KAAZING

CONFIGURING A VIRTUAL PRIVATE CONNECTION USING KAAZING WEBSOCKET GATEWAY

A TECHNICAL WHITEPAPER

Configuring a Virtual Private Connection

Overview

Kaazing WebSocket Gateway can be configured to allow TCP (and UDP) clients to connect to servers over the Web without the need for *any* special Kaazing or WebSocket libraries, thus creating a *virtual private connection*.

Kaazing WebSocket Gateway is designed not only to proxy TCP protocols using the WebSocket protocol, but it can also be run in a reverse mode of WebSocket protocol to TCP. This design allows system and network administrators to configure two or more gateways so applications can traverse the Web securely through firewalls and proxy servers. Enterprises and startup companies can now deliver sophisticated server-to-server systems and rich client applications over a LAN or WAN web infrastructure in the same manner as conventional distributed applications, all without the expense or complexity of a private line.

This allows many types of applications to be constructed: real-time supply chain, inter-cloud and cloud-to-cloud communication, fixed-income exchanges, local office management, external cloud applications using data remaining within the enterprise, and so on. Any legacy distributed application, like the one shown in Figure 1 that relies on having special ports opened on the corporate firewall, can now be extended over the Web with TLS encryption, traverse through firewalls and proxies, and retain all of its functionality.

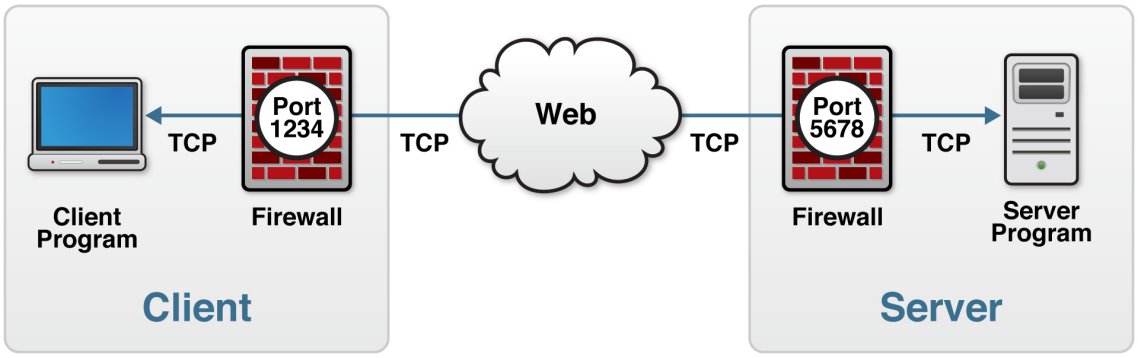


Figure 1. Standard Network Topology in Which a Client Communicates with a Server

To get this to work you need at least two Kaazing WebSocket Gateways. One configured to proxy TCP traffic over WebSocket and another configured in reverse to accept WebSocket traffic and proxy it to a TCP-based backend server or program. This way, you can set up communication between mutually exclusive networks, traverse proxy servers, and so on.

Figure 2 shows an example virtual private connection topology in which TCP traffic from the Client instance is proxied to the Server instance using WebSocket.

