UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ingeniería

"Implantación de arquitectura open source para servidores de aplicaciones mono núcleo"

INFORME DE ACTIVIDADES

que para obtener el título de

INGENIERO EN COMPUTACIÓN

bajo la modalidad de trabajo profesional

Presenta:

NANCY CAROLINA GUTIÉRREZ DAMIÁN

Asesor:

M.I. ALEJANDRO VELÁZQUEZ MENA

Ciudad Universitaria, México, Septiembre 2012

DEDICATORIAS Y AGRADECIMIENTOS

Las palabras nunca han sido mi fuerte pero hoy es más que menester agradecer a todas las personas que han cruzado por mi vida y me han enseñado tantas cosas. Gracias a los que colaboraron con su granito de arena.

Dedico el presente **a mi familia**, especialmente a la memoria de mi abuelo, Moisés Gutiérrez Pérez, ejemplo de perseverancia y amor por el arte y la vida.

Gracias **a la UNAM**, máxima casa de estudios. Gracias a todos mis maestros que con cariño y dedicación me transmitieron su conocimiento. A los que se fueron, a los que quedan y a los que llegan porque el poder de la enseñanza les confiere también grandes responsabilidades.

Gracias **a mis padres** por su apoyo y amor incondicional y por creer en mí. Por haberme forjado el ser humano que soy ahora, por enseñarme el camino, esperando mis logros para celebrar y mis tropiezos para alentarme a seguir adelante. La vida no me alcanzará para pagar y agradecer sus esfuerzos. Que Dios los bendiga y me los conserve muchos años. Mami, aquí va tu segunda graduada.

A mi hermano, por quererme tanto y apoyarme y cuidarme hasta el día de hoy. Rafita, espero pronto leer tu tesis. Te quiero.

A mi angelito caído del cielo, **Natalia**, porque con tu inocencia y tu vitalidad das sentido a las mañanas en casa desde el día que llegaste. Porque sigues esparciendo tu alegría y tu deseo de vivir, de saber, de conocer. Porque iluminas no sólo mi casa y mi vida sino la de todos los que te conocemos y te amamos. Dios te cuide y te bendiga. Ya aprenderás y me leerás un día del mismo modo que espero hacerlo yo también. Siempre voy a recordarte pidiendo tu "compu" para hacer tu tesis.

A mi aval, el MCC. Alberto Duarte, porque me abrió las puertas sin conocerme y luego me reafirmó su confianza al aceptar ser mi respaldo ante el presente comité de titulación. Gracias por guiarme con afecto y respeto.

A mi asesor, el M.I. Alejandro Velázquez, Coordinador de la Carrera de Computación en la Facultad de Ingeniería, por guiarme puntual, objetiva y acertadamente para estructurar el presente informe.

Gracias **a mis amigos** por todo su apoyo y por estar ahí para mí siempre dispuestos.

Y para cerrar con broche de oro agradezco al amor de mi vida, el **Lic. Luis David Zúñiga**, por ser parte de ella desde hace más de una década. Te admiro y respeto. Y cumpliendo con los requisitos en tiempo y forma, te presento con cariño este documento previo al sine qua non solicitado y que está más cerca. TE AMO y el título por el que me presento hoy, va por ti.

Índice

Introducción

El siguiente informe lo centré en el estudio de los servidores de aplicación mono núcleo multi *cluster,* enfatizando la estructura de los servidores actuales utilizados en las aplicaciones que se encuentran en desarrollo (pruebas y producción) en la empresa.

En el marco de los sistemas informáticos las primeras aplicaciones o programas tenían su propia interfaz y su propio servidor que eran independientes uno de otro en cada ordenador, actualmente las aplicaciones web desempeñan un papel de suma importancia, ya que, gracias a los avances tecnológicos, la evolución de los mismos implica un crecimiento basado en la mediación de un servidor web o un servidor de aplicaciones a través de un navegador.

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http [1]. Los servidores de aplicación se distinguen de los servidores web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

Los servidores de aplicaciones son dispositivos que dan servicio de aplicaciones a los clientes, cualquier dispositivo con acceso a la red, que direccionen alguna de las aplicaciones contenidas en él y que cuenten con las interfaces necesarias para la visualización de las mismas. P ej. Flash Player o cualquier plug-in [2].

Esta es la diferencia de arquitecturas de los desusados servidores Web frente a los servidores de aplicaciones, ver figura I.1.

En la parte superior se observa la arquitectura del servidor web que es más sencillo presentando sólo dos capas; una el navegador, que es el cliente y otra el contenedor web, que es el servidor. Abajo la estructura tricapa del servidor de aplicaciones que cuenta con una tercera capa que sirve de interfaz entre los recursos de la aplicación y el cliente.

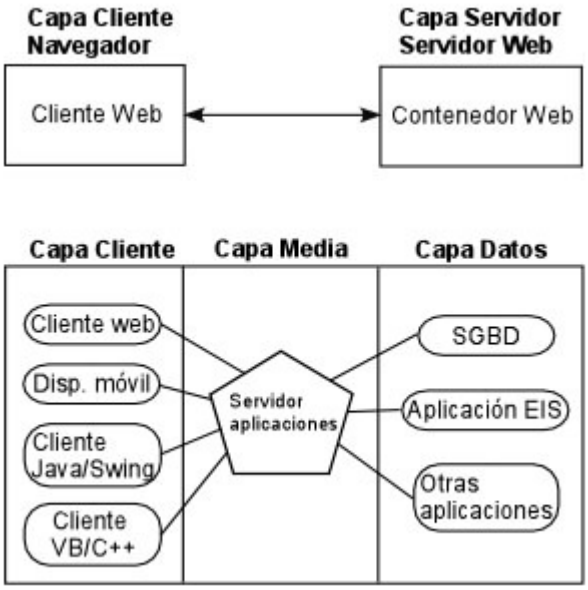

Figura I.1. Arquitectura del servidor web (dos capas) &
servidor de aplicaciones (tres capas)

Cabe destacar que la necesidad de los servidores de aplicaciones está en el desarrollo de aplicaciones empresariales robustas, que implican el uso de algo más que *servlets* y *JSP's*. Sin embargo, existe la posibilidad de que algunas aplicaciones no necesiten más que un *servidor web* para funcionar, dada su sencillez.

Los servidores de aplicación en la actualidad proporcionan un soporte robusto y adecuado para cualquier tipo de aplicación en casi cualquier nivel de desarrollo y despliegue, es por eso que el uso de un sólo núcleo en un servidor, es decir un sólo servidor, se ha convertido en algo exiguo, es ahí donde entra la arquitectura en *cluster* para los servidores de aplicaciones mono núcleo, lo que permite dar una mayor estabilidad a las aplicaciones y un mejor aprovechamiento de los recursos del mismo.

La implementación en *cluster* implica el uso de más de un servidor (físico o virtual) pero en un arreglo por software tal, que permita manejarles centralizadamente (desde un servidor principal al que llamamos administrador) a lo que hemos denominado mono núcleo, con las ventajas de tener varios servidores atendiendo peticiones de modo que si alguno presenta problemas en operación, los demás pueden entrar en acción (por medio del balanceo de cargas) para dar continuidad al servicio y eficientar el uso de recursos en una empresa.

En el capítulo tercero de este informe, llevaré paso a paso al lector hasta la consecución de un sistema de dos servidores de aplicaciones funcionando de manera conjunta (uno como administrador, el otro como cliente -del servidor administrador-) para crear un *cluster* operativo de servidores de aplicaciones J2EE con la versión libre más reciente del conocido GlassFish, la versión 3 .1. 2.

Anteriormente los servidores en *cluster*, eran administrados desde la consola y manualmente como complemento.

Este era el panorama de instalación y administración en las pasadas versiones.

Dos servidores (físicos/virtuales), uno de ellos es el principal –o administrador- el otro es el secundario –o cliente-. La instalación del software del servidor de aplicaciones GlassFish de versión inferior a la 3.1.2 debe hacerse en ambos servidores cuidando que el mapeo de los path de instalación en ambos sea igual, es decir; si se instala en el servidor1 en la unidad **X**: en la carpeta **GlassFish_v_xx,** debe crearse una carpeta IDÉNTICA en el servidor2, en la misma unidad, para dicha instalación.

Se usarán los mismos puertos (en un mismo ordenador es imposible usar un mismo puerto en 2 cosas diferentes. En este caso los servidores son independientes por lo que podemos, y más que eso debemos, configurar los mismos puertos) para que al hacer la comunicación desde internet con los servers, y si cumple con las características de conexión de seguridad configuradas en el firewall, se llame a la aplicación desde uno u otro servidor sin contratiempos.

Una vez instalados ambos servidores, sólo se configuraría el servicio de Windows para la consola de administración en el servidor elegido como administrador, por ejemplo, el servidor1.

Por medio del servicio creado, al iniciarlo, podremos llamar al servidor mediante la URL que estará compuesta del nombre de nuestro servidor seguida de **:** (dos puntos) y el puerto que hayamos fijado para la consola de administración, por ej. 4848 (que es el valor predeterminado). Así escribiríamos en el navegador algo semejante a **http://servidor1:4848**

Eventualmente es necesario copiar algunas librerías (en la carpeta **lib** del directorio **glassfish** que se crea al instalar el software) de los diversos proveedores de bases de datos -por ejemplo Oracle, Sybase, etc.- a fin de que estén disponibles para la creación de pools de conexiones para las diversas aplicaciones, que así lo requieran.

Se deben configurar los nodos (que son como particiones del servidor) para conectar ambos servidores (servidor1 y servidor2) en *cluster*. Esto permite que ambos servidores, según la configuración de atención de peticiones (balanceo de carga) despachen las solicitudes de manera tal que el usuario final (en la red) tenga continuidad en el servicio.

Una vez en la consola (que instalamos en servidor1) se pueden desplegar aplicaciones y crear los recursos, (JNDI, pool de conexiones, JMS's, etc) si se requieren.

Es importante resaltar que los recursos son compartidos -se crean en la consola para ambos servidores (lo que significa que comparten configuración)-; *mientras que las aplicaciones se instalan en la consola administradora y deben ser replicadas en el servidor secundario mediante la copia de la carpeta de aplicación respectiva en cada caso.*

La última versión de GlassFish es diferente no sólo en cuanto a la administración, que ahora es simultánea, por así decirlo, pues ya no se requiere la copia manual de las aplicaciones; también en instalación, pues la creación de los clusters se hace comunicando a los servidores a través de SSH, lo que hace más confiable el sistema.

Finalmente mostraré las ventajas que representó instalar la última versión de GlassFish, en la administración y la atención a los usuarios. En ambos rubros se disminuyeron los tiempos. En la parte administrativa las ventajas son para los administradores pues se optimizan tiempos que pueden ser dispuestos a otras empresas, distribuyendo de forma más eficiente la jornada laboral; en cuanto a los usuarios se disminuye el tiempo de espera para la actualización o carga de las aplicaciones y se reduce el tiempo en que las aplicaciones están fuera de línea en la red, tanto interna como en la www.

CAPÍTULO 1 Participación profesional

El presente informe lo redacté en base a mis labores desempeñadas durante un año como analista de sistemas, especialmente en la administración de servidores de aplicaciones. Me desarrollé en el sector bancario.

Colaboré en el área de tecnologías de la información, específicamente en la Subgerencia de Desarrollo de Sistemas de Gestión Operativa perteneciente a la Dirección de Tecnologías de la Información.

1.1 Panorama general de colaboración

A mi llegada en enero de 2011, la empresa había atravesado recientemente un proceso de actualización y migración de servidores y aplicaciones. El motivo de mi incorporación a la empresa fue que debía administrar los servidores y dar continuidad al proceso de migración de algunos otros.

La actualización fue importante no sólo para quien administra los recursos de red (en especial los servidores), pues implementaron la virtualización que optimizaría dichos recursos. También fue importante para mí como administrador propiamente de los *servidores de aplicaciones*. Desde la instalación del mismo software, pasando por la implantación de aplicaciones y el análisis de las bitácoras en casos de fallo (tanto de configuraciones propias del servidor como del desarrollo de las aplicaciones) hasta el monitoreo en general.

Un ejemplo de las ventajas de la actualización de servidores es el que se presentaba en la versión con la que se venía trabajando hasta entonces. Con el servidor GlassFish V 2.1, las aplicaciones eran desplegadas en bloque, como una carpeta comprimida (en un archivo .ear o .war) y el servidor se encargaba de desempaquetar (o “explotar”) internamente y colocar dichos paquetes en una carpeta (aplicación) que colocaba en una ruta específica ubicada, según la configuración, en un path determinado. Dado el esquema anterior, cualquier mínimo cambio en el paquete (.ear

o .war) requería una actualización total de la aplicación, lo que en algunos casos especiales de aplicaciones muy complejas se reflejaba en mucho tiempo de actualización y por tanto mucho tiempo fuera de línea –para la aplicación en la red-.

Con la nueva versión, se puede suministrar al servidor la fuente descomprimida lo que permite hacer pequeñas modificaciones de la aplicación directamente en alguna ubicación específica optimizando así los tiempos tanto de despliegue como de disponibilidad de las aplicaciones pues de esta manera el servidor debe, solamente, refrescar el cargado de la aplicaciones y en algunos casos es automático el cambio, es decir que se requiere simplemente la sustitución de los archivos.

Mis tareas en la empresa como analista de sistemas fueron diversas, en su mayoría como administrador de servidores de aplicaciones. Las más frecuentes eran las relativas al despliegue y actualización de aplicaciones en los servidores, esta actividad la realizaba a lo largo de la jornada. Ésta podía incluir la creación y/o actualización de los recursos utilizados por las aplicaciones en el servidor.

Como administrador de los servidores de aplicaciones, también fue mi deber solicitarlos con los requerimientos precisos y mantenerme en contacto con los responsables de la preparación y entrega para, una vez recibidos, verificar que cubrían las necesidades y tenían las configuraciones solicitadas. Una vez comprobado el estado de los servidores recibidos, me correspondía prepararlos para la instalación del software en cuestión, solicitar las conexiones de red necesarias, las configuraciones de seguridad, etc. para finalmente conformar el servidor de aplicaciones requerido en tiempo y forma por las diversas áreas usuarias del servicio.

También resolvía eventualidades en las que las aplicaciones dejaban de funcionar o lo hacían de manera errática o errónea por medio del estudio y análisis de las bitácoras generadas por el servidor y/o en algunos casos las bitácoras de las propias aplicaciones. Esta tarea de análisis de bitácoras puede parecer muy sencilla, una vez que uno ha tomado familiaridad con el tipo de reportes generados por los diversos errores. Pocos son muy claros como en el caso de la falta de creación de recursos donde específicamente el servidor manda a la bitácora el nombre del recurso faltante y es relativamente fácil su creación y por tanto la aplicación funciona de manera correcta. En otras ocasiones hay que romper paradigmas e ir más allá en busca de la aguja en el pajar. Cuando uno lee por primera vez las excepciones arrojadas no hay nada claro ni obvio. Son líneas y

líneas encabezadas por la fecha y hora con indicios del problema pero sólo eso, indicios. Recuerdo a manera de anécdota, una ocasión en la que la aplicación fue actualizada y así mismo su pool de conexiones. La aplicación se cargó de manera correcta pero no hacía las conexiones esperadas a base de datos. El pool de conexiones (que hace las veces de interfaz entre la aplicación y las bases de datos) tenía incorrectos (exactamente volteados) los datos del nombre de la base y el nombre del servidor en que ésta residía. Busqué en las bitácoras y el único error legible era que no podía establecerse la conexión mediante ese pool, distinguido por su JNDI name (un identificador único de este tipo de conexiones). El usuario revisó los datos de su configuración una y otra vez, se comprobó que tuviera la configuración solicitada por el usuario pero nada, no había conexión. Entonces comencé una búsqueda más minuciosa pero no muy científica. Contacté al administrador de la base de datos para preguntarle de la existencia de dicha base. Fue así como llegué a la solución; el usuario había proporcionado mal los datos y con sólo cambiar el nombre del servidor y de la base de datos, la aplicación funcionaba correctamente. Y este es sólo un ejemplo de las singulares eventualidades a las que podía enfrentarme en el día a día poniendo a prueba no sólo mi capacidad de análisis científico basado en los hechos sino mi capacidad de escrutinio para hallar, lejos de la información tangible, la solución real al problema.

Apoyaba a los usuarios en pruebas de concurrencia a las aplicaciones y en algunos casos también pruebas conjuntas con otras áreas para probar vulnerabilidades y desórdenes de las aplicaciones en desarrollo o el mismo desempeño de las aplicaciones ante probables eventualidades, en este caso, inducidas.

