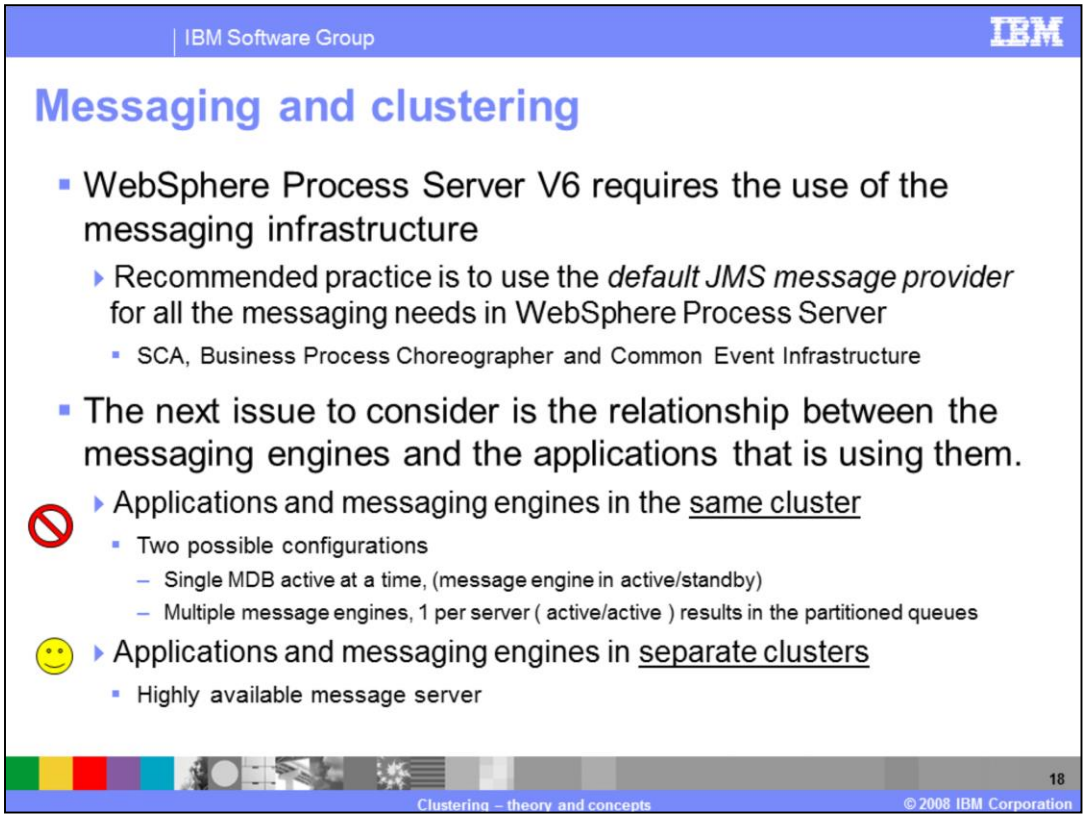
An application can not rely on the assumption that a message that has been put to the queue will still be there on the next connection.

When a cluster member fails, the messages in the associated queue are not available until that server comes back online; they are orphaned.

Because of the constraints imposed on the application, this topology is not recommended for general use.

Note that the pub/sub configuration is a special case where affinity to the queue can be obtained, but this is not applicable in the more general case used by the WebSphere

Process Server components.



The default JMS message provider is the one that comes with WebSphere Application Server V6. It is possible to use other JMS providers for Business Process Choreographer and the Common Event Infrastructure. Since the Service Component Architecture requires the default JMS message provider, the default JMS message provider should be used for the Service Component Architecture and the Common Event Infrastructure requirements as well.

Assuming that the applications are to be clustered then grouping the applications and the Message Engine into the **separate clusters** allows for greater flexibility in configuration and tuning and is the **recommended approach**.

When you consider the case with the applications and the Message Engine in the same cluster there are two alternatives to consider.

1.Active/Standby

- •This is the default provided when a cluster is added to a Service Integration Bus as a bus member.
- •There is one message driven bean per Message Engine and only one Message Engine is active at a time therefore this will limit the throughput of the applications.