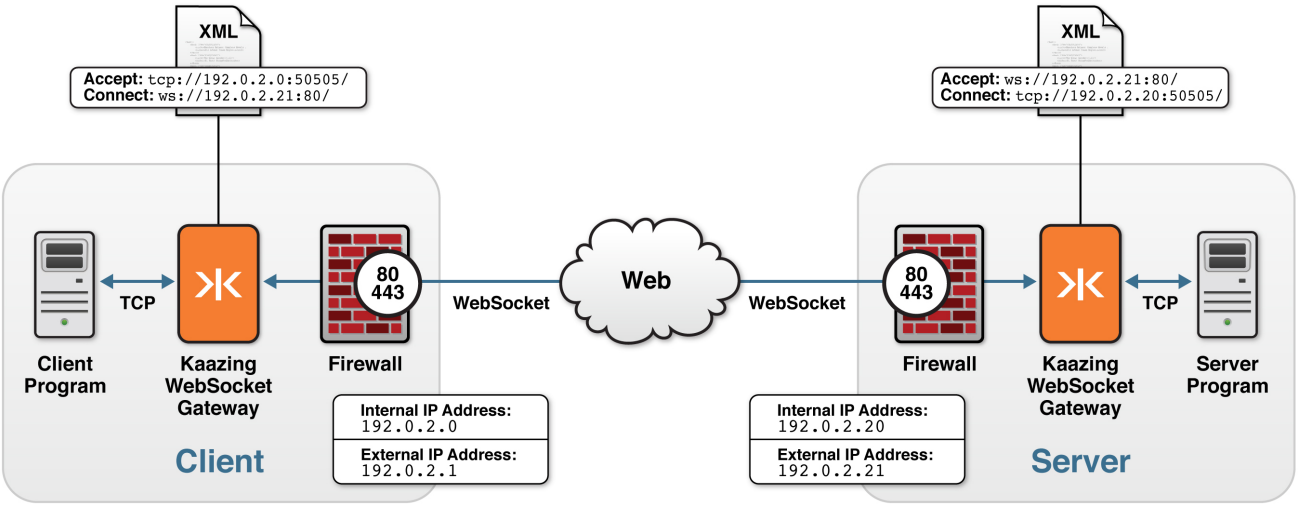


Figure 2. Kaazing Virtual Private Network topology

To configure a virtual private connection like the one shown in Figure 2, perform the following steps that are described in this whitepaper:

1. Configure the Client Instance
2. Configure the Server Instance
3. Use the Virtual Private Connection
4. Secure the Virtual Private Connection
5. Troubleshooting

Configure the Client Instance

This section describes how to configure the Client instance shown in Figure 2. The examples assume that the Client instance uses the ports and IP addresses shown in Table 1. Because this setup is configured in a cloud environment, there will be an internal and an external IP address. Kaazing WebSocket Gateway can be configured to do the appropriate network address mapping.

Table 1. Client Instance Details

Description	Value
Open ports	80, 443, 50505, 8000
Internal IP Address	192.0.2.0
External IP Address	192.0.2.1

Perform the following steps to configure the Client instance shown in Figure 2:

1. Create and connect to a cloud server instance.
   These instructions assume that you have already set up and connected to an Amazon EC2 instance. However, if you need help getting a cloud server up and running, then refer to the *Installing Kaazing WebSocket Gateway-HTML5 Edition on an Amazon EC2 Cloud Server* white paper and perform only the instructions in the following two sections of the white paper:

   - Create an EC2 instance
   - Connect to the EC2 instance

2. Download and install the Full[1] Kaazing WebSocket Gateway – HTML5 Edition.
   a. Download the Gateway software using the `wget` command on the cloud machine, or download the software from the Kaazing download page to your local machine and then transfer the file to the cloud machine using FTP. For example (ensure the command is all on one line):
   ```
   wget
   http://kaazingcorp.cachefly.net/com/download/html5/3.2.0.9/kaazing-websocket-gateway-html5-3.2.0.tar.gz
   ```
   b. Extract the contents of the downloaded file to your home directory. For example:
   ```
   tar -xf kaazing-websocket-gateway-html5-3.2.0.tar.gz
   ```
3. Edit the property defaults section in the *GATEWAY_HOME*`/conf/gateway-config-minimal.xml` on the Client Instance and change the `gateway.hostname` property value from `localhost` to either the host name or the (external) IP address of the Client instance. For example:

```
<properties>
  <property>
    <name>gateway.hostname</name>
    <value>192.0.2.1</value>
  </property>
  <property>
    <name>gateway.base.port</name>
    <value>8000</value>
  </property>
</properties>
```

[1] There are two Gateway download types: Base and Full. The Base download contains a minimal download of the Gateway for experienced users who are ready to use the Gateway in production. Full installation contains the Base download plus the Gateway documentation and the services needed to run the Gateway demos.

