Glassfish Architecture.

First part

Introduction.

Over time, GlassFish has evolved into a server platform that is much more than the reference implementation of the Java EE specifcations. It is now a highly configurable server that is capable of delivering high quality of service. Now, let's understand the main components of GlassFish, and how they are organized to provide a complete server environment.

GlassFish architecture with the developer profile

The developer profile of GlassFish is very light, and its architecture is illustrated in figure 1. Now let's get familiar with the two essential components of GlassFish, the server instance and the administrative domain, shown in the figure 1.

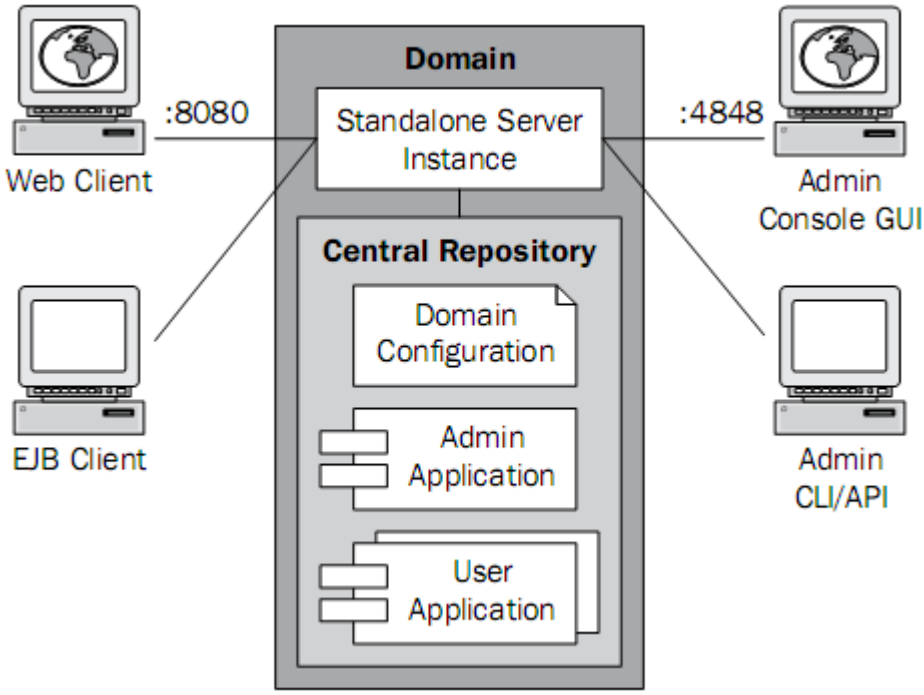

Figure 1: Glassfish Architecture

Server instances.

A server instance is a GlassFish Server running in a single Java Virtual Machine (JVM) process. The server instance implements containers for Java EE components, including the web container for web application modules and EJB container for EJB modules. In addition, the server instance provides the capability necessary for enabling client access and resource management. For example, the HTTP capability of the server instance is based on the Grizzly project , which provides a Java NIO based HTTP server that is highly scalable. The server instance hosts enterprise applications and resources. While it is usually suffcient to create a single server instance on a physical host, in certain scenarios, it can be benefcial to create multiple server instances on one physical host for

application isolation and rolling upgrades.
The server instance is fully Java EE compliant; therefore it always supports all the Java EE 5 component models and services. For example, the EJB container is always part of the server instance even if none of the deployed applications use EJB modules. Due to this, the server instance is sometimes considered heavy, and starting up a server instance could take some time. GlassFish v2 addressed the issue of startup time by starting services only when they are requested for the frst time, not when the server starts.

Administrative domains

An administrative domain (or domain) is a group of one or multiple server instances that are administered together. A server instance always belongs to a single domain, and server instances in a single domain can run on different physical hosts. Each domain has one domain administration server, and one or multiple server instances. A domain maintains its own confguration, log files, and application deployment areas. From the administration and confguration perspective, a domain represents a complete GlassFish Server runtime environment, which is responsible for hosting and managing applications and resources.