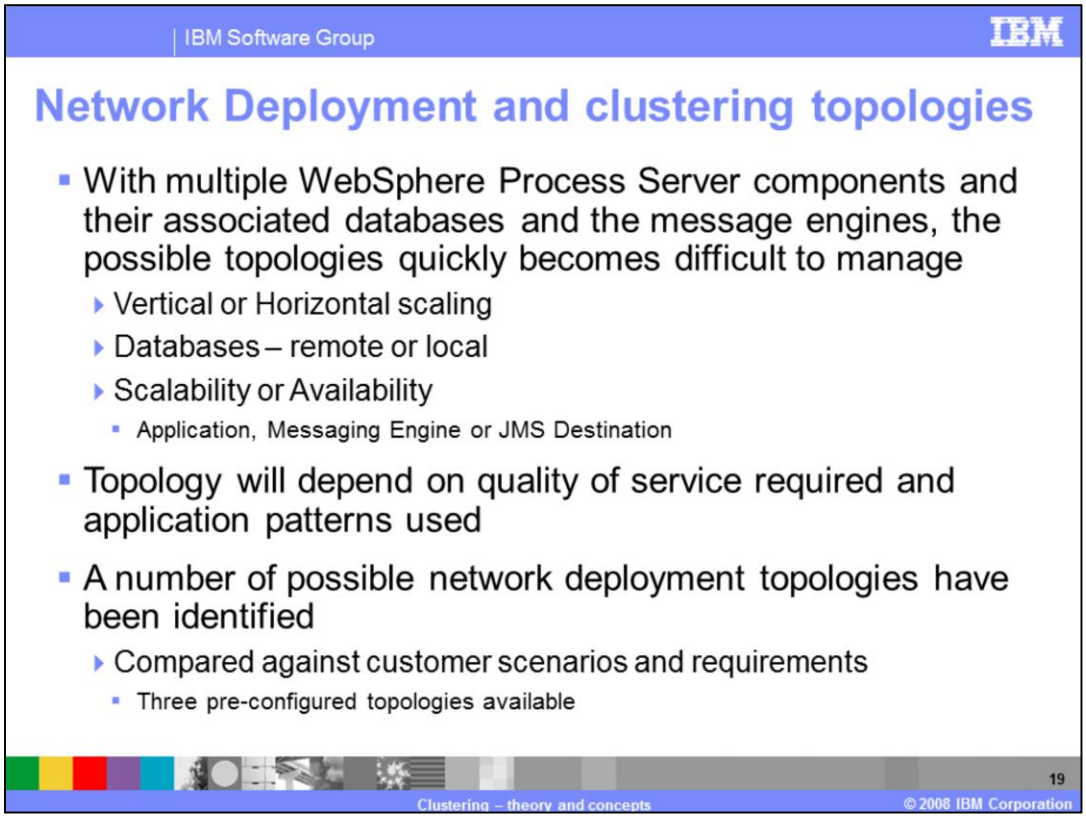
2.Active/Active

- •This requires extra configuration because it is not the default
- •Having multiple Message Engines per cluster will result in the partitioned queue problem discussed previously.

Neither of these alternatives is suitable.

Keeping the Message Engine in a separate cluster will create a highly available message server.

Separating the message engine and the applications is the recommended approach for achieving *high availability*.