Monitoreaba periódicamente la salud de los servidores de aplicaciones, verificando el estado del CPU utilizado y, a nivel servidor, monitoreaba cada nodo para verificar su correcta función. Estas operaciones de monitoreo las realizaba tanto periódicamente, gracias al sistema de alertas propio configurado en los servidores, como bajo demanda de los usuarios (que son los administradores de cada sistema) quienes recibían reportes propios mediante sus bitácoras.

También brindé asesoría y soporte a los usuarios relativa al uso de los recursos del servidor por las aplicaciones.

Mi objetivo principal era mantener operativos los servidores de aplicaciones, monitoreando para, oportunamente, intervenir asegurando así, la continuidad del servicio que

siempre es importante, pero de manera periódica es imprescindible su funcionamiento exacto y preciso en tiempo y forma debido a requerimientos específicos de la empresa.

Actualmente comienzo un nuevo proyecto de automatización de tareas administrativas en el ambiente de Windows Server.

1.2 Organigrama

La estructura organizacional de la empresa está separada por especialidades. Mi puesto fue el de **Analista de sistemas**, bajo la dirección del líder de Especialidad respectivo en la Subdirección de Desarrollo de Sistemas de Gestión operativa perteneciente a la Gerencia de Informática en la Dirección General de Tecnologías de Información, como se muestra en la figura 1.1.

Como Analista de sistemas es obligación precisamente "analizar"; escudriñar en los sistemas para conocerlos de manera que puedan mejorar, arreglar, ampliar etc. Cubriendo siempre las necesidades y requerimientos solicitados. Opcionalmente se pueden diseñar y desarrollar sistemas, nuevos o preexistentes.

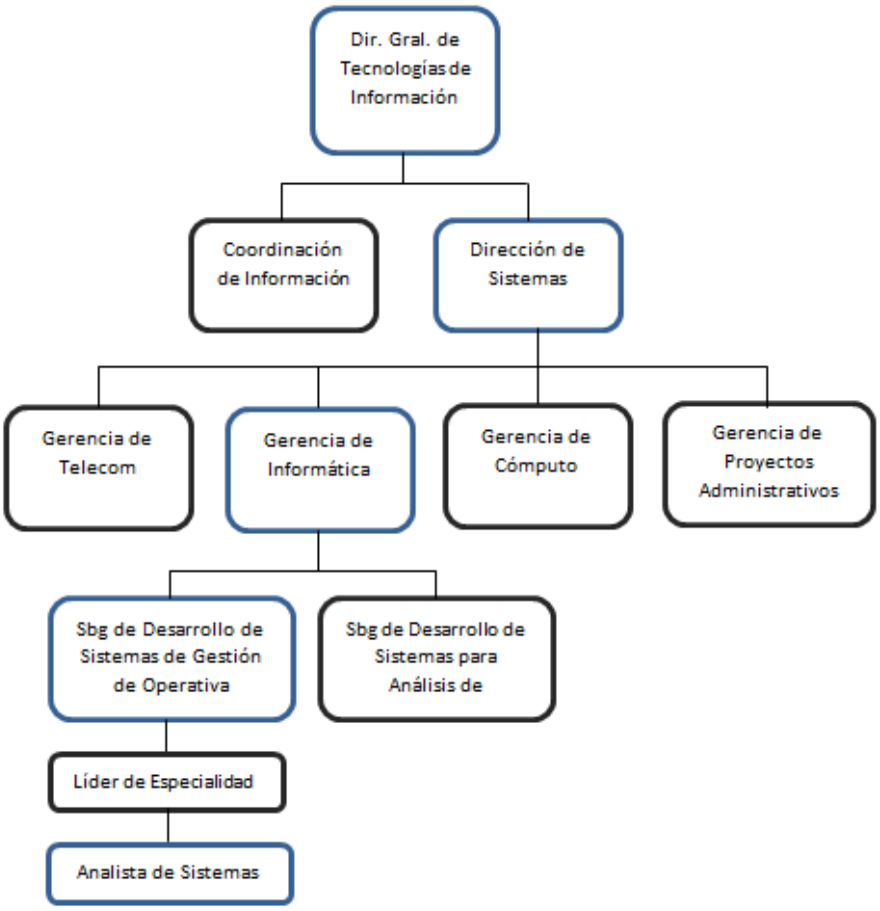

Figura 1.1 Estructura organizacional del área en que me desarrollé en la empresa.

CAPÍTULO 2 Implantación de arquitectura open source para servidores de aplicaciones mono núcleo.

2.1 Objetivos

-Dada la administración y el tipo de requerimientos de los desarrolladores de la empresa, es necesario diseñar una arquitectura de servidores tal, que permita la optimización de recursos tanto materiales como tecnológicos.

-Se eligió implementar un servidor de aplicaciones en *cluster*, conformado por un par de servidores virtuales. Uno de ellos el administrador y el otro el cliente para actuar como un sólo ente (se creará una guía paso a paso para la instalación del servidor elegido).

Esto se consiguió por medio del análisis y la observación del comportamiento de aplicaciones y servidores en la práctica diaria. Se instalaron y probaron otras versiones del servidor a fin de conocer sus ventajas y desventajas llegando hasta la elección de la versión 3.1.2 de GlassFish.

2.2 Marco teórico

En el mercado existe una amplia variedad de servidores de aplicaciones, en este caso me refiero a aquellos relacionados con aplicaciones J2EE que son aplicaciones java.

Anteriormente la empresa trabajaba sobre servidores de aplicaciones propietarios que requerían el uso forzoso de servidores físicos. La adquisición del software estaba acompañada de licencias que limitaban el número máximo de procesadores del servidor. La implementación de servidores físicos hoy en día y a pesar de su excelente desempeño, implica un uso exacerbado de recursos en cuanto a espacio y costos en relación con los de la virtualización. Respecto a la limitación de procesadores que mencioné, cabe destacar que es imposible la mejora en el desempeño del servidor físico, en cuanto a procesamiento, si no se cuenta con las licencias suficientes.

Es así como se comenzó a poner ojos en un joven pero bien respaldado servidor de aplicaciones de código libre. Hablo de GlassFish, que vio la luz en la empresa unos años atrás dando respuesta a las necesidades tanto operativas como de reducción de costos del lugar. Con la implementación de GlassFish opensource en su última versión 3.1.2 como *cluster* se alcanzan las ventajas del anterior sistema WebLogic.

Una vez superado el hecho de la necesidad de servidores físicos, se dio paso a la virtualización, un proceso en el cual se pueden soportar, con un equipo físico, a N equipos virtuales independientes en un sólo equipo físico.

2.3 Concepto de servidor de aplicaciones

Se denomina servidor de aplicaciones a la implementación del software (así llamado) que permite la administración de aplicaciones en él montadas para su utilización funcional como aplicación. Es el corazón de un gran sistema distribuido.

El concepto de sistema distribuido está muy relacionado con el de servidor de aplicaciones. A diferencia de un sistema monolítico, el distribuido, permite mejorar los tres aspectos más importantes y fundamentales en una aplicación: alta disponibilidad, escalabilidad y mantenimiento [3].

La alta disponibilidad se refiere a un sistema que trabaja todo el tiempo (24x7), para ello se implementa en los sistemas el *balanceo de cargas* y la recuperación ante fallos (o failover).

La escalabilidad se refiere a la capacidad de crecimiento que tenga el sistema ante el aumento de la demanda, para ello, los sistemas tienen la capacidad de aceptar un aumento de recursos.

El mantenimiento se refiere a que el sistema subsista con el paso del tiempo mediante depuraciones, actualizaciones y otras actividades propias de la administración del mismo.

En el caso de los sistemas monolíticos, como su nombre lo indica, son sistemas “cerrados” en los que cualquier cambio podría conllevar a resultados catastróficos.

En una red de computadoras ejecuta las aplicaciones en él montadas o instaladas. Cuando se habla de aplicaciones desarrolladas en java entonces tenemos servidores de aplicaciones J2EE, que refiere al estándar bajo el que están desarrollados sobre la plataforma java. La arquitectura J2EE es la que aparece en la figura 2.1

En la figura podemos ver los conceptos de contenedor de applets, aplicación cliente, contenedor web, entre otros.

El contenedor de applets, también conocido como cliente web es un navegador que interactúa con el contenedor web mediante HTTP. Es capaz de recibir páginas en HTML y XML y ejecutar applets y JavaScripts.

La aplicación cliente, son clientes que no se ejecutan dentro de un navegador utilizando cualquier tecnología para comunicarse con el contenedor web o directo a la BD.

EL contenedor Web o servidor web es la parte “visible” del servidor de aplicaciones que utiliza protocolos de comunicación tales como HTTP, SSL, etc.

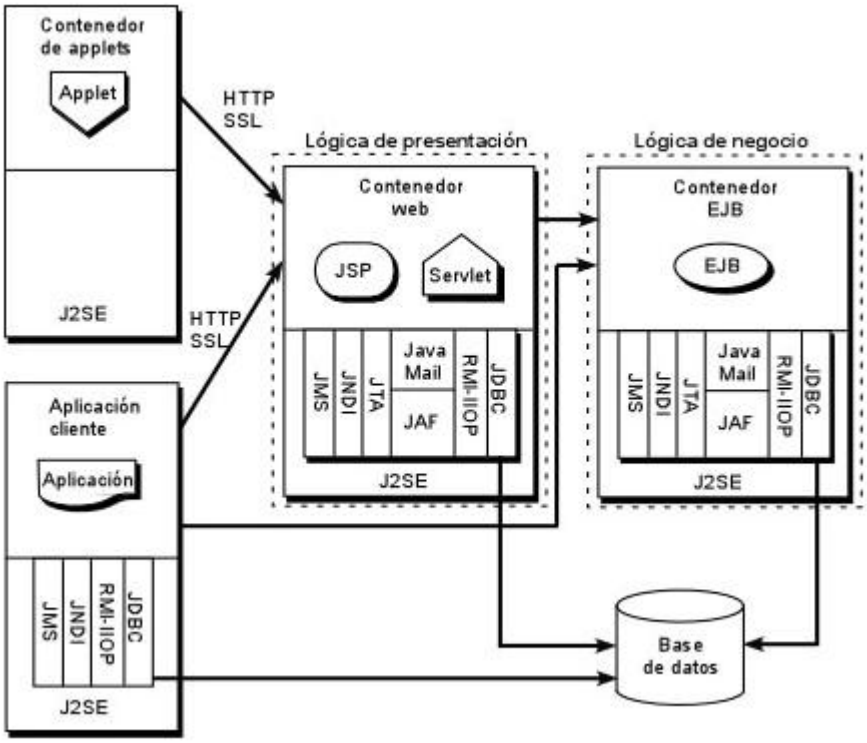

Figura 2.1 Arquitectura J2EE

Entre los servidores de aplicaciones J2EE (o Java EE) podemos encontrar una variedad entre libres y privativos. Podemos mencionar por ejemplo: Base 4 Server, Zope, BEA WebLogic, Oracle IAS y por supuesto GlassFish. Hablaré de la versión libre de Oracle, GlassFish [6].

2.4 Generalidades de GlassFish y WebLogic

GlassFish [7] es un servidor de aplicaciones de software libre, lo que implica, entre otras cosas, que no requiere pagarse licencia alguna para su uso.

Fue desarrollado originalmente por Sun Microsystems, compañía que sería absorbida por Oracle Corporation en 2009.

La Versión más actual de GlassFish es la 3.1.2, liberada en febrero del presente año. En esta empresa trabajé con las versiones 2.1, 2.1.1, 3.1.1 y la actual 3.1.2.

GlassFish es relativamente nuevo en el campo de servidores y el haber sido desarrollado por una compañía líder por el desarrollo del lenguaje Java nos indica que la compatibilidad entre estas tecnologías es total, además de tener todo el apoyo de la comunidad java por detrás. [8]

Por otro lado WebLogic Server utiliza tecnologías de la plataforma Java 2, Enterprise Edition (J2EE). J2EE es la plataforma estándar para desarrollar aplicaciones multi-capa basadas en el lenguaje de programación Java. Las tecnologías del componente J2EE fueron desarrolladas colaborativamente entre Sun Microsystems y otros vendedores de software entre los que se incluye BEA Systems. Las aplicaciones J2EE están basadas en componentes estandarizados y modulares. [9]

WebLogic es líder en el procesamiento de transacciones online plataforma (OLTP), desarrollada para conectar a los usuarios en un entorno de computación distribuida y para facilitar la integración de las aplicaciones de mainframe con distribuidas de datos y aplicaciones corporativas. [10]

2.5 Análisis de factibilidad del software

Me fue encomendada la labor de administrar los servidores de aplicaciones web. Para ello no sólo se incluyó la parte operativa de implementación de aplicaciones desarrolladas por personal de la misma empresa sino la gestión continua de la operatividad de los mismos.

De esta manera, me fue posible conocer los sistemas desde varias perspectivas, lo que me permitió observar su desempeño en el día a día haciendo, finalmente, posible el cambio con el que se concluyó mi participación en el proyecto.

Para llegar a la versión finalmente instalada, tuve que implementar varias versiones tanto en pruebas como en ambientes productivos a fin de evaluar el desempeño de servidores y aplicaciones.

En base al desarrollo de nuevas aplicaciones y tecnologías internas y con ayuda de los ambientes de pruebas con que cuenta la empresa, me fue posible analizar las deficiencias en la operación de tales aplicaciones y el desempeño de los servidores de aplicaciones, llegando a concluir que se debían a deficiencias que podían subsanarse con la actualización del servidor.

Por medio de la observación y análisis y con pruebas reales del desempeño de las diferentes versiones, finalmente llegué a la instalación de la más conveniente, GlassFish V 3.1.2 [11], para nuestros sistemas y lo instalé en *cluster* para satisfacer las nuevas necesidades. Dando como resultado un mejor manejo y aprovechamiento de recursos, tanto humanos como tecnológicos.

2.6 Desarrollo.

2.6.1 Entorno

El SO sobre el que se monté el servidor de aplicaciones es Windows Server 2008 R2 [12] Enterprise Edition, que está basado en el núcleo Windows NT y es el sistema operativo diseñado por Microsoft para la administración de servidores.

Windows NT es un sistema operativo completo y autónomo para microordenadores de altas prestaciones y estaciones de trabajo. Está dirigido, mayoritariamente, al mismo público del sistema operativo Unix. Es, por tanto, un sistema multitarea y multiusuario, que puede competir en el mercado de los sistemas operativos para estaciones de trabajo y para servidores de red, pero además ofrece una interfaz de usuario basada en el popular Windows 3.1, lo que lo hace mucho más amistosa que la interfaz de Unix. Con este sistema, plataformas de hardware muy diferentes pueden compartir la misma interfaz, es así como es capaz de ejecutar aplicaciones diseñadas para diversos sistemas operativos. [13]

Como mencioné en el punto 2.3, los servidores de aplicaciones tienen las ventajas de los sistemas distribuidos. La alta disponibilidad se logra por medio del *balanceo de cargas*, que es una técnica utilizada para distribuir las peticiones que llegan a los servidores, de forma que ambos atiendan simultáneamente y a la vez, que en caso de fallo de uno, el otro entre a atender las peticiones para que el servicio sea continuo[14].

Instalación en cluster.

Por lo anterior es útil la implementación de los servidores asociados mediante clustering que es una forma de configurar los servidores que permite asociar, al menos 2, servidores de forma que ambos actúen como una instancia única que permite el balanceo de carga y técnicas de failover. A dicho arreglo o configuración se le conoce como *cluster.*

Los clusters permiten aumentar la escalabilidad, disponibilidad y fiabilidad de los sistemas en la red.

Ya hemos hablado de escalabilidad y el concepto aplicado a clusters es el mismo; en cuanto a la disponibilidad y fiabilidad son conceptos relacionados pero diferentes; mientras la disponibilidad es la calidad de estar presente, listo para su uso, a mano o accesible, la fiabilidad es la probabilidad de un funcionamiento correcto. [15].

En este caso se trata de dos servidores. Uno el administrador *DAS* (Domain Admin Server) donde se lleva la administración de las aplicaciones en ellos instaladas; y el secundario o cliente en donde se replican las acciones tomadas en el *DAS*. La instalación en cluster presenta ventajas operativas en cuanto a la administración y balanceo de cargas asegurando la mayor disponibilidad para el despacho de peticiones en ambos.

2.6.2 Versiones

El servidor de aplicaciones GlassFish ha evolucionado desde sus inicios en 2005 cuando fue lanzado por primera vez por la compañía Sun Microsystems que sería comprada por Oracle más tarde.

Casi desde sus inicios fue diseñado basado en la plataforma java, lo que lo hace ideal si se pretende administrar aplicaciones desarrolladas en ese lenguaje.