For helpful tips about editing files on a cloud machine, see *Setting Up Kaazing WebSocket Gateway*.

4. Configure network protocol bindings for all services.
   Edit the `gateway-config-minimal.xml` file on the Client Instance to add the `<service-defaults>` and `<accept-options>` elements, and configure the `<protocol.bind>` element to specify a default value for each network protocol binding for all services. These elements must be added before the ending `</gateway-config>` element. The following example shows the portion of the `gateway-config.xml` file with sample network protocol bindings for the `ws` and `http` services:

```
<!-- Service Defaults -->
<service-defaults>
  <accept-options>
    <ws.bind>80</ws.bind>
    <http.bind>8000</http.bind>
  </accept-options>
</service-defaults>
```

5. Add a proxy service that accepts internal TCP traffic on port 50505 on the internal IP address of the Client Instance, and proxies to the external IP address on the Server Instance over port 80 using WebSocket. For example:

```
<!-- Proxy service -->
<service>
  <accept>tcp://192.0.2.0:50505/</accept>
  <type>proxy</type>
  <properties>
    <connect>ws://192.0.2.21:80/</connect>
  </properties>
</service>
```

   Important: In the example, note the use of trailing slashes (/) in the `<accept>` and `<connect>` URLs for WebSocket connections using a minimal path. This is not required for the TCP URLs.

6. Save the `gateway-config-minimal.xml` file and exit.
   Appendix A shows the complete `gateway-config-minimal.xml` file for the Client instance.
7. Start Kaazing WebSocket Gateway on the Client instance as follows:

```
cd ~/kaazing-websocket-gateway-html5-3.2.0/bin
./gateway.start --config ../conf/gateway-config-minimal.xml
```

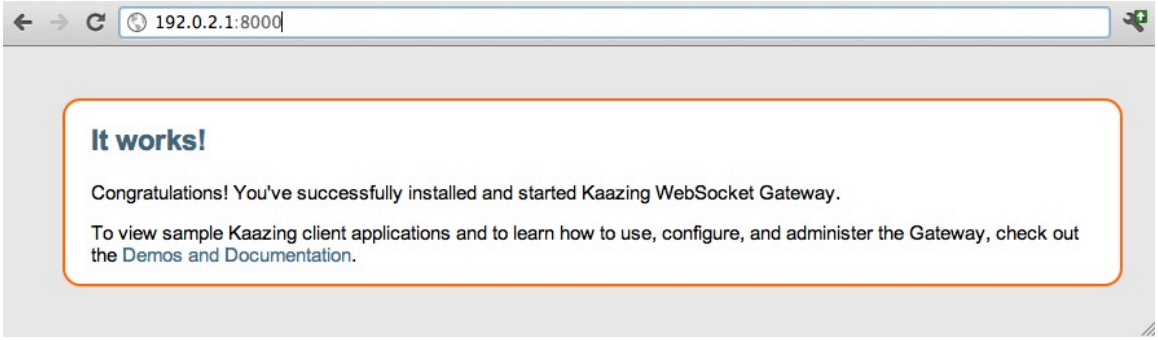
The Gateway should display startup messages similar to those shown in the following example.

Gateway Startup Messages

```
INFO Kaazing WebSocket Gateway - HTML5 Edition (3.2.0)
INFO Configuration file: GATEWAY_HOME/conf/gateway-config.xml
INFO  Checking license information
INFO    Found license: Developer License, Maximum 50 connections,
See LICENSE.txt
INFO  Starting server
INFO  Starting services
INFO    http://192.0.2.1:8000/ @ 0.0.0.0:8000
INFO    tcp://192.0.2.0:50505/
INFO  Started services
INFO  Started server successfully in 0.033 secs at 2044-06-06
00:54:17
```

Note: On certain operating systems, you may need root privileges (`sudo`) to start processes that listen on ports less than 1024.