The architecture of the developer profile

The developer profile of GlassFish uses one standalone server instance in the domain. This server instance is responsible for hosting both the Admin Console application and user level business applications. The separation of the administrative applications and user applications is achieved by using different HTTP connections. By default, user web applications are accessible through port 8080 for plain HTTP, and 8181 for HTTPS. On the other hand, the administrative applications are available through port 4848 for HTTP, or 4949 for HTTPS. The developer profile of the GlassFish Server does not include the clustering feature. By running only one server instance, the developer profle requires the minimum amount of system resources, while still providing a fully Java EE compliant environment. This makes the developer proflie perfect for application development and functional testing.

GlassFish architecture with the clustering profile

It is often desirable to enable the clustering feature of GlassFish in typical production deployment to achieve better system availability, performance, and throughput. The clustering feature of GlassFish is readily available: All you need to do is to install GlassFish using the clustering profile, or update the already installed GlassFish from developer profile to clustering profile. A typical production deployment of GlassFish Server contains several additional building blocks, as illustrated in figure 2. As you can see, besides administrative domains and server instances, the clustering profle has several additional components.

Domain Administration Server (DAS)

The Domain Administration Server (DAS) of a domain is a special server instance that is typically dedicated to hosting administrative applications, such as the Admin Console web application and other administrative tools. The DAS is responsible for authenticating the administrator, handling administrative requests, and communicating with other server instances in the domain to carry out administrative tasks. All this communication is based on the Java Management Extension (JMX). The administration tools include command-line utilities and the browser-based Administration Console (Admin Console). The GlassFish Server also provides a JMX-based API and ANT tasks for server administration. The DAS maintains a repository of the confguration and applications deployed to the domain. Each server instance maintains a cached copy of the repository. Because of

this, if the DAS is not available, there is no impact on the performance or availability of active server instances. However, in these circumstances, administrative changes cannot be made.