Comencé administrando y utilizando GlassFish V 2.1, pero dado el desarrollo de nuevas tecnologías y funcionalidades de las aplicaciones, con el tiempo, fue necesaria la actualización para que dicho servidor de aplicaciones fuese compatible con algunos nuevos desarrollos por lo que actualicé a la versión 2.1.1.

Hoy en día se ha liberado la versión 3.1.2 que aporta ventajas operativas que representan mayor libertad y confianza en la administración de las aplicaciones java 2EE en él instaladas.

2.6.3 Justificación de las versiones

El motivo principal para buscar la actualización del GlassFish V 2.1 con el que se había venido trabajando fue la necesidad de usar nuevas facilidades que, en la práctica diaria, los mismos desarrolladores encontraron en las nuevas versiones de GlassFish, se vio la factibilidad del uso de la versión 3.1.1, pero ésta fue insuficiente (en un caso específico en que se requería conectar Web Services protegidos por certificados). Fue así como, más a pruebas y error que científicamente, encontré que la versión 3.1.1 no era la adecuada y al probar la nueva versión 3.1.2 vi que el bug de la versión anterior estaba superado.

2.6.4 Arquitectura GlassFish V 3.1.2 Open Source

GlassFish está basado en el código fuente donado por Sun y Oracle Corporation, éste último proporcionó el módulo de persistencia TopLink. GlassFish tiene como base al servidor Sun Java System Application Server de Oracle Corporation, un derivado de Apache Tomcat y que usa un componente adicional llamado Grizzly y Java NIO para escalabilidad y velocidad. [13]

Para la implementación de esta versión fue necesario, no sólo la instalación del JDK, también se requirió el uso de una herramienta de comunicación segura *SSH.* Anteriormente la comunicación se llevaba a cabo a partir de la configuración en el *dominio,* del *cluster,* lo que hacía que se compartieran los recursos (como las colas y destinos JMS o los pools de conexiones) pero que la administración de las aplicaciones fuera manual, es decir la copia y borrado de las carpetas (que son fuente de las aplicaciones) en los servidores no era automática y mucho menos simultánea.

El uso de *SecureShell* nos permite la intercomunicación de los servidores a través del demonio y hace que la administración, en cuanto a recursos y aplicaciones se refiere, sea simultánea.

2.6.5 Instalación de JDK

Para la ejecución de GlassFish V 3.1.2, fue necesario instalar un kit de desarrollo Java con versión mínima 1.6.25 (según las recomendaciones de Oracle), instalé la versión 1.6.31.

Por conveniencia instalé en una unidad diferente a la convencional C:.

2.6.5.1 Con privilegios de administrador, corrí el ejecutable del JDK que elegí (click derecho: ejecutar como administrador). En lo posterior diré: ejecuté como administrador. Ver figura 2.1.

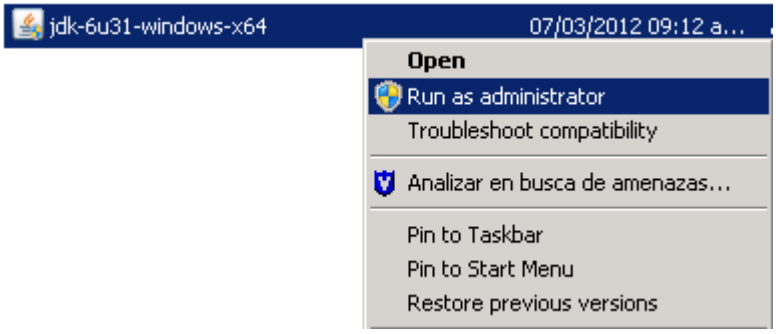

Figura 2.1 Ejecutar como administrador

NOTA: Si no se realizan todas las acciones como administrador pueden ocurrir dos escenarios, el primero que la instalación no pueda llevarse a cabo y la segunda es que se realice de manera inadecuada o incompleta. Por eso se hace precisa la ejecución como administrador.

2.6.5.2 (Creé la carpeta D:\Sun) Cambié el path de instalación a D:\Sun\jdk1.6.0_31\ y NO instalé el JRE (véanse las figuras 2.2 y 2.3). No es necesario para el funcionamiento de nuestro servidor, sólo desperdiciamos recursos.

Click Next.

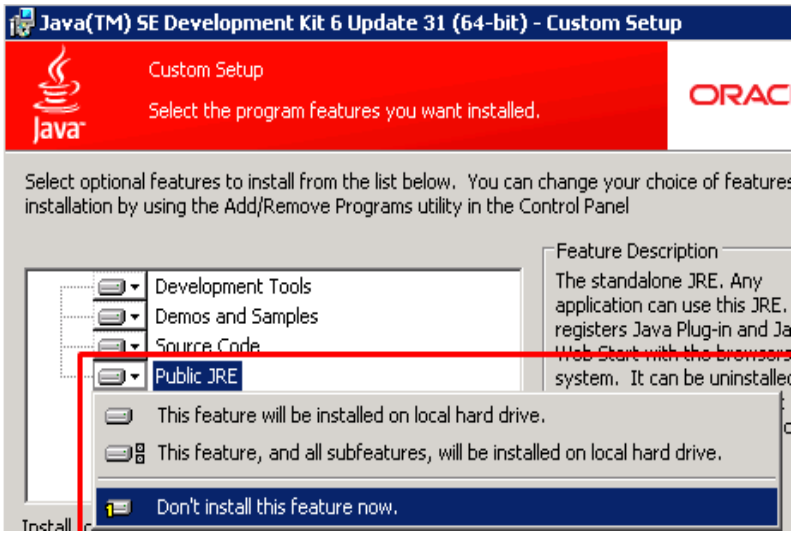

Figura 2.2 Deshabilitar la opción de instalar el complemento Public JRE.

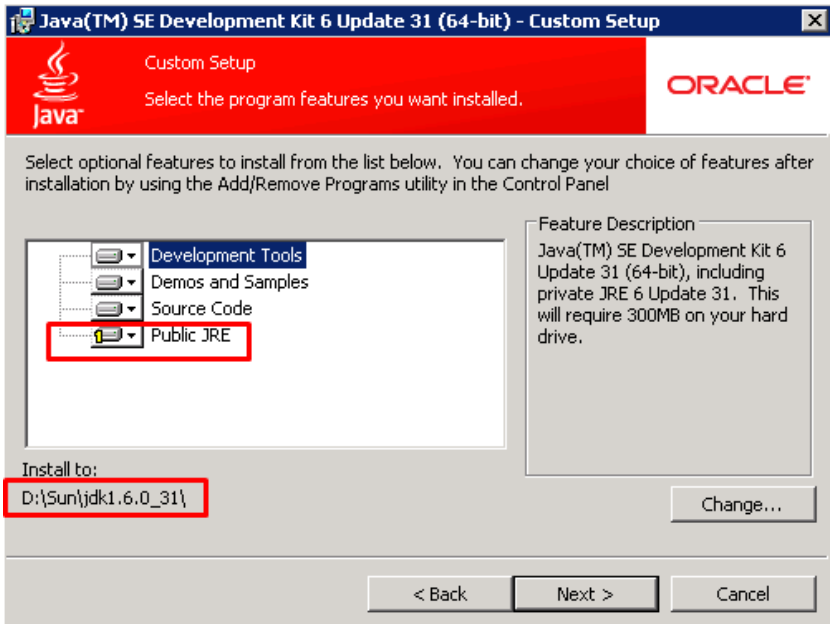

Figura 2.3 Deshabilitar la instalación del JRE y fijar la ruta de instalación requerida.

Una vez finalizada la instalación, se han creado todos los archivos en la carpeta especificada. Como estoy generando un servidor en cluster, se requiere la instalación del JDK en ambos servidores.

2.6.6 Instalación de GlassFish V 3.1.2

Este software lo instalé también en ambos servidores; el *DAS* y el cliente. Para llevar el órden de instalación del JDK, también instalé en la unidad D: en la carpeta Sun, donde creé su propia carpeta llamada GlassFishV312; es importante crear las carpetas idénticas en ambos servidores (cliente y *DAS*).

2.6.6.1 Creé carpeta para la instalación D:\Sun**GlassFishV312**

2.6.6.2 Corrí el ejecutable de GlassFish desde un CMD como administrador.

2.6.6.3 Ejecuté el comando: NOMBRE.EXE **–j** JDK_PATH como se observa en la figura 2.4.

En este caso usé el instalador GlassFish-3.1.2-windows.exe y recordando, el JDK, lo instalé en D:\Sun\jdk1.6.0_31

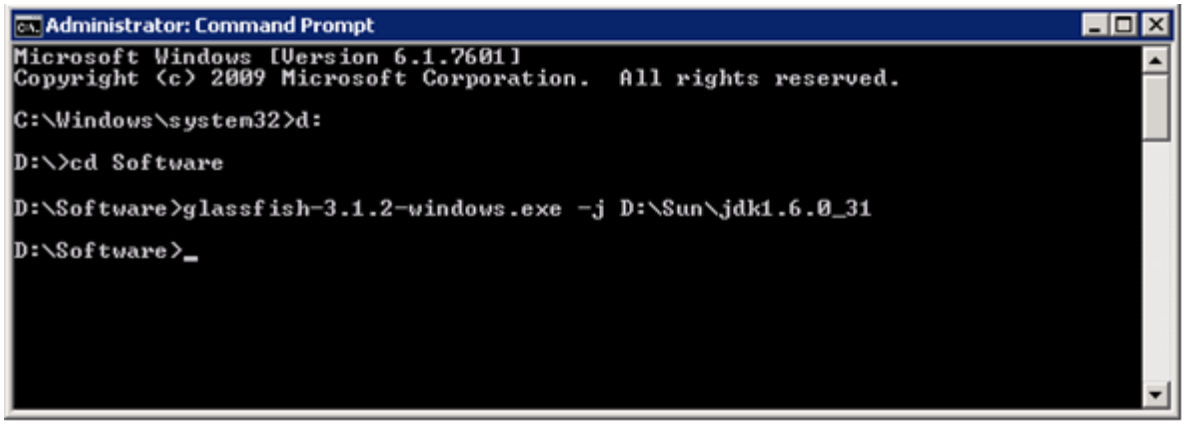

Figura 2.4 se ejecuta el instalador de GlassFish por medio de comandos.

2.6.6.4 A continuación apareció la pantalla de inicio de la instalación y elegí la instalación típica. figura 2.5.

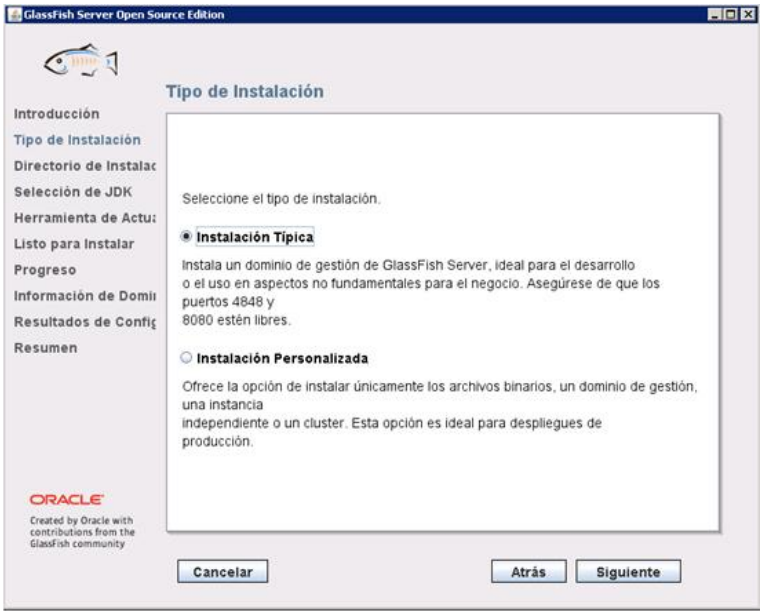

Figura 2.5 Instalación típica.

2.6.6.5 Cambié el path de instalación al creado en el punto 2.6.6.1. Ver fiigura 2.6.

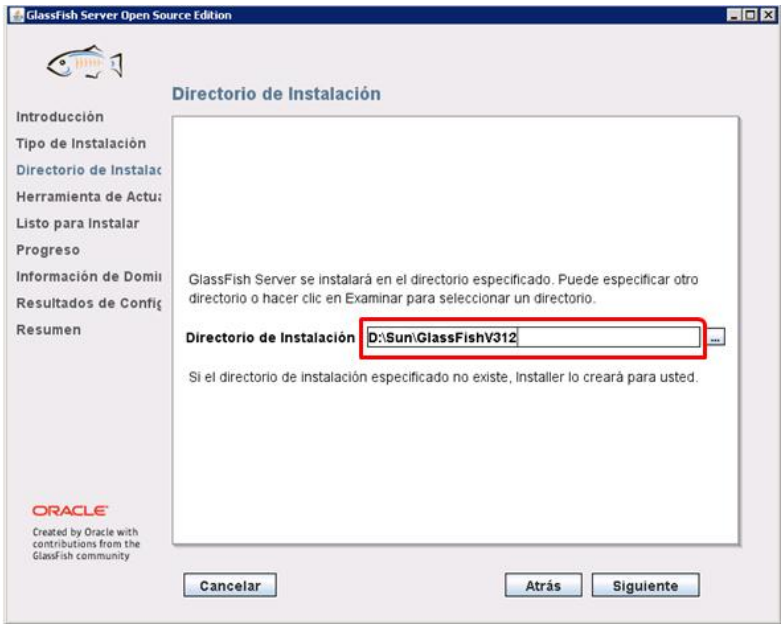

Figura 2.6 Se debe fijar el path de instalación correcto.

2.6.6.6 Deshabilité la opción de Actualización automática, figura 2.7. No es conveniente que se actualice automáticamente pues pudiera ser que algunas de las aplicaciones dejaran de operar correctamente; es importante tener un control de los cambios en el software, en especial en modo productivo.

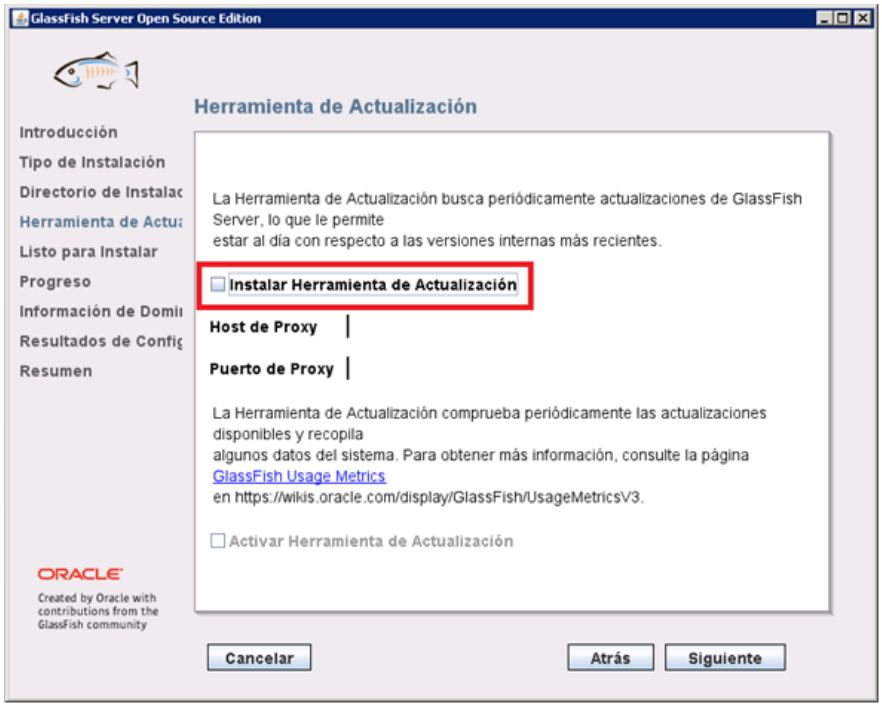

Figura 2.7 No se instaló la herramienta de actualización automática.

Elegí Siguiente\Instalar

2.6.6.7 La pantalla de información de dominio, fue la misma en ambos servidores a diferencia del nombre de dominio, elegí el nombre de cada servidor acompañado de la palabra domain.

Es importante señalar que hasta este punto la instalación es igual en ambos servidores, pero la opción de crear Servicio de SO para el Dominio sólo la habilité en el *DAS*, que es desde donde administraremos el *cluster*. Ver figura 2.8.