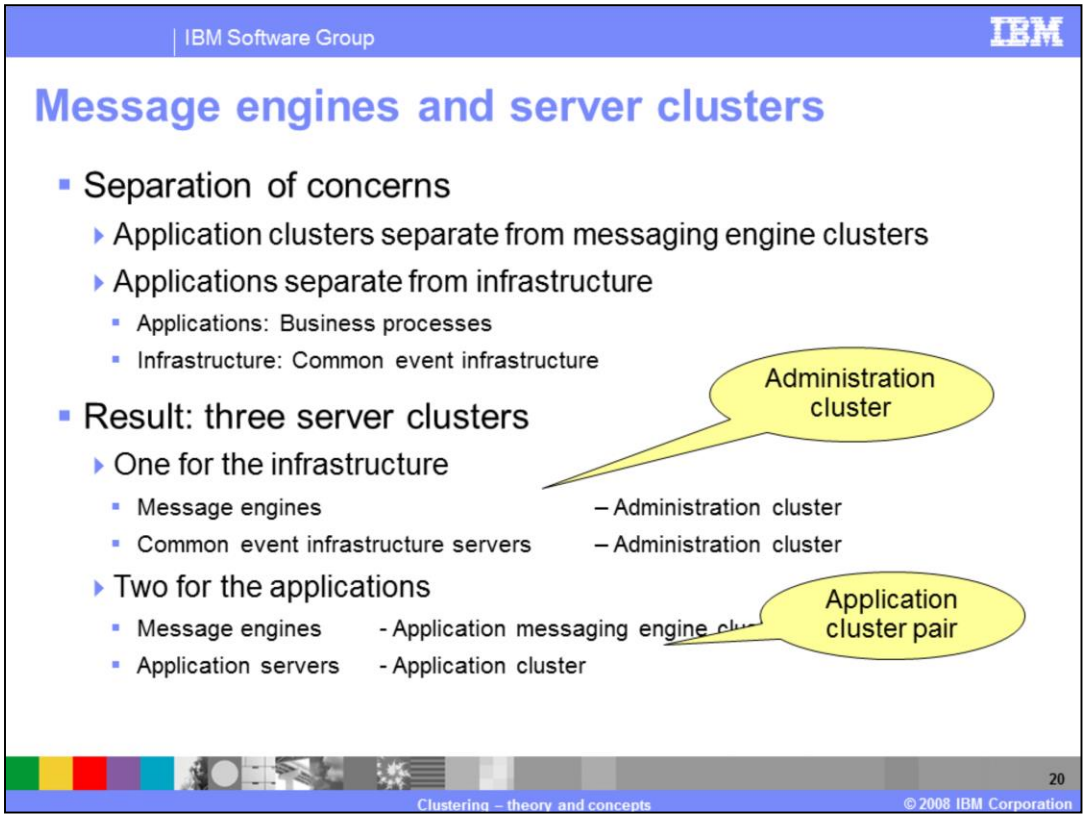
There are many facets to consider when deciding how to configure the Cell topology. Is the goal high availability or scalability or some combination of both? If so, which one is most important based on the requirements of the application?

Will the databases be remote or local?

Will there be several databases or will tables be combined into one or two databases, where it is feasible?

Ultimately it will depend on the administrative processes already in place and the quality of service required by the applications. There are many ways to configure the cell topology and WebSphere provides the flexibility to do what is needed.

Based on typical customer requirements for scalability and high availability, there are three available pre-configured topologies that are presented as a vehicle for discussing how to configure a complex WebSphere Process Server topology. These are discussed in a separate presentation.



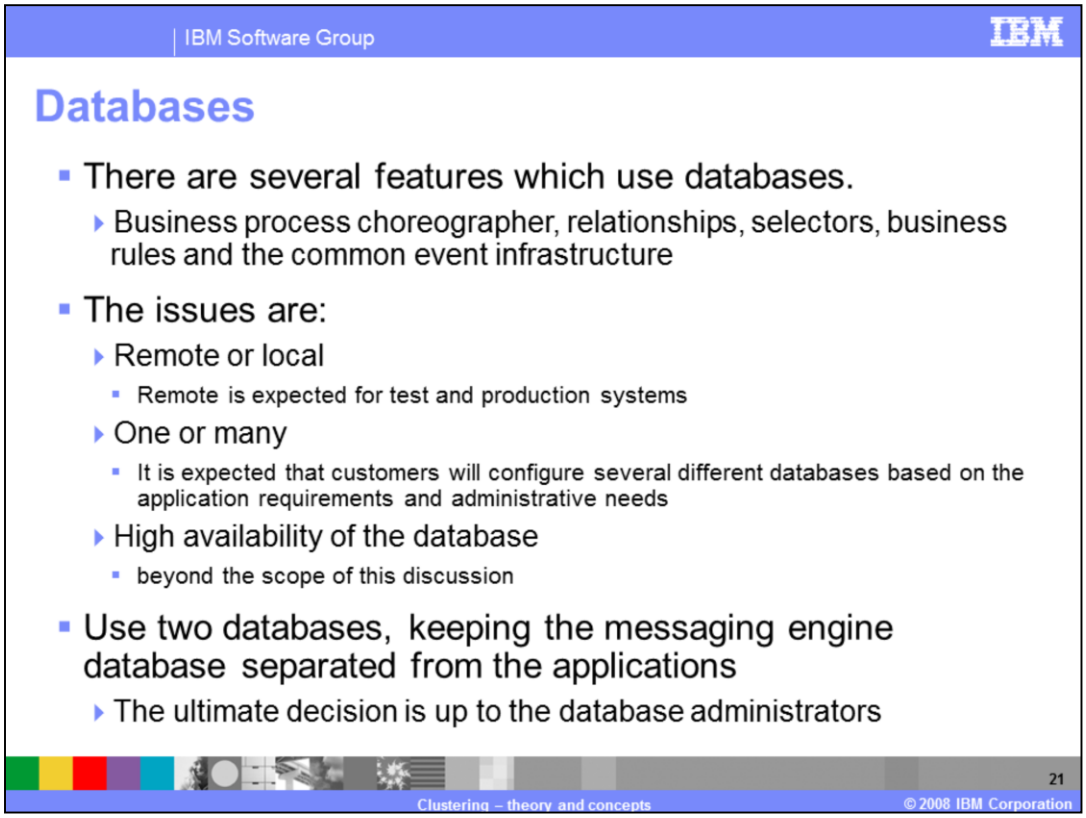
The topology being presented here is recommended as the most likely topology based on the quality of services available to the applications. It is used to demonstrate the steps for setup and configuration of a clustered topology. Once you understand the principles and the steps, you can easily develop alternative solutions.