8. Verify the Gateway setup by opening a browser and navigating to the Gateway home page, hosted by the directory service on port 8000 (for example, enter the URL: `http://192.0.2.1:8000`) on your cloud server. **Note**: You can later remove the directory service. You should see the following message:



Configure the Server Instance

This section describes how to configure the Server instance shown in Figure 2. The examples assume that the Server instance uses the ports and IP addresses shown in Table 2. Because this setup is configured in a cloud environment, there will be an internal and an external IP address. Kaazing WebSocket Gateway can be configured to do the appropriate network address mapping.

Table 2. Server Instance Details

Description	Value
Open ports	80, 443, 50505, 8000
Internal IP Address	192.0.2.20
External IP Address	192.0.2.21

Perform the following steps to configure the Server instance shown in Figure 2:

1. Create and connect to a cloud server instance.
   These instructions assume that you have already set up and connected to a cloud server. However, if you need help getting a cloud server up and running, then refer to the *Installing Kaazing WebSocket Gateway-HTML5 Edition on an Amazon EC2 Cloud Server* white paper and perform only the instructions in the following two sections of the white paper:

   - Create an EC2 instance
   - Connect to the EC2 instance

2. Download and install the Full[2] Kaazing WebSocket Gateway – HTML5 Edition.
   a. Download the Gateway software using the `wget` command on the cloud machine, or download the software from the Kaazing download page to your local machine and then transfer the file to the cloud machine using FTP. For example (ensure the command is all on one line):
   ```
   wget
   http://kaazingcorp.cachefly.net/com/download/html5/3.2.0.9/kaa
   zing-websocket-gateway-html5-3.2.0.tar.gz
   ```
   b. Extract the contents of the downloaded file to your home directory. For example:
   ```
   tar -xf kaazing-websocket-gateway-html5-3.2.0.tar.gz
   ```

[2] There are two Gateway download types: Base and Full. The Base download contains a minimal download of the Gateway for experienced users who are ready to use the Gateway in production. Full installation contains the Base download plus the Gateway documentation and the services needed to run the Gateway demos.

3. Edit the property defaults section in the *GATEWAY_HOME*/conf/gateway-config-minimal.xml on the Server Instance and change the gateway.hostname property value from localhost to either the host name or (external) IP address of the Server instance. For example:

```
<properties>
  <property>
    <name>gateway.hostname</name>
   <value>192.0.2.21</value>
  </property>
  <property>
     <name>gateway.base.port</name>
     <value>8000</value>
  </property>
</properties>
```

For helpful tips about editing files on a cloud machine, see *Setting Up Kaazing WebSocket Gateway*.

4. Configure network protocol bindings for all services.
To do this, edit the gateway-config-minimal.xml file on the Server Instance to add the <service-defaults> and <accept-options> elements, and configure the <protocol.bind> element to specify a default value for each network protocol binding for all services. These elements must be added before the ending </gateway-config> element.

The following example shows the portion of the gateway-config-minimal.xml file with sample network protocol bindings for the ws and http services:

```
<!-- Service Defaults -->
<service-defaults>
  <accept-options>
    <ws.bind>80</ws.bind>
    <http.bind>8000</http.bind>
  </accept-options>
</service-defaults>
```

5. Add a proxy service that accepts WebSocket traffic on port 80 on the external IP address of the Server instance, and proxies that using TCP to a back-end server program that is listening on port 50505 on the Server instance's internal IP address. For example:

```
<!-- Proxy service -->
<service>
  <accept>ws://192.0.2.21:80/</accept>
  <type>proxy</type>
  <properties>
    <connect>tcp://192.0.2.20:50505/</connect>
  </properties>
</service>
```

Important: In the example, note the use of trailing slashes (/) in the `<accept>` and `<connect>` URLs for WebSocket connections using a minimal path. This is not required for the TCP URLs.