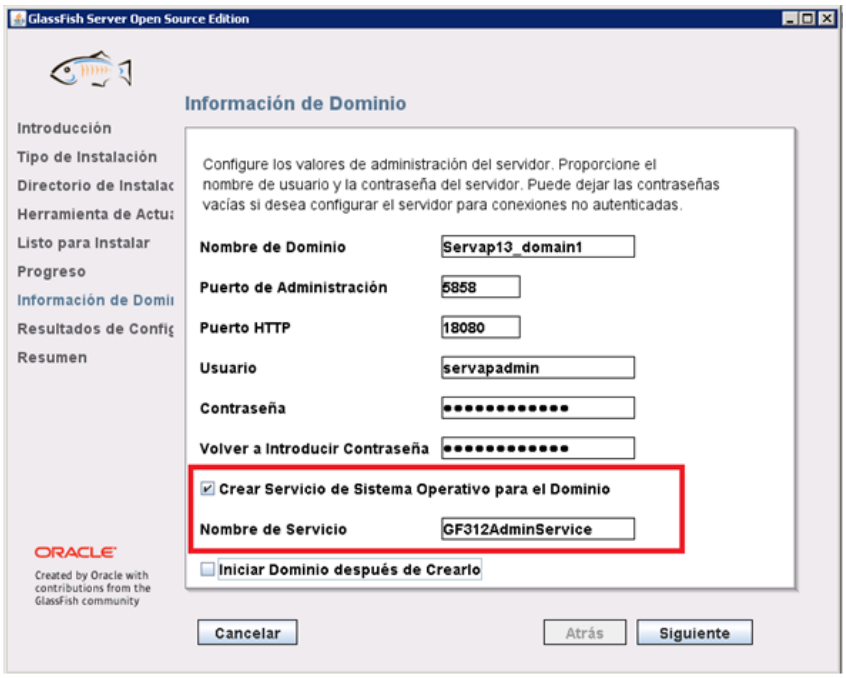

Figura 2.8 Esta opción sólo se habilita en la instalación en el DAS.

2.6.6.8 Una vez finalizada la instalación nos muestra un resumen y el estado. Ver figura 2.9.

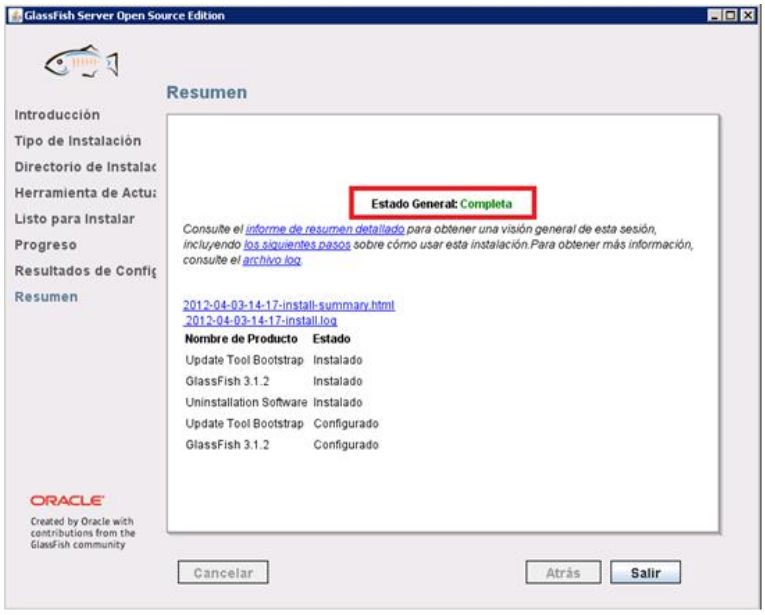

Figura 2.9 Resumen general de instalación y estado final de la instalación.

2.6.7 Configuración del nodo administrador

2.6.7.1 Esta parte sólo es válida para el *DAS*, es decir que en los nodos no lo configuré.

Después de conectarme remotamente al servidor virtual administrador, busqué en los servicios de Windows el de la consola de administración para iniciarlo, figura 2.10. El nombre del servicio tiene un formato parecido al siguiente:

ServapX_domainGlassFish Server

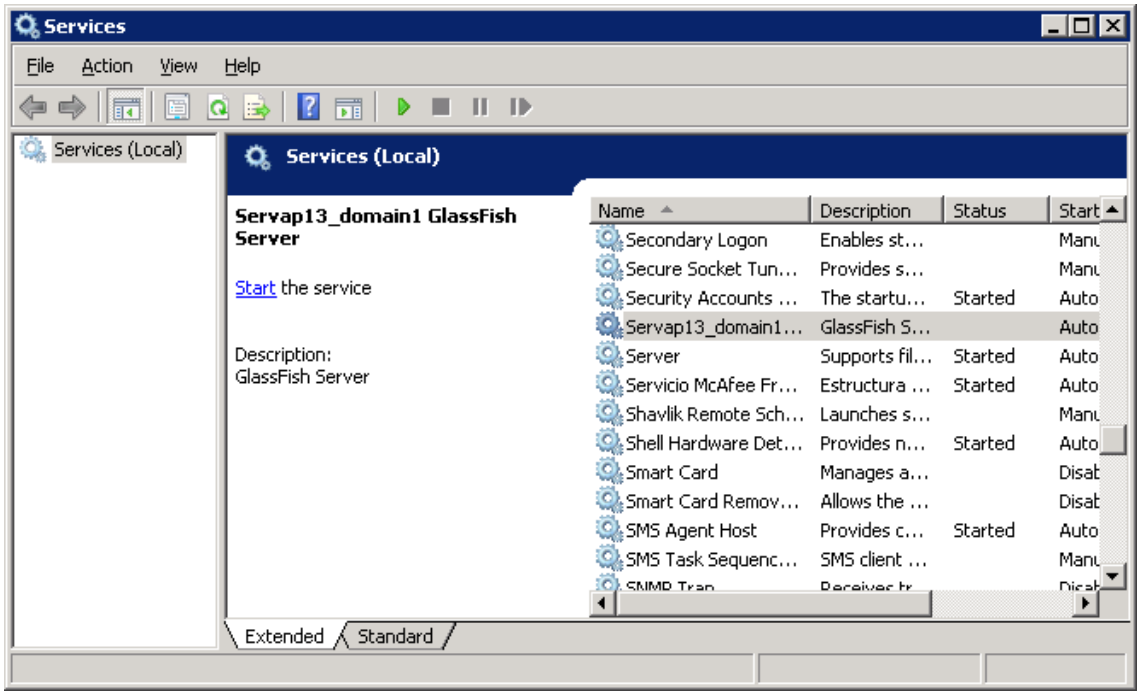

Figura 2.10 Servicios de Windows.

2.6.7.2 Con el servicio del dominio arriba, ejecuté como administrador un CMD. Cambié el directorio a la ruta en que se encuentra un archivo .bat al que llamé setenv312

D:\Sun>setenv312.bat

Nota: el setenv312, es un archivo que creé sólo para fijar y asegurarme de que estaba usando la máquina virtual de JAVA correcta. Si no existiera el archivo, tendría que asegurarme de ejecutar los comandos desde el path correcto en la ventana de comandos.

Como se muestra en la siguiente imagen (figura 2.11) ya me situé en el directorio en que instalé el servidor de aplicaciones.

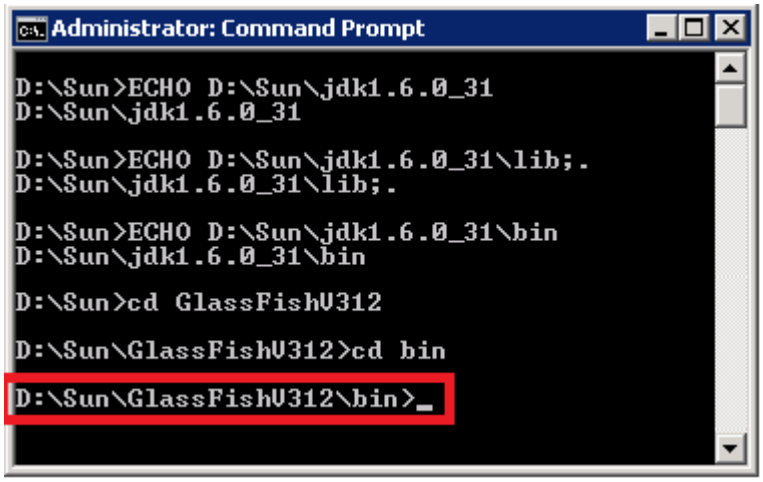

Figura 2.11 Al correr el archivo .bat creado, el sistema fija la ruta deseada.

2.6.7.3 Corrí el comando **asadmin --host xx.xx. xx.xx –port YYYY enable-secure-admin ,** de esta forma aseguré que tendremos acceso por https. Ver figura 2.12. Este paso se puede omitir en general pero se usa por conveniencia.

Figura 2.12 Se habilita el acceso a consola, para administración segura, desde https.

2.6.7.4 REINICIÉ el servicio para que tomara los cambios. Una vez realizado, la consola de administración se accesó por https con una Url semejante a la siguiente:

https:\\servidor1:4848 que es el nombre del servidor o la IP y el puerto de administración que definimos durante la instalación de GlassFish. Ver figura 2.13.

2.6.7.5 Me conecté a la consola de GlassFish y firmé con el usuario y contraseña usados durante la configuración. En la siguiente imagen se puede ver el aspecto de la consola de administración.

Figura 2.13 Consola de administración de GlassFish V3.1.2 accesada por https.

NOTA: Siempre después de cada cambio y antes de cambiar de pestaña de configuración es indispensable guardar (SAVE) los cambios; de otro modo las configuraciones no tendrán efecto.

En Configuration/server-config

2.6.7.5.1 En JVM Options(figura 2.14): cambié

–client por –server

6666 por 16666

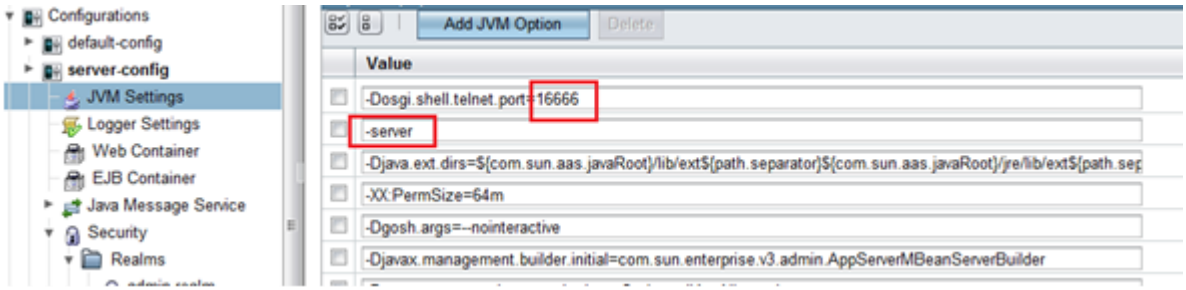

Figura 2.14 Se cambiaron los valores resaltados.

2.6.7.5.2 En Logger Settings, tab General cambié el path del log file a uno conocido y fácil de identificar para futuras referencias:

Por ejemplo en F:/GlassFishV312/logs/AdminServer.log (figura 2.15)

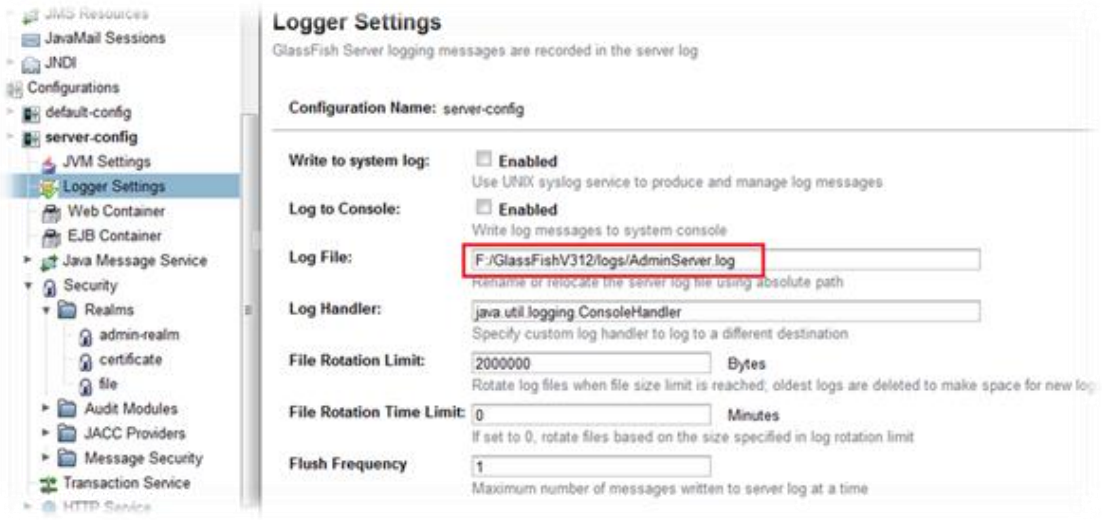

Figura 2.15 El path del log file se puede cambiar por la ubicación deseada.

2.6.7.5.3 En Thread Pools/http-thread-pool y cambié a 1 el tamaño mínimo de thread (Ver figura 2.16).

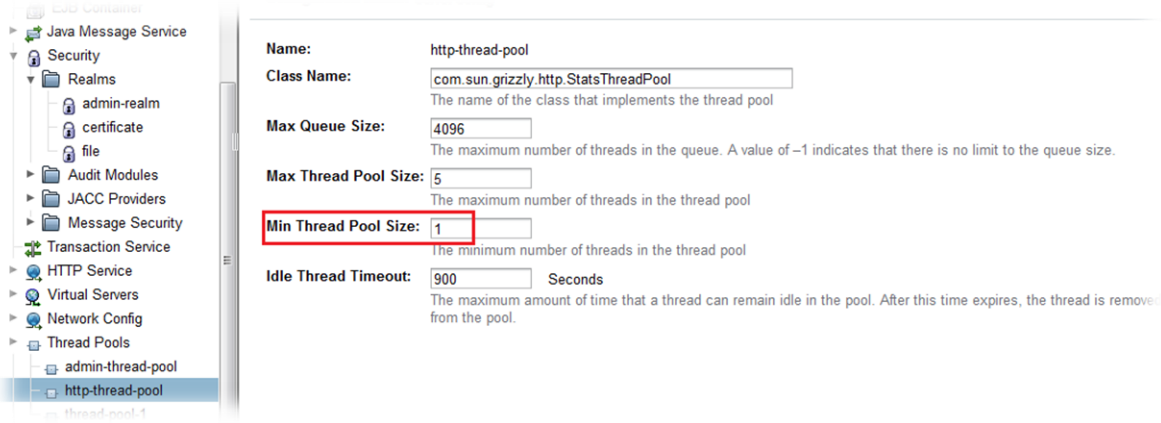

Figura 2.16 Se fija el número mínimo de hilos a 1.

2.6.7.6 En el Domain: deshabilité las opciones de reload y auto deploy. Ver figura 2.17.

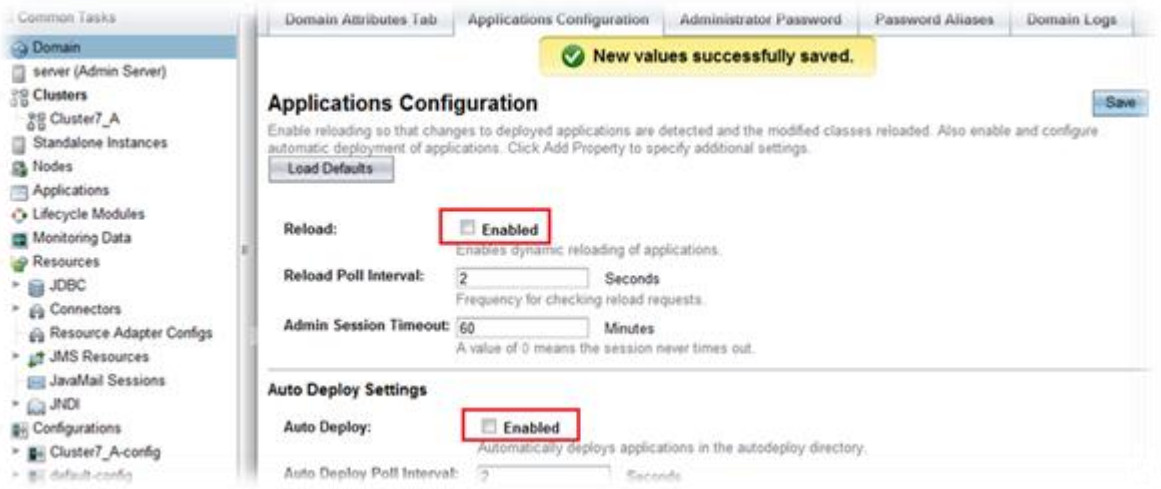

Figura 2.17 Se deshabilitan las opciones de reload y redeploy.