The Common Event Infrastructure will likely be an integral part of all WebSphere Process Server applications and it is therefore considered to be an infrastructural component of the system.

The principle of “Separation of Concerns” is used as a guide for deciding how to partition the system and distribute the function. Based on the previous discussion regarding the message engine and the applications, there are message engine clusters and application clusters.

Considering functional boundaries, you can partition the system based on whether it is an administrative function or functionality associated with the end-user application. An example of a administrative function can be the Common Event Infrastructure (CEI) or the business rules manager. Both of these are WebSphere Process Server components and therefore part of the overall infrastructure.

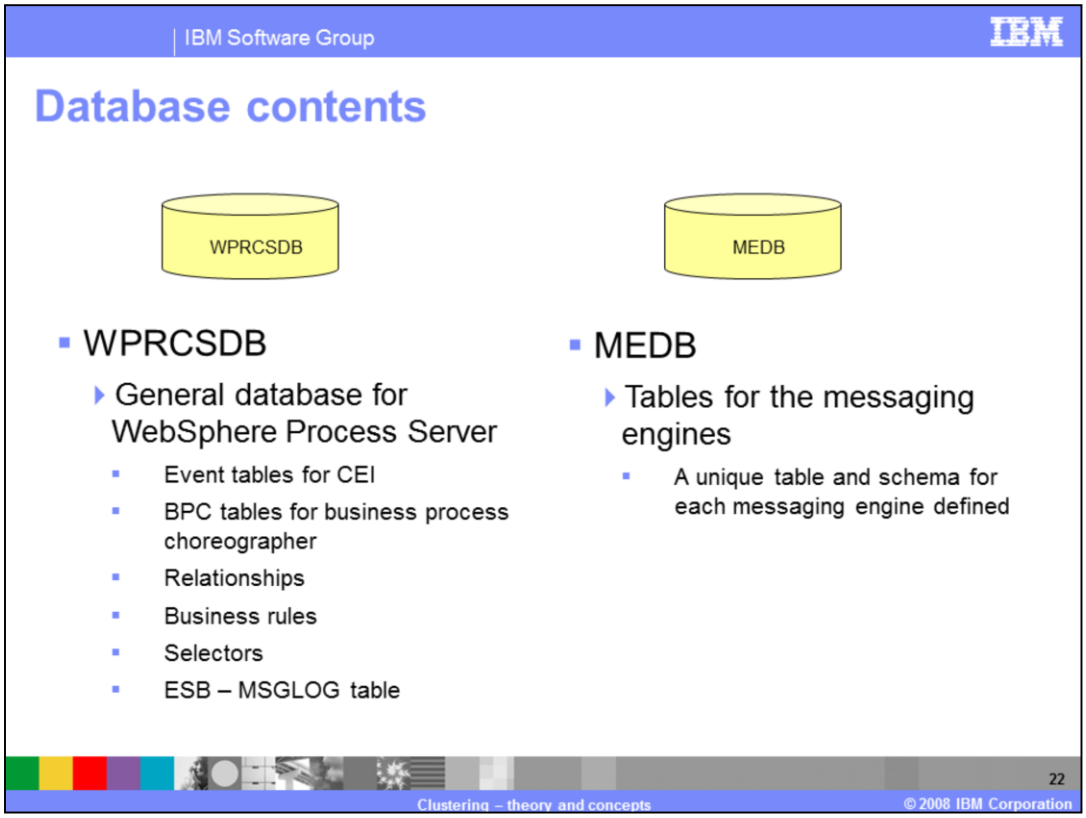
Using this criteria, the resulting topology starts with three server clusters. As the system grows there can be additional application cluster pairs, but only one administration cluster may be necessary.