6. Save the `gateway-config-minimal.xml` file and exit.
   Appendix B shows the complete `gateway-config-minimal.xml` file for the Server instance.
7. Start Kaazing WebSocket Gateway on the Server instance as follows:

```
cd ~/kaazing-websocket-gateway-html5-3.2.0/bin
./gateway.start --config ../conf/gateway-config-minimal.xml
```

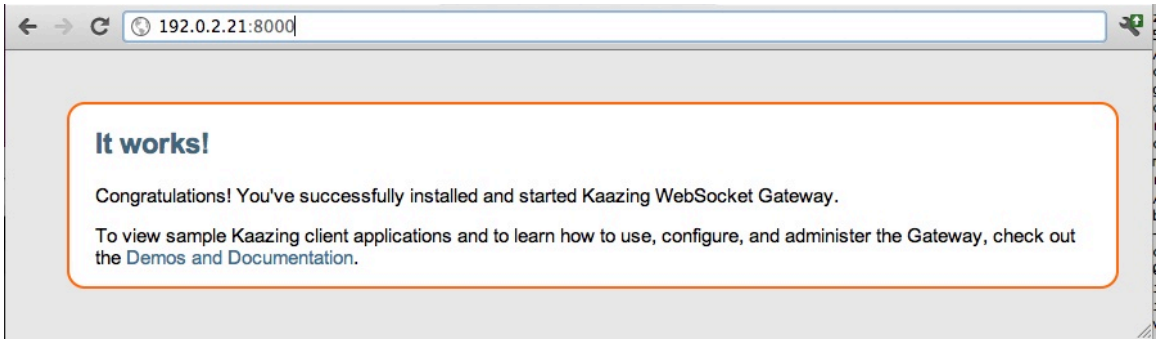
The Gateway should display startup messages similar to those shown the following example.

Gateway Startup Messages on the Server Instance

```
INFO Kaazing WebSocket Gateway - HTML5 Edition (3.2.0)
INFO Configuration file: GATEWAY_HOME/conf/gateway-config.xml
INFO  Checking license information
INFO    Found license: Developer License, Maximum 50 connections,
        See LICENSE.txt
INFO  Starting server
INFO  Starting services
INFO    http://192.0.2.21:8000/ @ 0.0.0.0:8000
INFO    ws://192.0.2.21:80/ @ 0.0.0.0:80
INFO  Started services
INFO  Started server successfully in 0.033 secs at 2044-06-06
00:54:17
```

Note: On certain operating systems, you may need root privileges (`sudo`) to start processes that listen on ports less than 1024.

8. Verify the Gateway setup by opening a browser and navigating to the Gateway home page, hosted by the directory service on port 8000 (for example, enter the URL: http://192.0.2.21:8000) on your cloud server. **Note**: You can later remove the directory service. You should see the following message:



Using the Virtual Private Connection

The following example uses a simple Perl "echo" client program on the Client instance to talk to a Perl "echo" server program on the Server instance, using WebSocket.

Perform the following steps:

1. Open your favorite text editor on the Client instance and create a new Perl file called `echo-server.pl` (for example, create it in a folder called `/home/username/kaazing`).

2. Add the Perl code shown in the following example:

Code for the Echo Server Program

```
#!/usr/local/bin/perl
# usage: echo-server.pl [port]
# Echo server
use strict;
use Socket;
use IO::Handle;
use constant MY_ECHO_PORT => 50505;

my ($bytes_out,$bytes_in) = (0,0);
my $port = shift || MY_ECHO_PORT;
my $protocol = getprotobyname('tcp');

$SIG{'INT'} = sub {
  print STDERR "bytes_sent = $bytes_out, bytes_received
=                  $bytes_in\n";
  exit 0;
};

socket(SOCK, AF_INET, SOCK_STREAM, $protocol) or
  die "socket() failed: $!";
setsockopt(SOCK,SOL_SOCKET,SO_REUSEADDR,1) or
  die "Can't set SO_REUSEADDR: $!";

my $my_addr = sockaddr_in($port,INADDR_ANY);
bind(SOCK,$my_addr) or
  die "bind() failed: $!";
listen(SOCK,SOMAXCONN) or
  die "listen() failed: $!";

warn "Waiting for incoming connections on port $port...\n";

while(1) {
  next unless my $remote_addr = accept(SESSION,SOCK);
  my ($port,$hisaddr) = sockaddr_in($remote_addr);
  warn "Connection from [",inet_ntoa($hisaddr),",$port]\n";

  SESSION->autoflush(1);
  while(<SESSION>) {
    $bytes_in += length($_);
    chomp;

    my $msg_out = "$_\n";
    print SESSION $msg_out;
    warn "Echoing back: $msg_out\n";
    $bytes_out += length($msg_out);
    }

    warn "Connection from
[",inet_ntoa($hisaddr),",$port]           finished\n";
    close SESSI}
```

