HIGH AVAILABILITY GUIDE

COUNTERSOFT

Contents

Gemini High availability Guide

Introduction

Gemini supports high availability and can scale up out of the box with minimal configuration. This ensures that the Gemini application is always up even if we have a web server failure. Furthermore enterprises that have heavy workload on the Gemini application can improve performance and scale up by running it in a cluster, where you have more than one web server.

Most implementations of Gemini run using a single instance of a web server, see **figure 1**, which means if that web server fails then we cannot access Gemini at all.

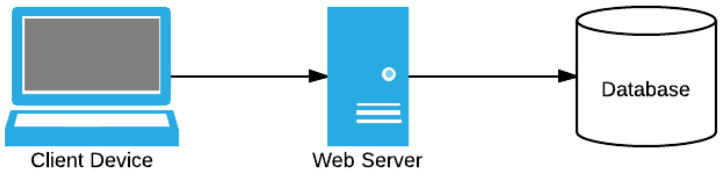

Figure 1 – Single Instance

When we need to scale up and improve performance or need to ensure that Gemini is always available even we have a web server failure we can run multiple instances of Gemini, each connected to the same Gemini database, see **figure 2**. This way we divide the workload onto several web servers, reduce the response time and increase scalability. We also guarantee that Gemini will be highly available as we are not dependant on a single web server.

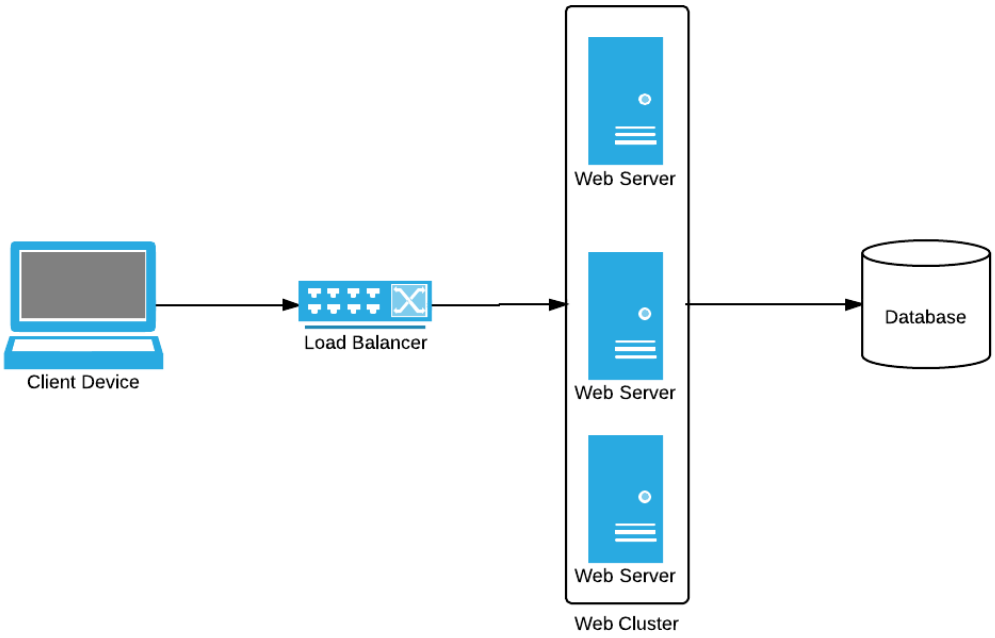

Figure 2 - Multiple Instances

This will ensure high availability of Gemini because if one web server goes down then all subsequent requests will be routed to the other web servers by the load balancer without the end user noticing any issues.

Implementation

1. Set the session state

In order to run Gemini in a web cluster we have to first configure a state server to store the end user's session data. In other words, if the load balancer routes the user to server A when the first request comes and to server B for the same user's subsequent request, we need the user's "in memory" data to carry forward to server B. To do so you need to amend the web.config file for each Gemini instance to use the state server service or SQL Server. Open the web.config file and change the sessionState mode:

```
</pages>
<sessionState mode="StateServer" stateConnectionString="tcpip=hulk:42424" />
<httpHandlers>
  <add verb="POST,GET,HEAD" path="errors.axd" type="Countersoft.Gemini.Infrastruct
</httpHandlers>
```

Figure 3 - Setting the session state

This needs to be done for each instance of Gemini.

Note that if you are using the state server service then make sure you have set it to allow remote connections by amending the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\aspnet_state\Parameters\AllowRemoteConnection registry key to the value of 1 before starting the service.