With WebSphere Process Server there are quite a few databases introduced.

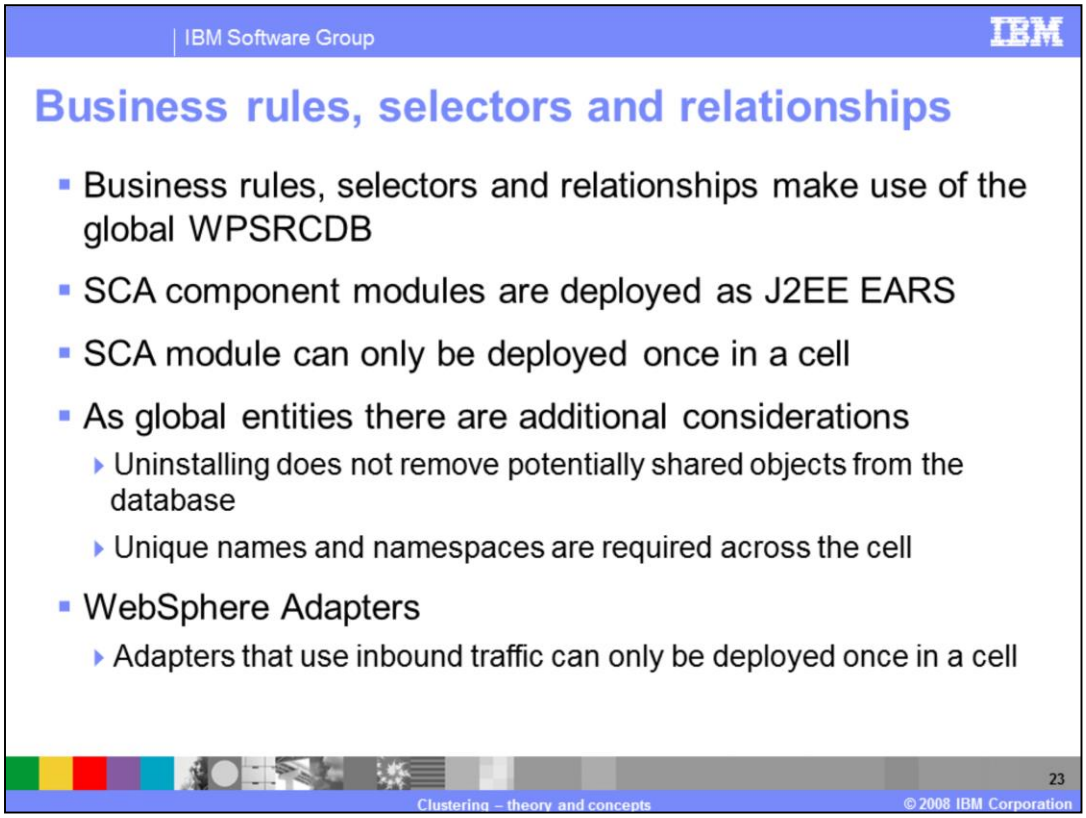
In a typical production environment there are several databases, the ESBLogMedDB, the EVENTDB, the BPEDB, the WPRSCDB, the MEDB, along with the end-user application databases. It is expected that they are remote, residing on database server, managed by database administrators.

For the configuration being presented there are two databases. The tables for the EVENTDB, ESBLogMedDB and BPEDB are created in the WPRCSDB, keeping the Message Engine data stores isolated in the MEDB. This decision is made for convenience only.



Note that the creation of the WPRCSDB before creating the Deployment Manger node (DMGR) is needed when manually creating a network deployment environment. There are scripts available that can be modified to help do this.