3. Save the echo-server.pl file to the /home/username/kaazing folder on the Server instance.

4. On the Server instance, run the echo server program, as follows:

```
perl echo-server.pl
```

After the program starts successfully, you should see the following server message:

```
Waiting for incoming connections on port 50505...
```

5. Create another Perl file, called `echo-client.pl` (for example, in a folder called `/home/username/kaazing`) on the Client instance.

6. Add the Perl code, shown in the following example, to the `echo-client.pl` script.

Code for the Echo Client Perl Program

```
#!/usr/local/bin/perl
# Echo client
# usage: echo-client.pl [host] [port]

use strict;
use Socket;
use IO::Handle;

my ($bytes_out,$bytes_in) = (0,0);
my $host = shift || 'localhost';
my $port = shift || getservbyname('echo','tcp');
my $protocol = getprotobyname('tcp');

$host = inet_aton($host) or
  die "$host: unknown host";

socket(SOCK, AF_INET, SOCK_STREAM, $protocol) or
  die "socket() failed: $!";

my $dest_addr = sockaddr_in($port,$host);

connect(SOCK,$dest_addr) or
  die "connect() failed: $!";

SOCK->autoflush(1);

while (my $msg_out = <>) {
  print SOCK $msg_out;
  my $msg_in = <SOCK>;
  print "Kaazing: $msg_in";

  $bytes_out += length($msg_out);
  $bytes_in  += length($msg_in);
}

close SOCK;
print STDERR "bytes_sent = $bytes_out, bytes_received
=               $bytes_in\n";
```

7. Save the file and copy it to the `/home/username/kaazing` folder on the Client instance.

8. On the Client instance, run the `echo-client.pl` program. The following command includes the Client instance's internal IP address and port 50505:

```
perl echo-client.pl 192.0.2.0 50505
```

After the script starts successfully, you should see a message in the Server instance's terminal window where you started the echo-server console:

```
Connection from [192.0.2.20,39535]
```

9. Type some text in the Client instance's terminal window where you started the `echo-client.pl` program:

```
Hello, Virtual Private Connection
```

10. If everything is configured correctly, you should see your text echoed back to you:

```
Kaazing: Hello, Virtual Private Connection
```

Securing the Virtual Private Connection

To secure the wire traffic, which will result in greater success when traversing intermediate servers---such as proxy servers, firewalls and load balancers, you can use WebSocket Secure scheme (WebSocket over TLS). This relies on the proper configuration of certificates and the use of a keystore, which is outside the scope of this document. Refer to the section titled "Configuring Wire Traffic Encryption" in the Kaazing *Security Guide* for more information.

Troubleshooting

This section describes some common errors and actions you can take to resolve them.

ERROR: CANNOT ASSIGN REQUESTED ADDRESS ERROR

While starting the Gateway, you receive the following error message:

```
java.lang.RuntimeException: Unable to bind to resource:
wss://192.0.2.1:443/ cause: Cannot assign requested address
.
.
.
```

Cause

This error is most likely caused by improper network address mapping configuration. For example, in this case, you may not have specified a mapping for the external address 192.0.2.1:9000 to the correct internal address.

Action

Add or update the *protocol*`.bind` element to your `gateway-config-minimal.xml` file. See the *Administrator's Guide* for more details.