2. Set the Machine Key

In order for all web servers to use the same state server data we need to make sure that the machine key setting in all the web.config files of all Gemini instances are the same. The easiest way to do so is to open IIS, navigate to the Gemini site on one instance and double click on the machine key icon, see **figure 4**:

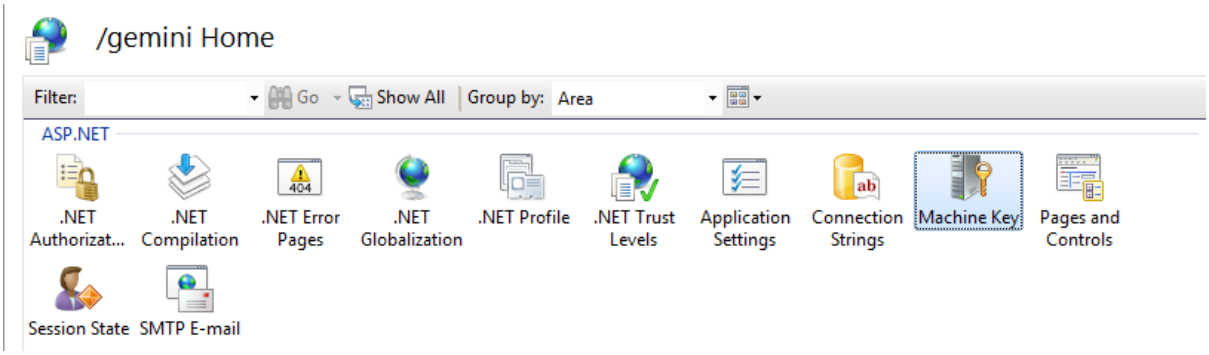

Figure 4 - Double click the "Machine Key" icon

You will be presented with the machine key screen, please un-tick all 4 checkboxes and click on Generate Keys on the right hand pane, see **figure 5:**

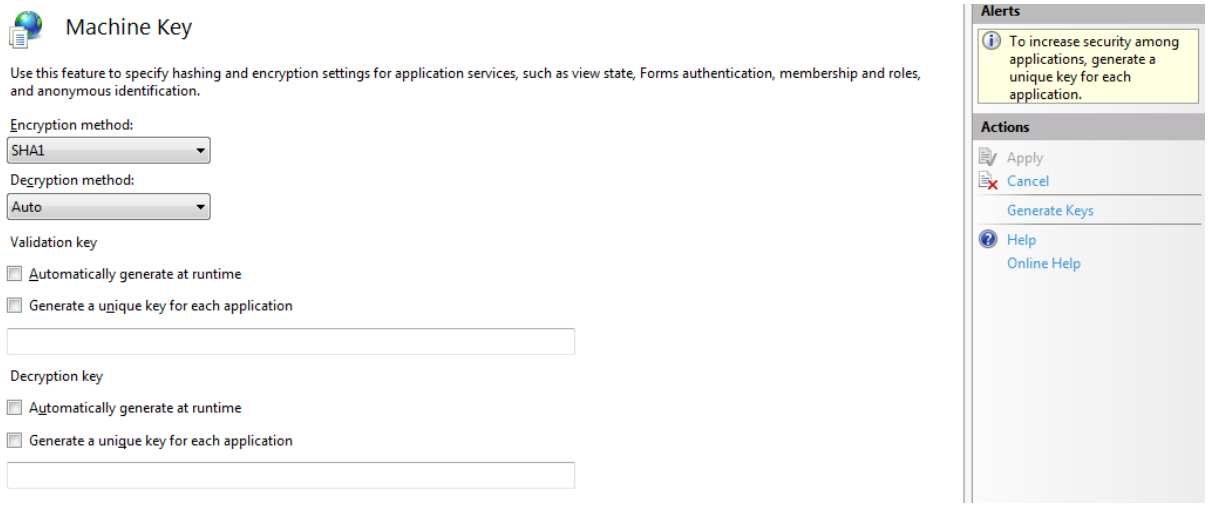

Figure 5 - Generate Keys

Click apply on the right hand pane and copy the generated machinekey section from the web.config file to all other instances, see **figure 6:**

```
    </httpModules>
    <httpRuntime executionTimeout="90" maxRequestLength="44096" useFullyQualifiedRedirectUrl="false
    <machineKey decryptionKey="BA525227C709A3224564369427EE67CE96D400234DCA7F34" validationKey="829
  </system.web>
```

Figure 6 - Copy the machinekey section

Once you have done all of the above, Gemini will take care of the rest. Each instance will register itself in the database and will know about the other instances. Gemini has a built in process that will communicate with each instance when the local cache needs to be refreshed. For example when a workspace is updated all other instances will refresh their cache with the new updated workspace.

Gemini supports timer apps which can run by any instance of Gemini. Gemini's timer job manager will synchronize all timer job execution so no 2 timer jobs of the same type run at the same time. The job manager will ensure that in such case only one job will run and the others won't.

3. Replicate any custom apps and Project Templates

If you have created any Gemini custom apps, you will need to deploy the apps to all instances. You will also need to ensure that all web app files are copied over to all other web servers. If you have any modified or new templates you will need to copy the Project Template files from the following folder to the same target folder on all web servers:

- app_data\templates\[Template Key]
- assets\images\meta[Template Key]