2.6.7.7 En HTTP Service: cambié los puertos (agregar p. ej. **1**8080 en lugar de 8080). Ver figura 2.18.

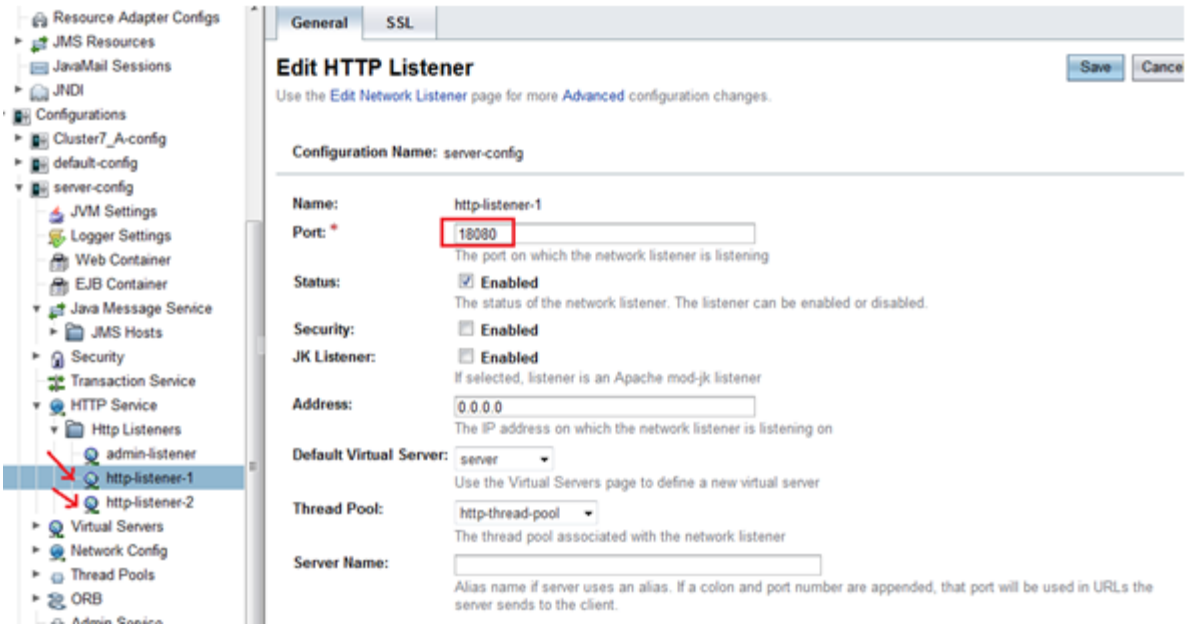

Figura 2.18 Es importante cambiar este puerto para que no choque con otros puertos que configuré.

2.6.7.8 En ORB/IIOP Listeners: cambié el puerto igual que en el caso anterior (para los tres listeners). Ver figura 2.19.

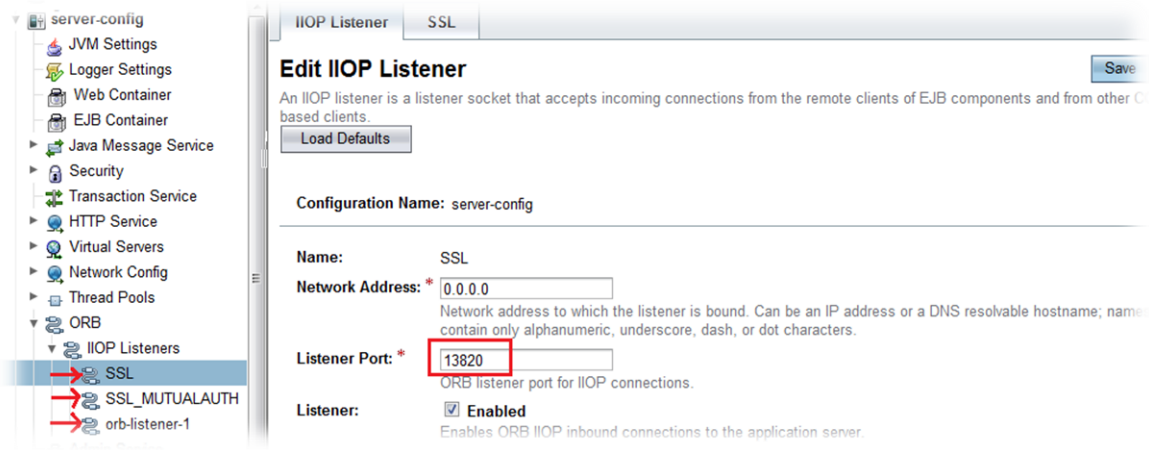

Figura 2.19 Se cambia el puerto del listener.

2.6.7.9 En Java Message Service/JMS Host/default_JMS_host: cambié el puerto. Ver figura 2.20.

Figura 2.20 Cambiar el Puerto de JMS del servidor. P ej. Precediendo "1" al valor default.

Una vez finalizadas las configuraciones, fue necesario reiniciar para que el servidor tomara los cambios.

2.6.8 Instalación de SSH (Cygwin)

Como se puede observar, el proveedor de SSH que elegí para su implementación en el presente informe fue Cygwin, que es un software de, relativamente, sencilla instalación y para el que no se requiere pagar licencia alguna pues se distribuye habitualmente bajo los términos de la GPL (General Public License) además de que, en caso necesario, puede utilizarse con cualquier tipo de software cuya licencia esté de acuerdo con la definición de software libre [17].

Esta instalación sólo la hice en el servidor cliente.

2.6.8.1 Inicié una sesión en el servidor cliente con una cuenta de administrador.

2.6.8.2 Creé el folder **cygwin** en la unidad C:

2.6.8.3 De la página de Cygwin [18] bajé el ejecutable **setup.exe** de SSH y lo copié en el servidor.

2.6.8.4 Ejecuté como administrador el **setup.exe**. Ésta es la imagen del instalador, figura 2.21.

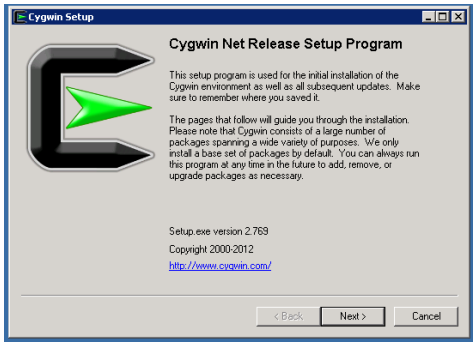

Figura 2.21 Instalador Cygwin

2.6.8.5 Se puede elegir las opciones de instalación y/o descarga de la imagen, yo lo instalé desde internet. Ver figura 2.22.

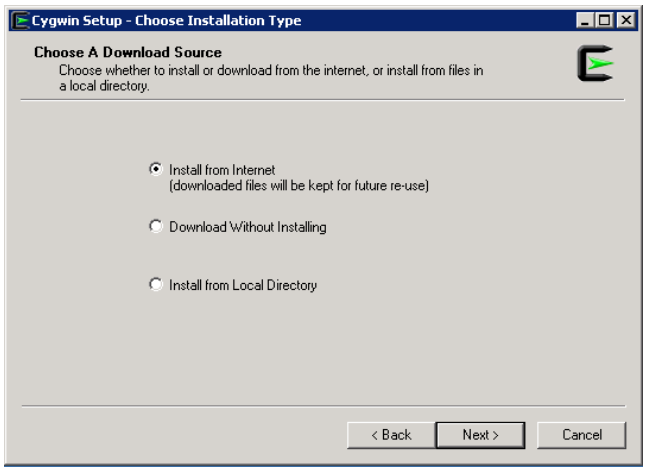

Figura 2.22 Elegir el tipo de instalación.

2.6.8.6 Cambié la ruta de instalación a C:\cygwin y elegí la opción de instalar para todos los usuarios. Ver figura 2.23. A diferencia de la instalación de GlassFish que se hizo en una unidad diferente a C; la instalación de Cygwin sí la hice en dicha unidad, porque como se verá más adelante en la configuración, es necesario empatar el home de Windows y Cygwin por lo que resulta favorable, práctico e igualmente funcional. Ver figura 2.23.

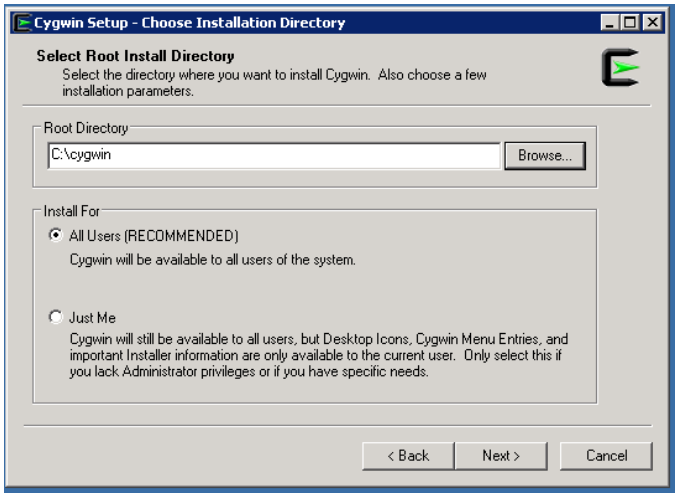

Figura 2.23 Fijar el directorio en que se hará la instalación.

2.6.8.7 Fijé una ruta en la que se descargan los paquetes necesarios para la instalación. Puede ser la misma que el path de instalación, Cygwin descarga los paquetes en una carpeta ahí dentro. Ver figura 2.24.

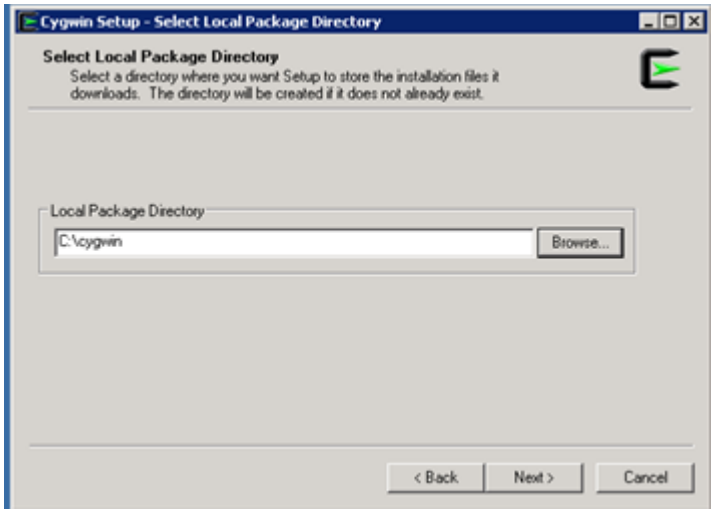

Figura 2.24 Fijé una ruta para descarga de los paquetes extras de Cygwin.

2.6.8.8 Seleccioné conexión directa a internet. Ver figura 2.25

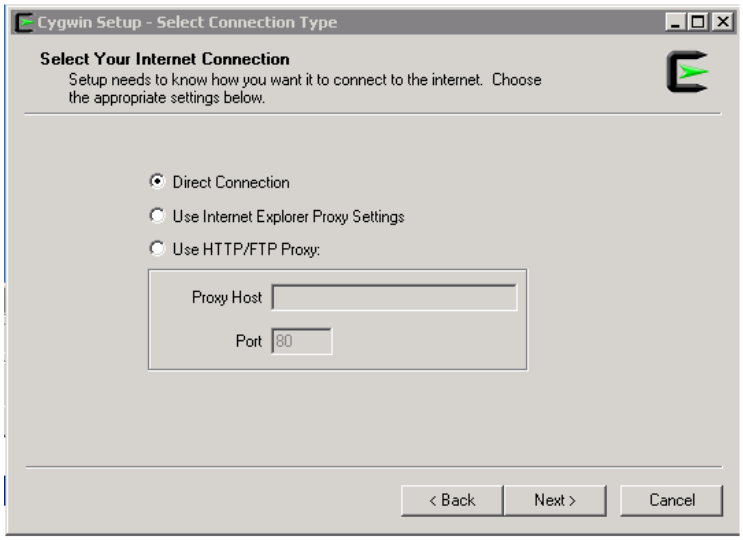

Figura 2.25 Usé la conexión directa a internet.

2.6.8.9 Se presenta una lista de sitios de donde se pueden descargar los paquetes para la instalación. Seleccioné el que se muestra en la imagen. Ver figura 2.26. Aunque cabe señalar que todos los sitios son válidos, sin embargo a veces están lentos o fuera de servicio.

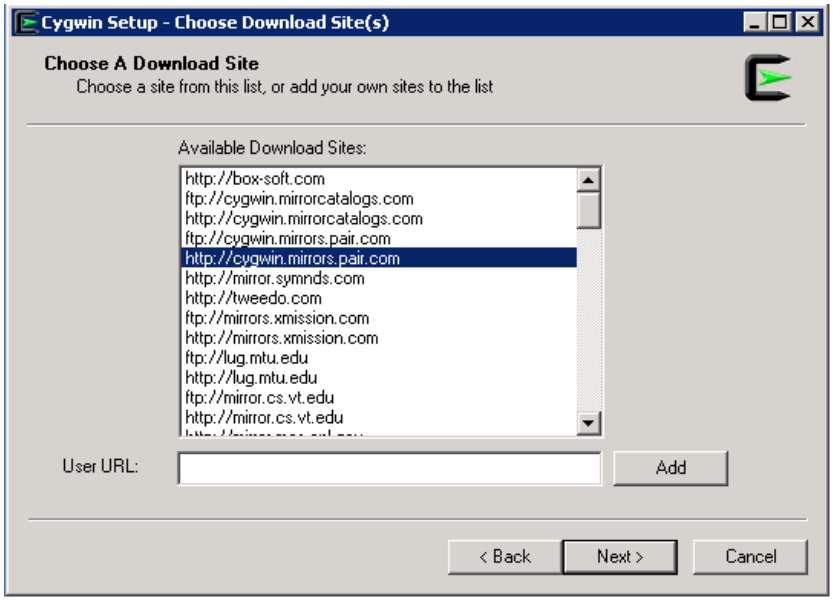

Figura 2.26 Seleccioné un sitio desde el que se descargaron los paquetes.

2.6.8.10 Aparece un mensaje de Cygwin que nos advierte de nuevas versiones (si aplica) figura 2.27, en caso de tener instalaciones previas habrá que actualizarlas o volver a instalar de cero una nueva versión. En nuestro caso, parto de un servidor nuevo para la implementación, por tanto omití el mensaje:

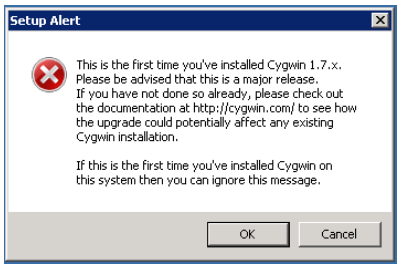

Figura 2.27 Advertencia sobre instalaciones previas de Cygwin y las versiones actualizadas disponibles.

2.6.8.11 Busqué bajo la categoría NET (figura 2.28), **openssh** (figura 2.29).

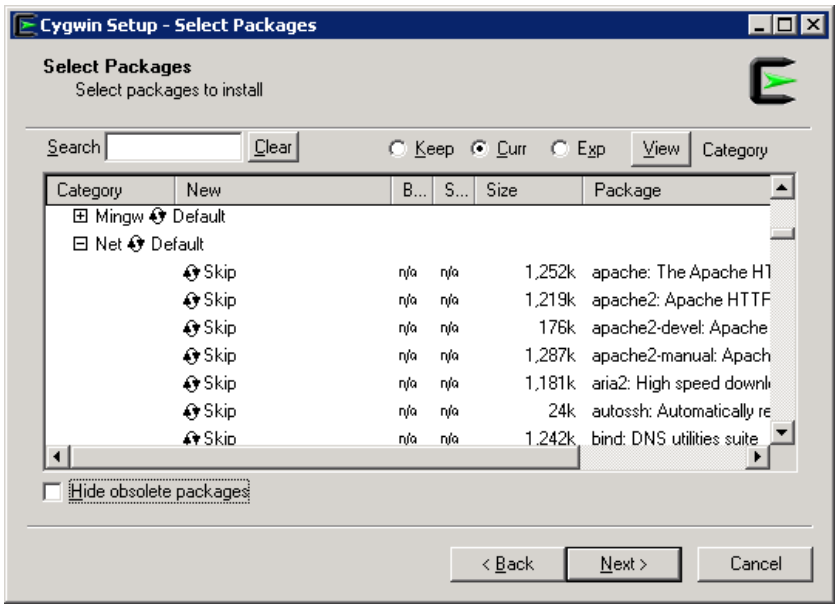

Figura 2.28 Ubicar la categoría NET

2.6.8.12 Una vez localizado. Ver figura 2.29.

Figura 2.29 buscar subcategoría openssh

Hice click en Skip para que se descargara. Ver figura 2.30.