ERROR: PERMISSION DENIED

While starting the Gateway, the following error message is displayed:

```
java.lang.RuntimeException: Unable to bind to resource:
ws://192.0.2.21:80/ @ ws://192.0.2.20:80/ cause: Permission denied
```

Cause

On certain operating systems, you may need root privileges to start processes that listen on ports less than 1024.

Action

Update the port you are accepting on, or obtain root privileges.

Appendix A Client Instance gateway-config-minimal.xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--

    Copyright (c) 2007-2011, Kaazing Corporation. All rights reserved.

-->

<gateway-config xmlns="http://xmlns.kaazing.com/gateway-config/excalibur">

  <!-- Property defaults -->

  <properties>
    <property>
      <name>gateway.hostname</name>
      <value>192.0.2.1</value>
    </property>
    <property>
      <name>gateway.base.port</name>
      <value>8000</value>
    </property>
  </properties>

  <!-- Directory service -->

  <service>
    <accept>http://${gateway.hostname}:${gateway.base.port}/</accept>

    <type>directory</type>
    <properties>
      <directory>/base</directory>
      <welcome-file>index.html</welcome-file>
      <error-pages-directory>/error-pages</error-pages-directory>
      <options>indexes</options>
    </properties>
  </service>

  <!-- Proxy service -->
  <service>
    <accept>tcp://192.0.2.0:50505/</accept>
    <type>proxy</type>
    <properties>
      <connect>ws://192.0.2.21:80/</connect>
    </properties>
  </service>

  <!-- Service Defaults -->
  <service-defaults>
    <accept-options>
      <ws.bind>80</ws.bind>
      <http.bind>8000</http.bind>
    </accept-options>
  </service-defaults>

</gateway-config>
```

Appendix B Server Instance gateway-config-minimal.xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--

    Copyright (c) 2007-2011, Kaazing Corporation. All rights reserved.

-->

<gateway-config xmlns="http://xmlns.kaazing.com/gateway-config/excalibur">

  <!-- Property defaults -->

  <properties>
    <property>
      <name>gateway.hostname</name>
      <value>192.0.2.21</value>
    </property>
    <property>
      <name>gateway.base.port</name>
      <value>8000</value>
    </property>
  </properties>

  <!-- Directory service -->

  <service>
    <accept>http://${gateway.hostname}:${gateway.base.port}/</accept>

    <type>directory</type>
    <properties>
      <directory>/base</directory>
      <welcome-file>index.html</welcome-file>
      <error-pages-directory>/error-pages</error-pages-directory>
      <options>indexes</options>
    </properties>

  </service>

  <!-- Proxy service -->
    <service>
      <accept>ws://192.0.2.21:80/</accept>
      <type>proxy</type>
      <properties>
        <connect>tcp://192.0.2.20:50505/</connect>
      </properties>
    </service>

  <!-- Service Defaults -->
  <service-defaults>
    <accept-options>
      <ws.bind>80</ws.bind>
      <http.bind>8000</http.bind>
    </accept-options>
  </service-defaults>

</gateway-config>
```

About Kaazing Corporation

Kaazing is a web infrastructure software company that enables companies to quickly deliver massively scalable, real-time enterprise web applications and significantly reduce operational costs by simplify back-end server infrastructure.

Kaazing Corporation's patent-pending, WebSocket Acceleration™ technology is the only solution in the marketplace today that facilitates full-duplex web communication, which makes it possible to extend any TCP-based messaging protocols to the Web seamlessly and reliably, without buying expensive new hardware.

Unlike older, non-standard technologies that rely on "polling" or "long-polling,"—complex and inefficient techniques that result in higher latency and skyrocketing server infrastructure costs—Kaazing delivers ultra-high application performance at the lowest operational cost.

Related Links:

- Kaazing: http://www.kaazing.com
- Kaazing Technology Network: http://tech.kaazing.com
- Kaazing Expert Training: http://www.kaazing.com/training
- Questions? E-mail doc@kaazing.com