The decision to use two databases rather than many simplifies things for demonstration purposes.

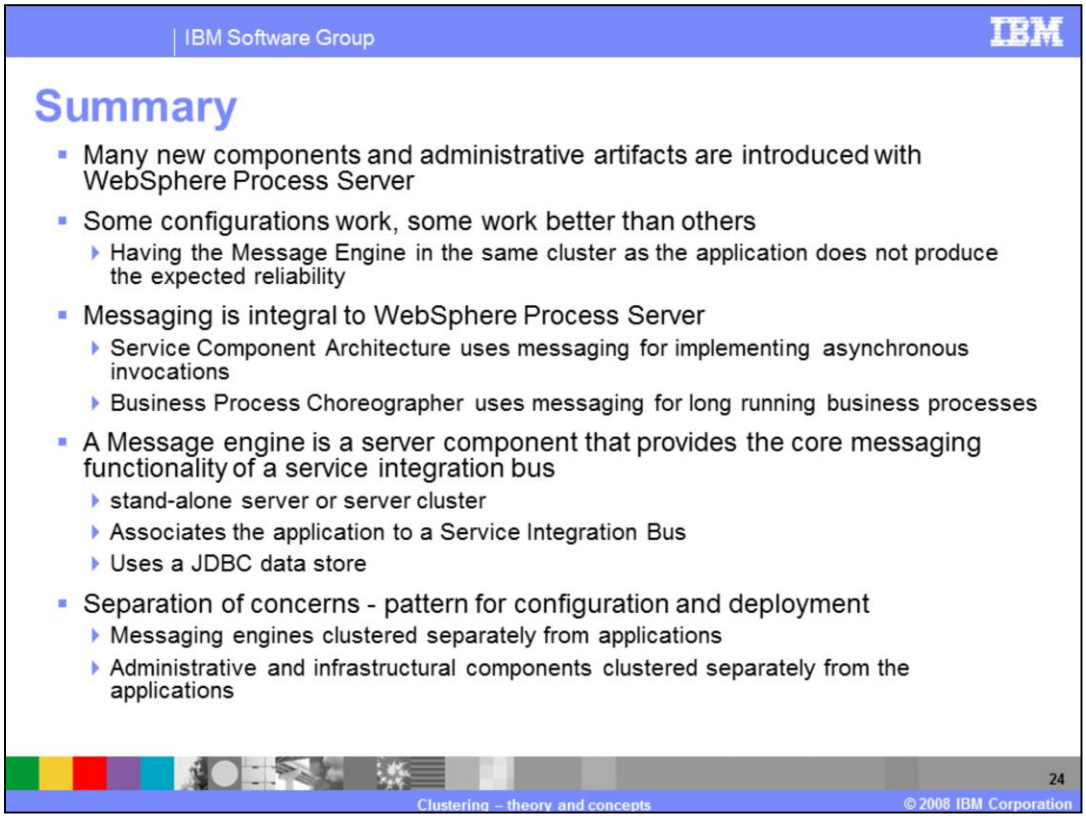


Here are a few considerations regarding the use of other Service Component Architecture components, which can impact the design of the cell, the databases and application server clusters.

The global nature of the WebSphere Process Server components presents challenges when deploying and redeploying applications.

The potential for name clashes prohibits duplicate names in many situations and the use of common components means that an entity might not be removed during uninstallation as expected, causing a deployment failure on the next deployment cycle.

WebSphere Adapters that are used for inbound traffic can only be deployed once in a cell.



Moving to WebSphere Process Server V6 introduces a lot of new parts and pieces to consider and work with.

There are many combinations for deployment and configuration, some work, some work better than others and some just do not work in a realistic manner.

The key element in business process applications and all WebSphere Process Server applications is messaging. Messaging lies at the heart of everything and when it comes to creating applications that use the WebSphere clustering capabilities the Message Engine is the component that must be considered first.

The deployment patterns discussed here has two features. First separating the Message Engine cluster from the application cluster. This is imperative for reliable messaging. The second feature presented as part of this pattern is the separation of the administrative or infrastructural components of the system from your application components. This second feature is optional and may not be necessary in all cases. For any enterprise deployment, it is the approach to use.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

DB2 WebSphere

J2EE, Java, JDBC, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.