Figura 2.30 Al quitar la opción Skip, se marca la subcategoría que se descargará.

Aparece una lista de dependencias de los paquetes que hay que dejar intacta. Elegí Next/Finish.

2.6.8.13 En el paso anterior dejé activada la opción de crear un acceso directo a la terminal *Cygwin* desde el escritorio.

2.6.9 Setear los Paths de Cygwin y Windows

Para que el SSH funcione adecuadamente, en el servidor cliente donde lo instalé, incluí en la variable de entorno de Windows **PATH,** las rutas del directorio bin del JDK y el cygwin de la siguiente manera:

2.6.9.1 Dentro del servidor cliente con cuenta administradora, fui al menú Inicio/Equipo (con el botón derecho del mouse) Propiedades/Configuración Avanzada del Sistema/Variables de entorno.

2.6.9.2 Busqué la variable PATH y agregué la siguiente línea:

;C:\cygwin\bin;D:\Sun\jdk1.6.0_31\bin que incluye los directorios que mencioné separados por un punto y coma y sin espacios.

2.6.9.3 Verifiqué la existencia de la variable llamada **Cygwin** con valor **tty**

2.6.9.4 Fijé también el home path de SSH igual al de Windows (Lo obtuve mediante el comando **set home** en un CMD **c/Users** en mi caso).

2.6.9.5 Busqué el archivo passwd el C:\cygwin\etc\passwd, lo abrí modificar el home path del usuario que creamos como administrador, por ejemplo **servadmin** cambiando **/home/** por **/c/Users**

Quedó como sigue:

servadmin:unused:1003:513:Administrador de Servaps,U-SERVAP14\servadmin,S-1-5-21-1146313620-723355851-1958474259-1003**:/c/Users/**servadmin:/bin/bash

2.6.10 Configuración de Cygwin

2.6.10.1 Busqué la terminal de Cygwin, que se creó en el paso final del punto 2.6.8.13. Click derecho en el ícono de la terminal cygwin y ejecuté como administrador.

NOTA: Una vez abierta la terminal, no la cerré hasta terminar la configuración.

2.6.10.2 Para asegurarme de que estaba fijado un password para el usuario administrador, corrí el comando passwd nombreDeUsuario:

$ **passwd servadmin**

Me pidió ingresar y confirmar el password (es el mismo que el password que usé para conectarme al servidor como administrador en Windows).

2.6.10.3 Una vez fijado el password configuré el SSH con el siguiente comando en la terminal (figura 2.31):

$ **ssh-host-config –y**

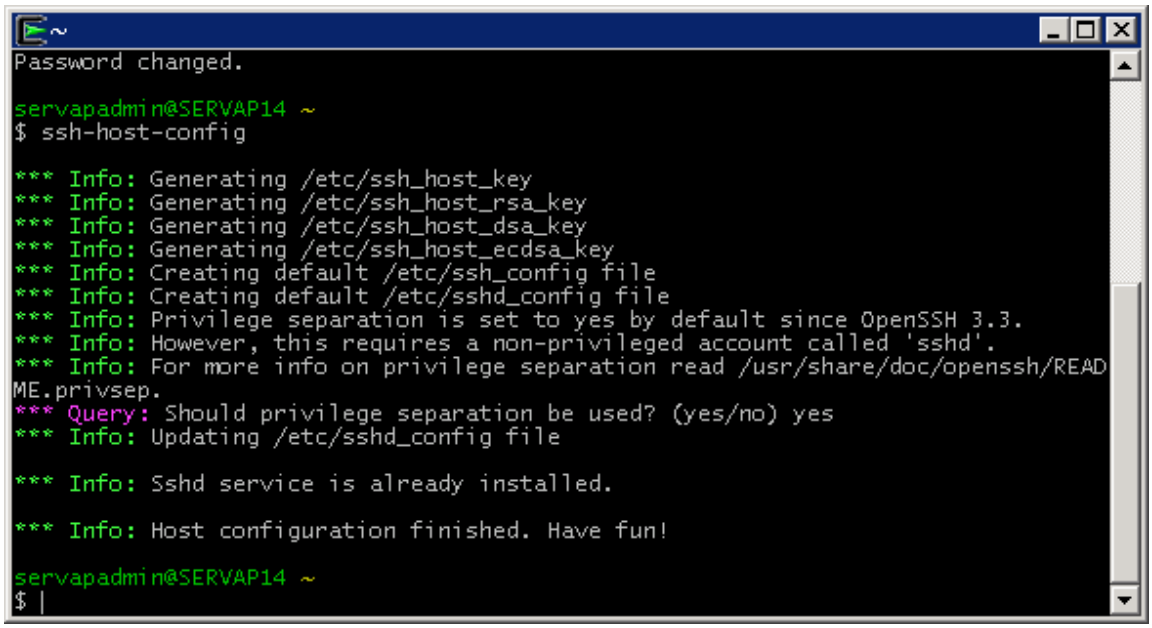

Figura 2.31 Terminal Cygwing donde se configura el demonio.

2.6.10.4 Entré al archivo sshd_config que está en C:/cygwin/etc/ y para verificar y modificar que la configuración de las siguientes propiedades fuese la siguiente:

StrictModes **yes**

PubkeyAuthentication **yes**

AuthorizedKeysFile .ssh/authorized_keys

2.6.10.5 En la terminal, corrí el comando:

$**net start sshd**

2.6.10.6 Confirmé que el demonio de SSH está activo con el comando:

$ **cygrunsrv --query sshd**

(El demonio de SSH es el programa que escucha conexiones de red desde clientes ssh, maneja autentificación y ejecuta las peticiones de comando [19]).

Apareció la siguiente información que confirmó la correcta configuración.

Service : sshd

Display name : CYGWIN sshd

Current State : Running

Controls Accepted : Stop

Command : /usr/sbin/sshd –D

2.6.11 Pruebas de conectividad con el SSH

Para realizar estas pruebas me conecté a la consola de administración.

2.6.11.1 Creé el NodeAgent del servidor cliente.

2.6.11.1.1 En el menú NODES/new. Ver figura 2.32.

Llené la siguiente información:

Name : servapY_NA

Type:SSH

Node host: servapY (el servidor cliente)

SSH User Name: Nombre de usuario administrador

SSH User Autentication: Password

SSH Password: el que usamos para Cygwin

Verifiqué la conectividad con el ping.

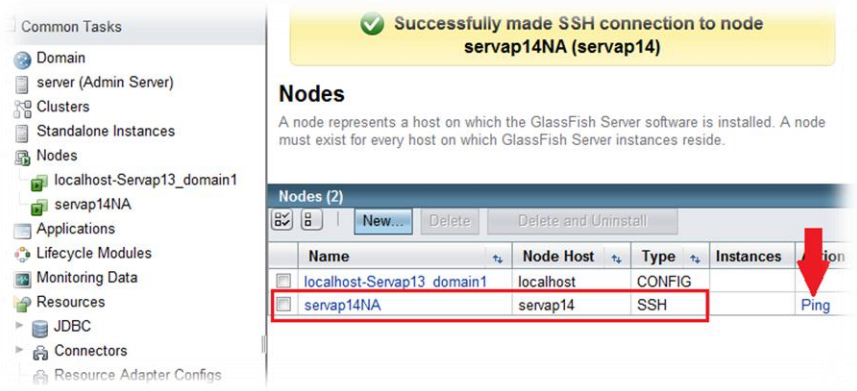

Figura 2.32 Configuración del nodo y verificación de conectividad con el demonio de Cygwin.

2.6.12 Creación de los clusters.

Con ambos nodos, el local host y el Node Agent que creé para las pruebas del punto anterior; formé el cluster.

2.6.12.1 En CLUSTERS/New

Le di un nombre y elegí las opciones según la imagen de la figura 2.33:

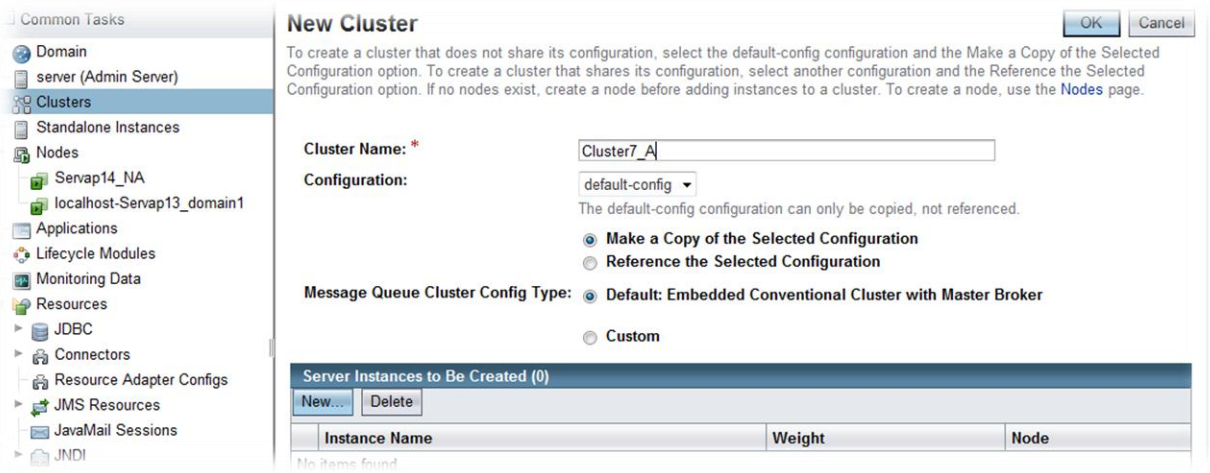

Figura 2.33 Opciones de configuración del cluster.

2.6.12.2 Creé las instancias de cada cluster. Ver figura 2.34:

En Clusters/(el cluster creado) en la pestaña Instances/New Instance

Nombré la instancia y elegí el Node Agent respectivo (el del *DAS* o el del cliente según era el caso), verificando que el nombre de la instancia correspondiera con el nodo seleccionado. Ver figura 2.35.

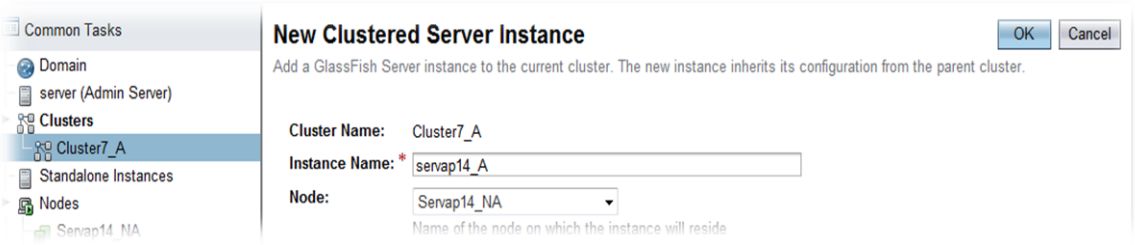

Figura 2.35 Configuración de las instancias.

2.6.12.3 Una vez creadas Iniciar las instancias:

Inicié las instancias. Ambas quedaron en estatus Running. Ver figura 2.35.

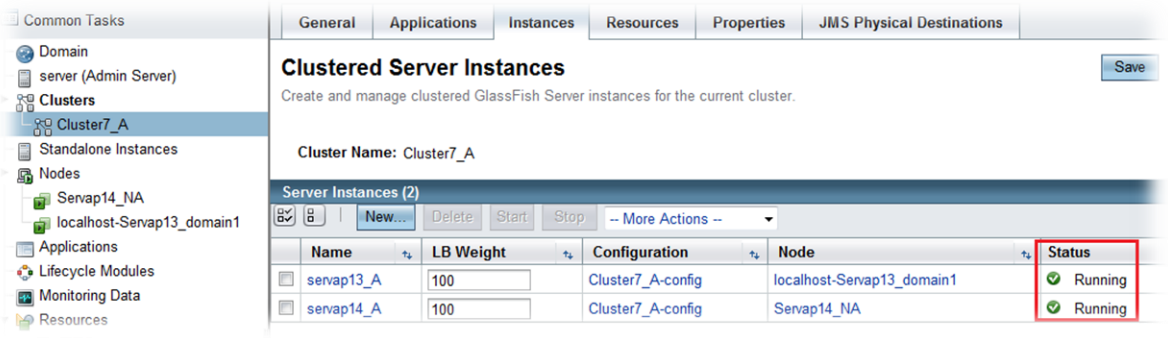

Figura 2.35 El estado de las instancias debe ser Running lo que indica que se crearon satisfactoriamente.

2.6.13 Generación de servicios Windows de los clusters

2.6.13.1 Me conecté a cada servidor con una cuenta administradora. Ver figura 2.36.

2.6.13.1.1 Corrí un CMD como administrador. Cambié a la ruta donde está el setenv.bat y lo ejecuté:

D:\Sun \>setenv312.bat

2.6.13.1.2 Cambié a la ruta D:\Sun\GlassFishV312\bin

2.6.13.1.3 ejecuté la siguiente línea:

>**asadmin --host**(ipServer) **--port**(si es diferente a 4848) **--user** NombreUsuarioAdministrador **create-service** NombreNodo

P. ej:

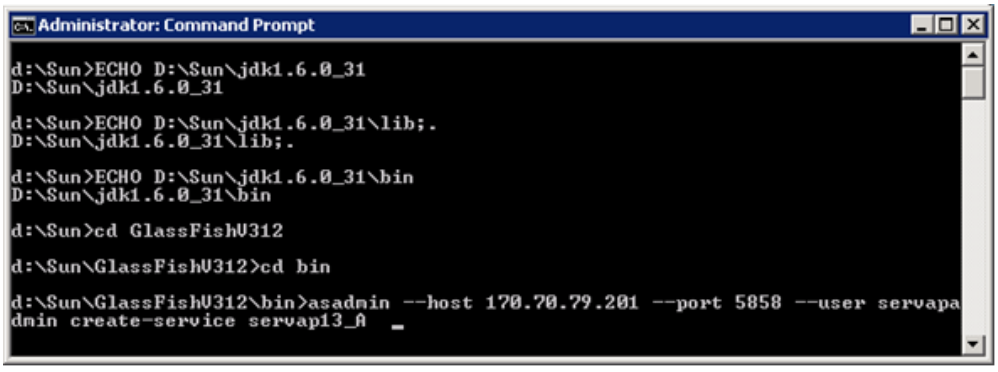

Figura 2.36 Creación de los servicios de Windows, correspondientes a los nodos creados, mediante línea de comandos.

NOTA: Es necesario tener los nodos arriba, de lo contrario no se crean los servicios.

2.6.13.2 Una vez creados, comprobé que los servicios controlan a los nodos. Detuve los servicios desde Windows y verifiqué que en la consola cambiara el estado a Stoped. Luego los Inicié de nuevo. Ver figura 2.37.

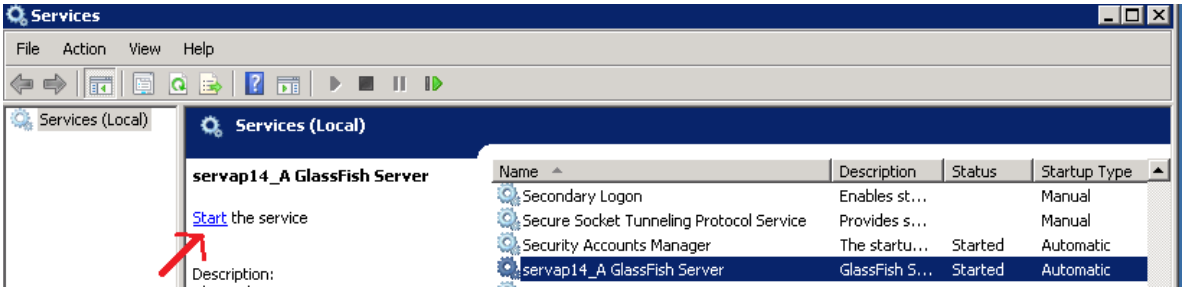

Figura 2.37 Se localizan los servicios y se detienen/inician desde Windows.