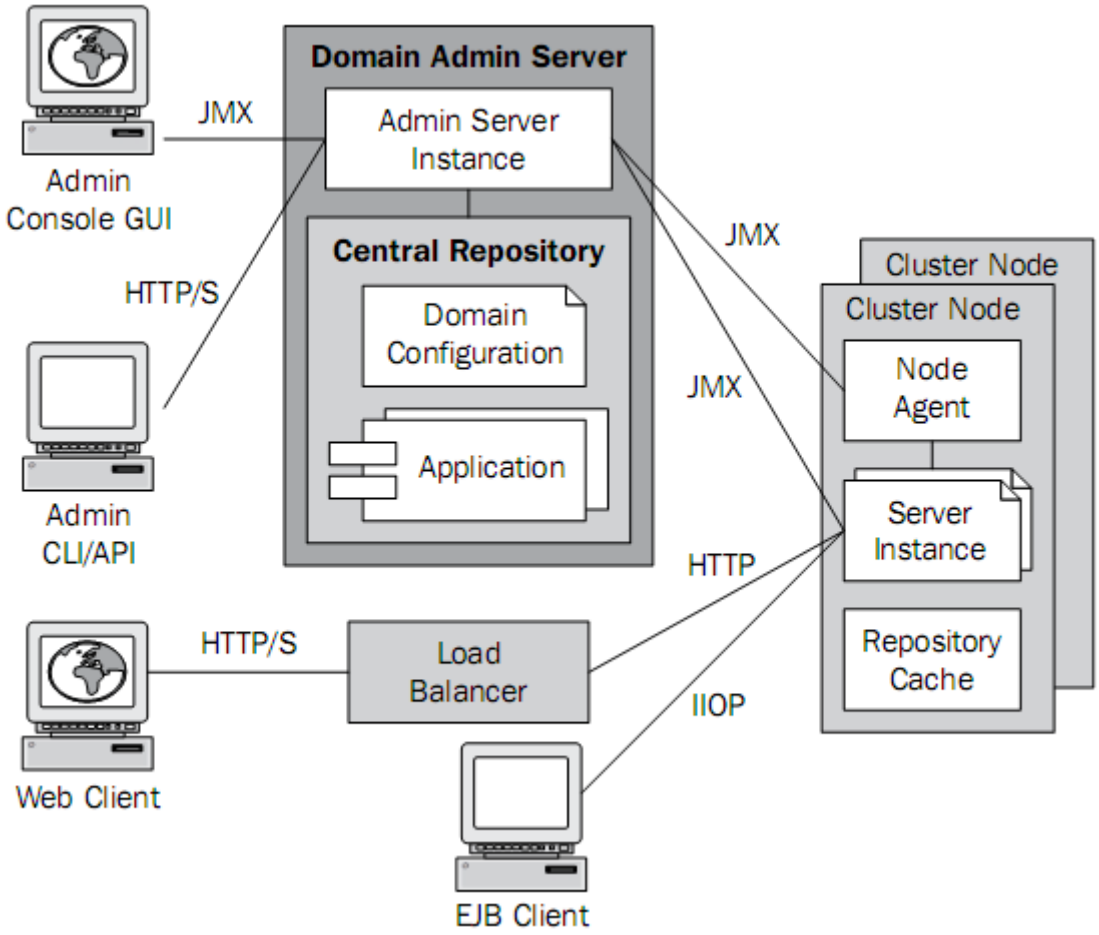

Figure 2: A typical production deployment of GlassFish Server

Clusters

A cluster is a named collection of server instances that share the same applications, resources, and configuration information. Typically, a cluster includes server instance members created on different physical hosts to improve the throughput and availability of the server deployment. This is sometimes called horizontal scalability. Server instance members of a cluster are typically configured in a load-balancing strategy to simplify the client's view of the server environment. A client does not need to know the physical location information of individual server instances. Clusters simplify server administration. A server administrator can easily control the lifecycle of each server instance, and only one confguration needs to be maintained for a cluster. In a cluster environment, when you deploy an application, you do not need to deploy it to each server instance. Instead, deploying the application to a cluster automatically pushes the application to each server instance. Furthermore, in GlassFish, additional capabilities such as session replication and failover can only be achieved by a clustered deployment.

Node agents

A node agent is a lightweight process which runs on every physical host that runs GlassFish Server instances, including the machine that hosts the DAS. The node agent is responsible for the following tasks:

- Creating server instances on the physical host, and initializing the server instance by retrieving the confguration information from the DAS repository.
- Starting, stopping, and restarting server instances as instructed by the DAS or in the event of a server instance failure.
- Providing diagnostic information about the lifecycle change events that occurred to server instances, such as a failure.
- Synchronizing each server instance's local confguration repository with the DAS's central repository.
- Performing appropriate cleanup when a server instance is deleted.

Each physical host must have at least one node agent for each domain to which the host belongs, and the node agent must always be running to control the lifecycle of server instances.

Using the clustering profile

The clustering profile not only allows multiple server instances to be used to provide more server processing power, it also provides several additional advantages. For example, the clustering profile simplifes the confguration of multiple server instances. We only need to maintain one configuration at the cluster level, regardless of how many server instances the cluster contains. Also, application and resource deployment is also easier, because they only need to be deployed to the cluster, and they are automatically deployed to each server instance. Furthermore, the clustering profile provides a light-weight, in-memory session fail over solution to improve the availability of the server environment. Due to these excellent features, we should consider using GlassFish with the clustering profile in most production environments, and also in environments where system load and stress tests are performed.

Using the enterprise profile

There is a third usage profile supported in GlassFish, that is, enterprise. The enterprise profile is quite similar to the clustering profile, wherein it supports all the GlassFish components like domains, server instances, and so on. The differences between clustering and enterprise profiles are mainly in the following areas:

- The enterprise profle uses the High Availability Database (HADB) to persist user session date, while the clustering profle essentially uses the server memory to do the same.
- The enterprise profile is not open-source; due to the license structure of HADB, the enterprise profile is not supported in the open source distribution of GlassFish. You can only get this feature by using the commercially supported distribution, the GlassFish Enterprise Server.

The following table highlights the differences of the three usage profiles.

Configuration	Developer	Clustering	Enterprise
Available in open source distribution	Yes	Yes	No
Server resource requirement	Low	High	Very high
Separate DAS	No	Yes	Yes
Use node agent	No	Yes	Yes
Support clustering	No	Yes	Yes
State replication	Not supported	In memory	Database

Next Chapter : Understanding the GlassFish deployment structure