2.6.14 Configuración de los clusters.

2.6.14.1 En En Configurations/ClusterX-config:

2.6.14.1.1 JVM Settings/JVM Options

Agregué las propiedades (figura 3.38):

–Xms y –Xrs

Cambié los valores:

–Xmx y -XX:MaxPermSize

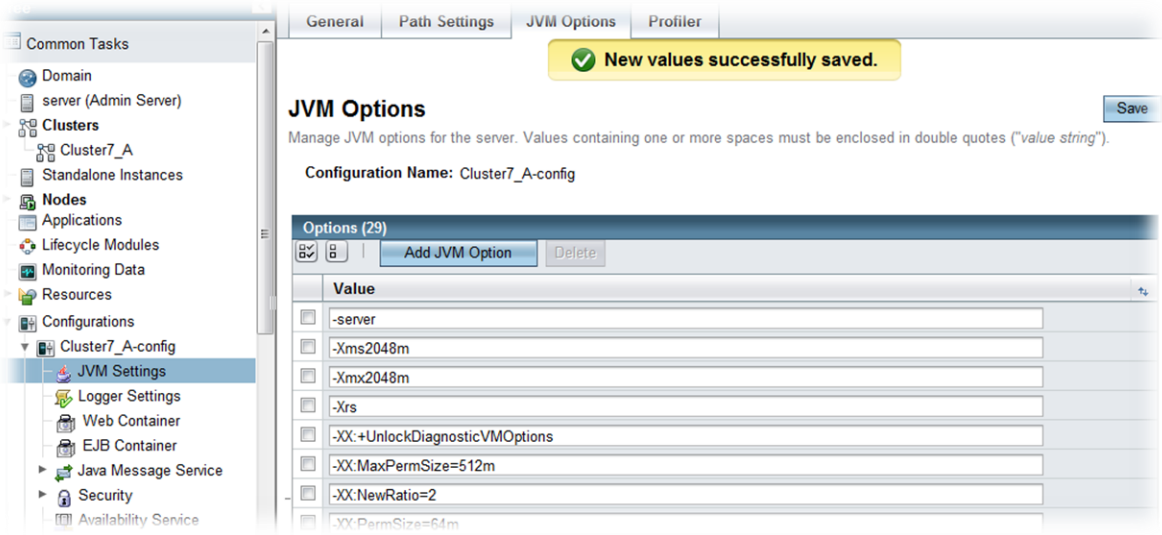

Figura 2.38 Se agregan y modifican algunas propiedades de las opciones de la Máquina Virtual

2.6.14.1.2 Logger Settings, pestaña General

Cambié el path de logueo (figura 2.39)

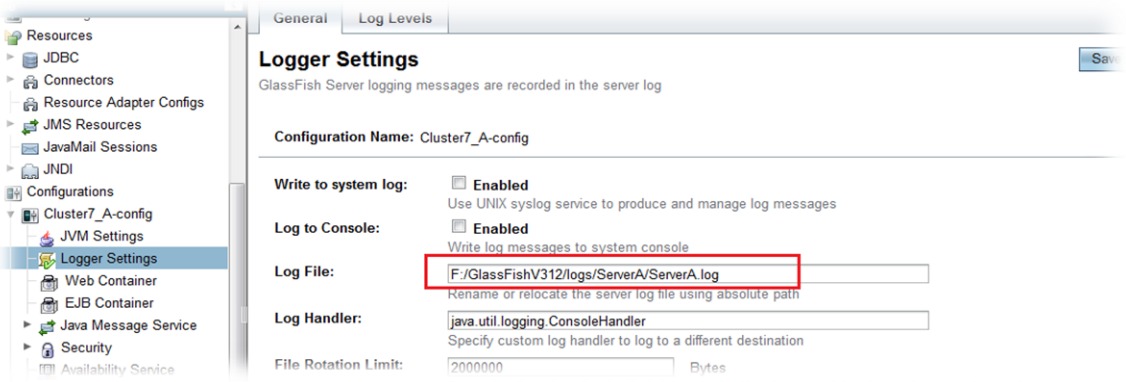

Figura 2.39 Cambiar path de logueo

2.6.14.1.3 JMS. Borré el JMS Host default y creé uno para cada nodo. Ver figura 2.40.

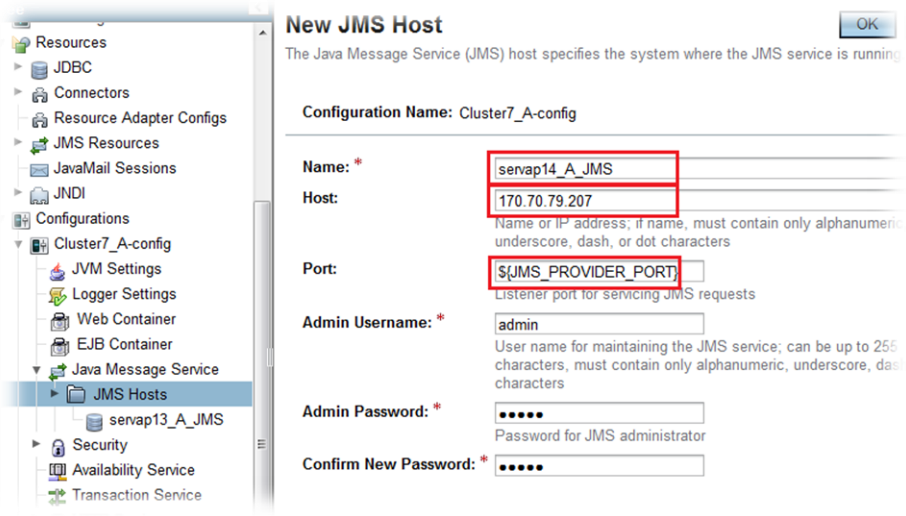

Figura 2.40 Se crean JMS Host nuevos para cada nodo.

2.6.14.1.4 Creé el JMS sólo para el administrador. Verifiqué la conectividad con un ping. Ver figura 2.41.

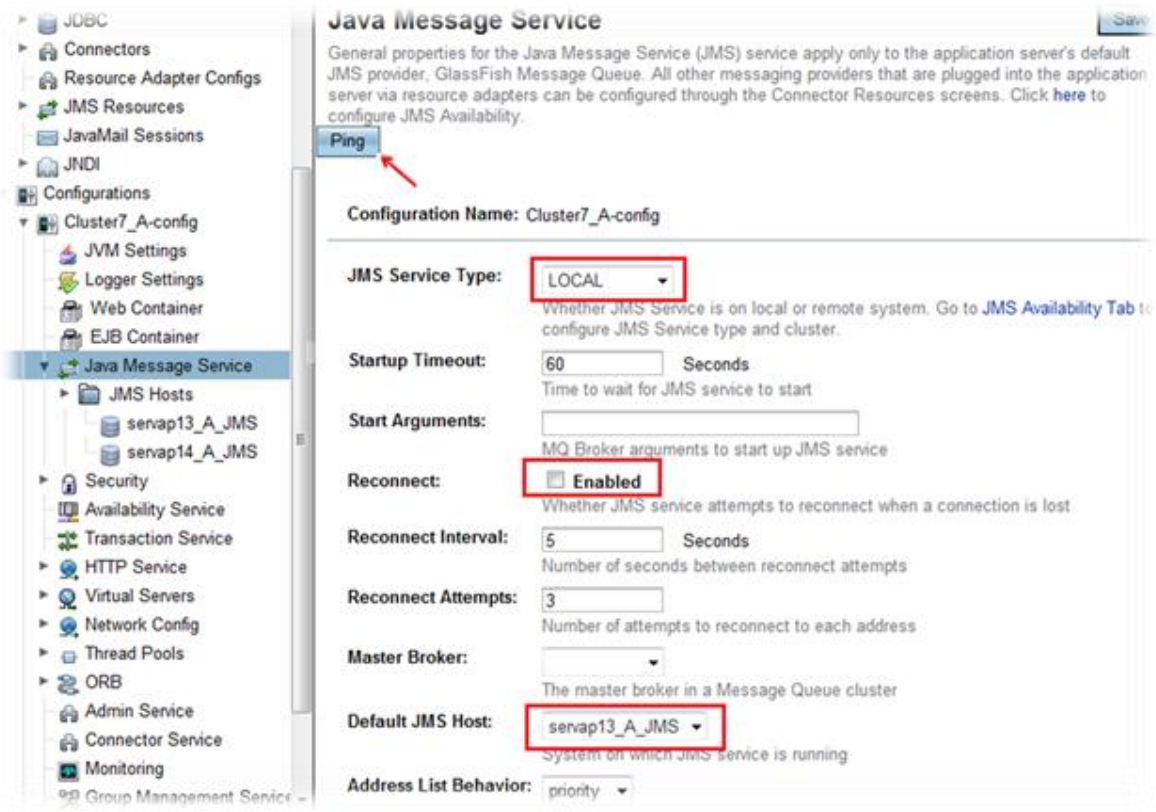

Figura 2.41 Con la opción “Ping” se puede verificar la correcta creación de cada JMS.

2.6.14.1.5 Verifiqué que en HTTP Services y ORB/IIOP-Listeners, los listeners tuvieran en el valor puerto una variable del tipo ${Nombre del puerto}

2.6.14.1.6 Configuré los puertos. En System properties. Vea la figura 2.42.

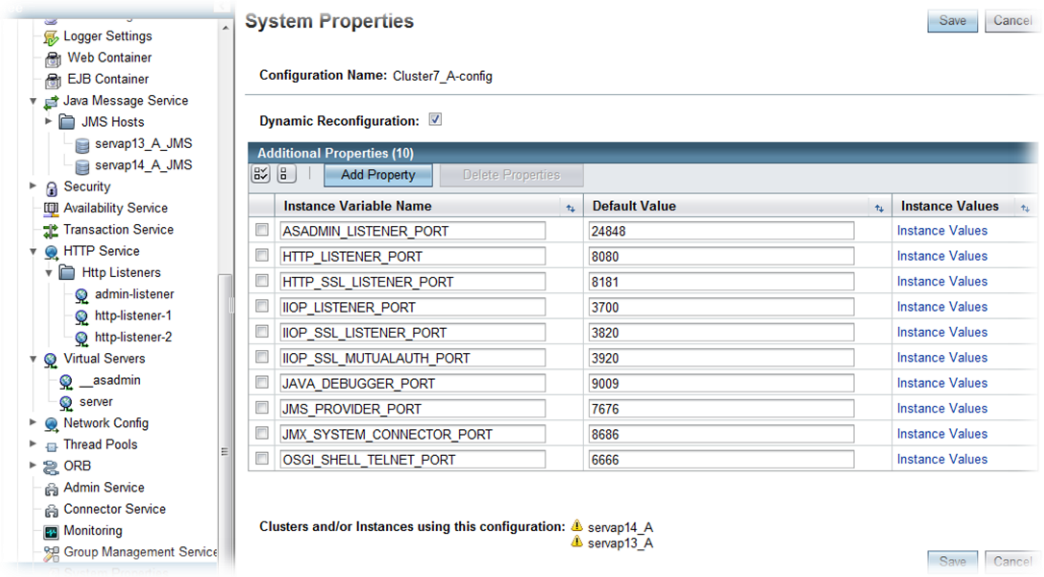

Figura 2.42 Configuración de puertos en cada nodo.

2.6.14.1.7 Levanté ambos nodos del cluster

NOTA: De no levantar los nodos habría que revisar las bitácoras.

2.6.14.1.8 REPETÍ todos los pasos de las sección 2.6.14 por cada cluster.

CAPÍTULO 3 Resultados

3.1 Resultados y aportaciones

3.1.1 **Mejora en los tiempos de atención a los usuarios**. Esta versión automatiza el copiado y borrado de aplicaciones durante la implementación y baja de aplicaciones.

En versiones anteriores de GlassFish instalado en cluster (mono núcleo) había que hacer algunas operaciones manuales como replicar la información en el servidor cliente o secundario.

Ejemplo. Vea la figura 3.1. Una aplicación llamada ***Convenios*** la cual se encuentra desplegada en la versión 2.1.1 en servidor1 y servidor2, donde servidor1 es el principal.

Como se observa en la figura 3.1, la aplicación ya fue desplegada, sin embargo la implementación, a nivel de archivos, sólo actúa en servidor1, por lo que la copia en servidor2 debe hacerse manual, como se ve en la figura 3.2.

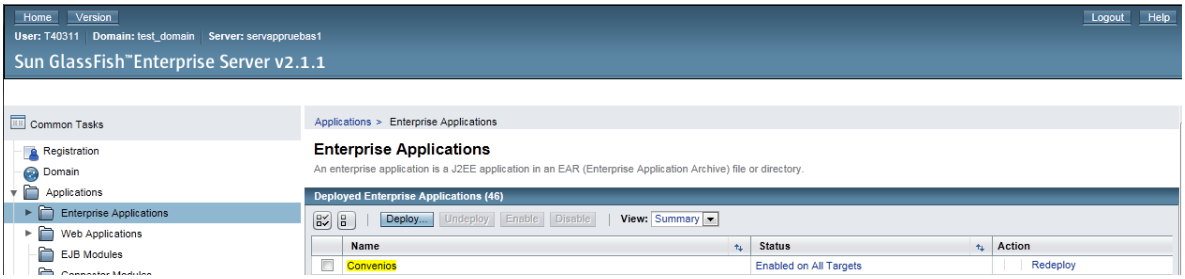

Figura 3.1. En la imagen se observa la aplicación, Convenios, desplegada en el servidor.

Gráficamente podemos observar que sólo se despliega en el servidor principal, servidor1, por lo que hay, literalmente, que copiar la carpeta desplegada al servidor secundario, servidor2.

Con la versión 3.1.2 esto ya no es necesario debido a que la réplica de información es automática.

Hay que recordar que en las versiones anteriores (2.1 y 2.1.1) del servidor, para su despliegue, las aplicaciones eran recibidas por los administradores en forma de paquetes ***.ear*** *o* ***.war****. Vea la figura 3.4. Se colocaban en carpetas específicas donde se guardaban los respaldos (por ejemplo C:/Aplicaciones de TI/…) y desde esa ubicación se desplegaban, así el servidor las tomaba como paquetes y las "explotaba" en una ruta diferente (por ejemplo C:/GlassFishVxx/…) de donde se servían las peticiones de cada aplicación en el servidor web. Es importante denotar que el repositorio de respaldos es independiente del funcionamiento de las aplicaciones, no así el repositorio donde se explotan las aplicaciones, cualquier cambio en los archivos en él contenidos, afectan directamente el funcionamiento en línea de las aplicaciones.*

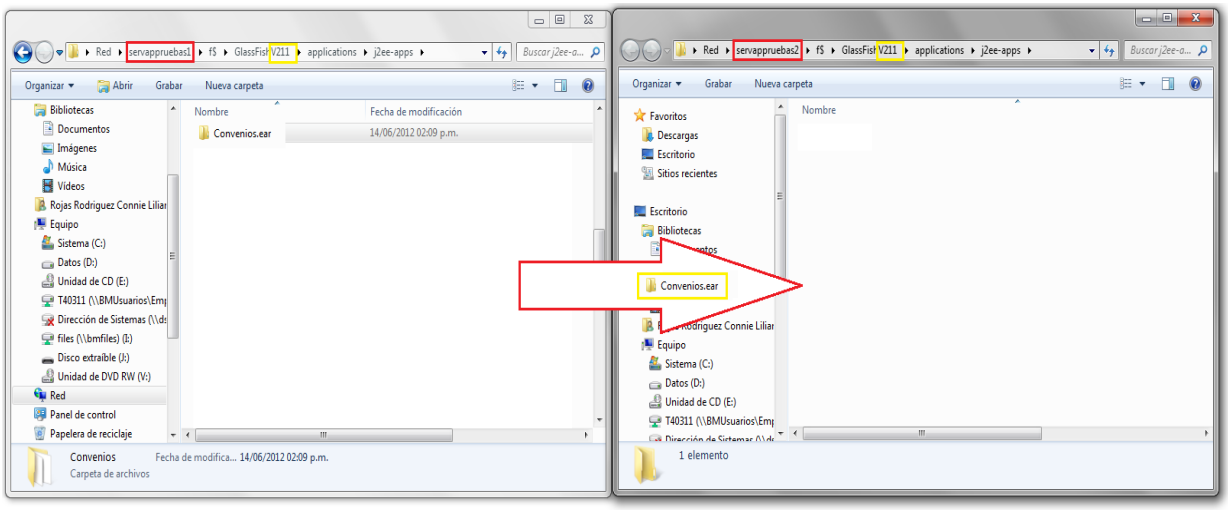

Figura 3.2. La carpeta de la aplicación, sólo se explota en el servidor principal, servidor1

En la figura 3.3 vemos cómo pueden visualizarse las aplicaciones simultáneamente en ambos servidores. Inmediatamente después del despliegue. Así mismo, al despliegue se generan las carpetas en ambos servidores (Contrario a la imagen de la figura 3.2 en que sólo se genera la del servidor administrador).

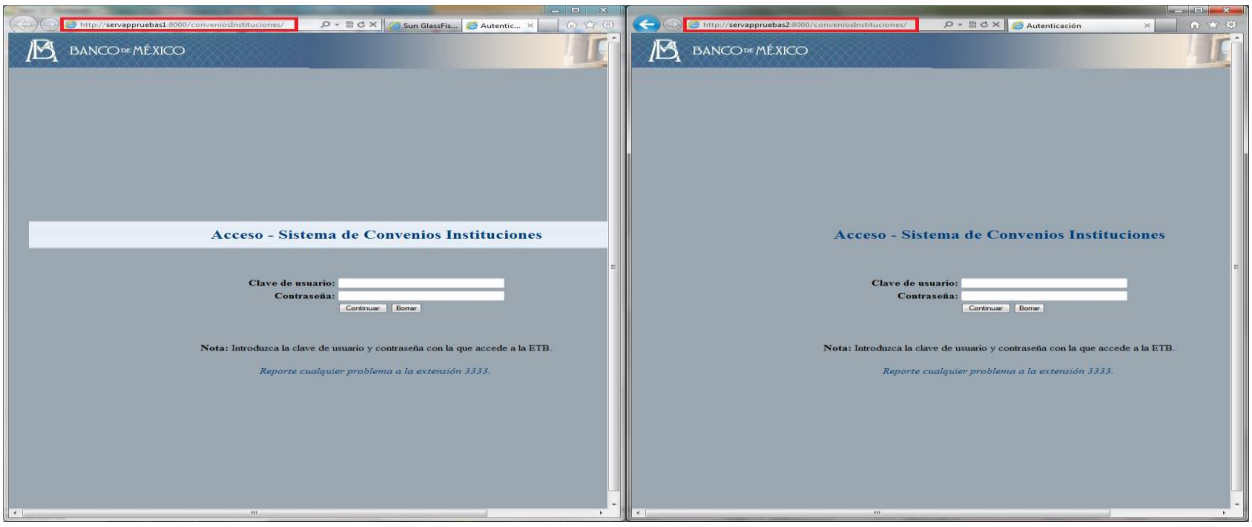

Figura 3.3 Sin necesidad de copias manuales, la aplicación se puede llamar y visualizar desde ambos servidores (principal y secundario)

3.1.2 **Optimización de tiempo de los administradores.** Los administradores no aplicarán tiempos innecesarios en la actualización de "partes" de las aplicaciones.

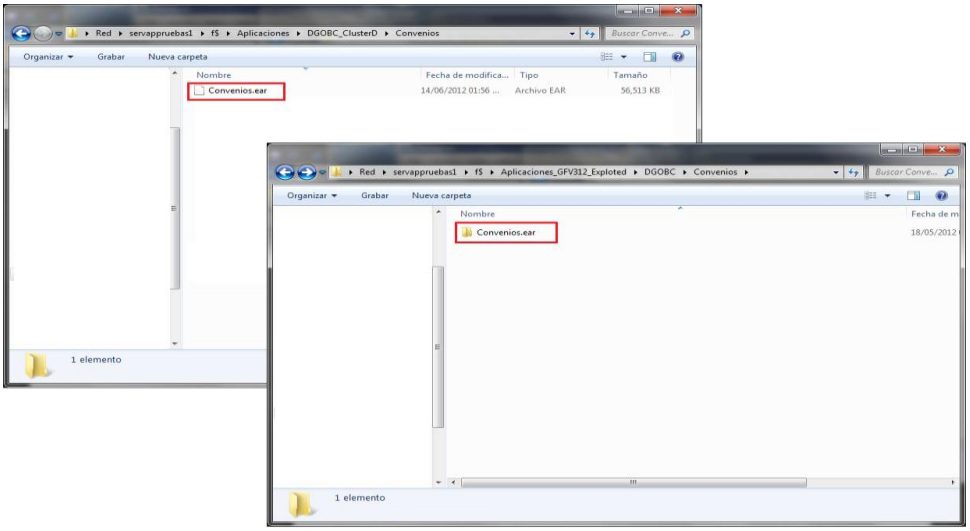

Figura 3.4 Izq. carpeta de respaldos con archivos empaquetados. Der. carpetas explotadas.

Siguiendo con el ejemplo anterior, y como ya lo apunté, en la versión 2.1 y 2.1.1 de GlassFish se requería el paquete completo y sin explotar de una aplicación para poder hacer el despliegue. Esto implicaba que el más mínimo cambio (a excepción de las JSPs) requería la generación del archivo .ear o .war completo y el tiempo de undeploy (con todos los posibles contratiempos) y deploy correspondientes.

Suponiendo que se requiere sólo sustituir un jsp, bastará con eliminar la antigua versión y copiar sobre la ubicación adecuada, la versión actualizada del recurso. Ver figura 3.5

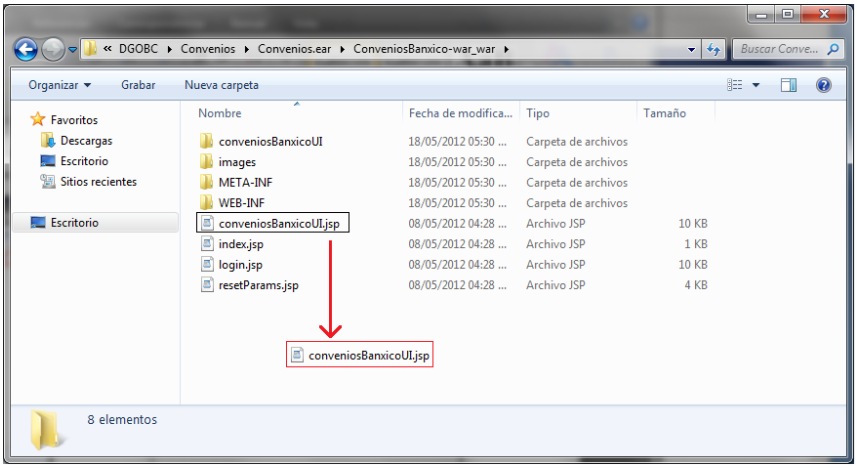

Figura 3.5. Para sustituir una parte de la aplicación (pueden ser archivos o carpetas por ejemplo de hojas de estilos) es necesario conocer su ubicación y sustituir.

La ventaja de la versión 3.1.2 es que al tener una aplicación en un paquete ya explotado podemos acceder a los elementos de la misma y modificar sólo las necesarias sin necesidad de bajar la aplicación, armar de nuevo el paquete y volverla a subir; ahorrando recursos de disco duro, memoria y tiempo.

3.1.3 **Mejor aprovechamiento de recursos y carpetas compartidas en red**.

En versiones anteriores de GlassFish y por necesidades de ordenamiento de la empresa, se requería crear una carpeta de respaldos y una carpeta para las aplicaciones “explotadas”, lo que ocasionaba una duplicidad en la información y un mal aprovechamiento del disco duro en cada servidor.

En el caso de la versión 3.1.2 de GlassFish este problema ha sido notoriamente mejorado, ya que la carpeta de respaldo contiene las aplicaciones ya explotadas, es decir que es la misma por lo que no hay duplicidad de información y así continuamos con un mejor aprovechamiento del espacio destinado para los datos en cada servidor.

En general, ahora es posible aprovechar de mejor forma los recursos humanos y tecnológicos.

Conclusiones

Los servidores de aplicaciones, en mi caso, el servidor GlassFish, han sido de gran utilidad pues al estar desarrollado completamente sobre la plataforma java y bajo los estándares J2EE permiten la implantación de las nuevas aplicaciones fácilmente debido a la entera compatibilidad del servidor con éstas.

Encontrar e instalar la nueva versión 3.1.2 de GlassFish permite optimizar recursos y aprovechar funcionalidades explotables desde el desarrollo java.

Con las versiones anteriores en instalaciones similares, es decir, en cluster la administración no era automatizada pues debía borrar (al quitar) y copiar (al desplegar) una nueva aplicación en el servidor. La comunicación segura que nos proporciona la herramienta Cygwin hace ahora tales funciones, minimizando los errores que se presentaban antes durante el borrado/copiado de las aplicaciones tales como la corrupción de los archivos y por tanto el mal funcionamiento de la aplicación en el servidor cliente.

Podemos concluir que los objetivos planteados se cubrieron satisfactoriamente, por un lado con la optimización de tiempos tanto del lado de los usuarios como de los administradores y por otro lado con la generación del manual de instalación de GlassFish versión 3.1.2 que contiene paso a paso, los requerimientos básicos para la configuración del sistema.

La mejora en el aprovechamiento de recursos dará mayores frutos con el paso del tiempo redituando a mediano plazo.

Glosario de términos

API: Application Programming Interface. Es el conjunto de funciones métodos (en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son pequeñas aplicaciones usadas por otras aplicaciones.

Balanceo de carga: Es una técnica utilizada para distribuir las peticiones entre varios servidores de tal forma que todos los servidores respondan al mismo número de peticiones.

Cluster: (clustering o asociación): Los clusters permiten asociar máquinas y servidores para que actúen de forma conjunta como una única instancia. La creación de un cluster permite el balanceo de carga y la recuperación frente a fallos.

Cygwin: Es la herramienta SecureShell que usamos como interfaz de comunicación entre los miembros del cluster, la comunicación se lleva a cabo en forma segura y directa.

DAS: Domain Admin Server, es el servidor principal del cluster. En él se instala la consola administradora y es el orquestador de las modificaciones para todo el dominio.

Dominio: unidad administrativa. Sirve para declarar varios servidores, aplicaciones, etc. y que todos ellos estén asociados mediante el nombre del dominio.

EJB: Un Java Bean es un componente utilizado en Java que permite agrupar funcionalidades mientras que el Enterprise Java Bean es un Java Bean deployable.

GlassFish: Es un servidor de aplicaciones, en versiones de código libre y propietario. Desarrollado por Sun Microsystems, actualmente propiedad de Oracle.

HTTPS: Hypertext Transfer Protocol Secure, es una combinación del protocolo HTTP y protocolos criptográficos. Se emplea para lograr conexiones más seguras en los casos en que se intercambie información sensible (como claves) en internet.

Interfaz: es una conexión entre dos elementos de niveles o naturalezas diversas.

J2EE: refiere a un estándar o a las aplicaciones JAVA desarrolladas bajo dicho estándar. Define rigurosamente un conjunto de servicios que un servidor de aplicaciones debe tener, junto con una API estándar para acceder a estos servicios.

Java NIO: es una colección de API's java que ofrecen operaciones de Entrada/Salida

JMS: Java Message Service, es un estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes.

JSP: es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Nodo: entiéndase cada servidor que forma parte del cluster.

Open Source: es el término con el que se conoce al software distribuido y desarrollado libremente.

Path: Es la ruta, la referencia exacta de ubicación de un archivo o directorio dentro de cierto sistema operativo.

Ping: es una utilidad diagnóstica en informática para comprobar la comunicación entre servidores/nodos de una red locales y remotos

Plug-in: Un plug-in es un módulo de hardware o software que añade una característica o servicio específico a un sistema más grande. **E**l nuevo componente se "enchufa" simplemente al sistema existente.

Recuperación ante fallos (failover): cualquier implementación que permite evitar la caída de un sistema cuando una máquina deja de funcionar o funciona incorrectamente.

Red: es un conjunto de equipos informáticos y software conectados entre sí por medio de dispositivos físicos que envían y reciben datos, con la finalidad de compartir información, recursos y ofrecer servicios.

Servidor de Aplicaciones: implementación del software que permite la administración de aplicaciones en él montadas para su utilización funcional como aplicación.

Servidor Web: es la parte visible del servidor de aplicaciones. Utiliza los protocolos HTTP y SSL (seguro) para comunicarse.

Servlet: clase java, forma parte de una aplicación web; recibe peticiones http y da una respuesta por medio de las instrucciones request y response. Proviene de los applets que son pequeñas aplicaciones que se ejecutan dentro del contexto de un navegador web.

SGBD: Sistema de Gestión de Bases de Datos. Es una agrupación de programas que sirven para definir, construir y manipular una base de datos.

SSH: SecureShell, es un intérprete de órdenes seguras y es también el nombre de un protocolo y del programa que lo implementa. Es útil para accesar de manera remota a equipos de una red.

Thread: es la unidad de procesamiento más pequeña (de una aplicación) que puede ser procesada por el SO. Comparte recursos y permite a las aplicaciones realizar tareas concurrentes.

Tomcat: Es un contenedor de servlets y JSP es un antecesor de los servidores de aplicaciones (del que puede, actualmente, formar parte).

TopLink: Es un ORM (mapeo objeto-relacional); paquete para desarrolladores de Java. Proporciona estructura para el almacenamiento de objetos Java en una base de datos relacional o para convertir objetos Java a documentos XML. (forma parte del servidor de aplicaciones).

URL: es una secuencia de caracteres, bajo cierto estándar, usado para referenciar objetos en internet.

WebLogic: es un servidor de aplicaciones J2EE y también un servidor web HTTP de Oracle, para Unix, Linux, Microsoft Windows, y otras plataformas. Es propietario.

Windows Server: es el nombre de un SO de Microsoft para servidores.

Referencias

[1] http://www.alegsa.com.ar/Dic/servidor%20de%20aplicaciones.php, Concepto servidor de aplicaciones, última consulta 12/06/12

[2] http://www.masadelante.com/faqs/plug-in última consulta 12/06/12

[3]http://www.marlonj.com/blog/2009/10/que-es-GlassFish/ , Concepto de GlassFish, última consulta 19/04/2012

[4] http://bannysolano.wordpress.com/2009/08/23/%C2%BFque-es-GlassFish/ , Características de GlassFish, última consulta 21/04/2012

[5] http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/sa/sesion1-apuntes.htm , Introducción a los servidores de aplicaciones, última consulta 10/06/12

[6] https://blogs.oracle.com/jaimecid/entry/presentacion_GlassFish_javaee5 , Servidor J2EE GlassFish, última consulta 30/04/12

[7] http://GlassFish.java.net/ , GlassFish 3.1.2, última consulta 17/04/2012

[8] http://jnetmatservice.wordpress.com/tag/GlassFish/ , Desarrollador del software, última consulta 30/04/12

[9] http://www.programacion.com/articulo/bea_weblogic:_introduccion_129 , Plataforma J2EE, última consulta 16/05/12

[10] http://searchsoa.techtarget.com/definition/WebLogic , Concepto de WebLogic, última consulta 20/04/12

[11] http://glassfish.java.net/public/comparing_v2_and_v3.html , Comparación entre versiones de GlassFish, última consulta 20/04/2012

[12] http://www.microsoft.com/latam/technet/windowsserver/longhorn/evaluate/whitepaper.mspx , Información técnica Windows Server 2008, consulta 25/06/12

[13]http://www.elprofesionaldelainformacion.com/contenidos/1993/septiembre/el_entorno_windows_qu_es_y_qu_aporta_a_la_documentacin.html , Entorno Windows, última consulta 21/04/2012

[14] http://www.ecured.cu/index.php/Balanceo_de_cargas , Balanceo de cargas, última consulta 22/06/2012

[15] http://mmc.igeofcu.unam.mx/LuCAS/Manuales-LuCAS/doc-cluster-computadoras/doc-cluster-computadoras-html/node8.html , Concepto de cluster, última consulta 20/06/2012

[16] http://docs.oracle.com/ , Documentación relativa a GlassFishV3.1.2. (Release notes, Server) , última consulta 27/06/12

[17] http://www.cygwin.com/cygwin-ug-net/overview.html#what-is-it , Cygwin overview , última consulta 12/06/12

[18] http://www.cygwin.com/ , Descarga de Cygwin, última consulta 28/02/12

[19] http://programoweb.com/72076/el-demonio-ssh-manual-administracion-de-redes-con-linux/ , Demonio SSH, última consulta 27/04/2012

[20]http://sicuz.unizar.es/documen/doc/windows.PDF , entorno Windows, última consulta 05/03/12

[21] http://www.masadelante.com/faqs/plug-in , plug-in, última consulta 12/06/12

[22] http://www.alegsa.com.ar/Dic/sgbd.php , Definición SGBD, consulta 10/06/12

[23] http://www.informationweek.com.mx/networking/el-failover-perfecto/ , Failover, consulta 11/06/12

[24] http://www.osmosislatina.com/java/ejb.htm , Def. EJB, consulta 02/06/12