Amazon Auto-Scale

KENTICO REFERENCE GUIDE

Kentico
unlimited website possibilities

www.kentico.com

INTEGRATION DETAILS

Prologue

This paper summarizes the steps necessary to integrate Kentico CMS v6 and v7 with Amazon's Auto-scaling services. The goal of the paper is to explain our current support, describe related issues and offer a viable solution for clients interested in deploying their Kentico instances to the Amazon cloud, leveraging its scaling and load-balancing capabilities. If not stated otherwise, the info discussed in the text below applies to both v6 and v7. Differences are explicitly pointed out whenever necessary.

Prerequisites

It is absolutely necessary to get familiar with Kentico Web farm support and the inner workings before you continue reading through this paper. Please pay particular attention to the various settings for configuring synchronization behavior.

Assumptions and Current State

Kentico fully supports deployment to Amazon EC2. Both the web application and underlying SQL Server can run completely on the cloud. As far as scaling is concerned, vertical scaling is supported out-of-the-box. However, horizontal scaling (adding more instances to handle high load) requires some customization.

When running a website in the cloud with auto-scaling enabled, the number of Amazon EC2 instances increases/decreases based on the conditions specified (traffic threshold, resource utilization, other).

When running Kentico in a web farm environment, each **new Kentico instance** needs to be registered with the rest of the web farm using Kentico Web farm support. When new EC2 instances running Kentico are added to an existing Amazon Auto-scale group, they **need to be registered with the other Kentico web farm nodes**. **The registration does not happen entirely automatically and needs to be handled using custom code**.

Registering Kentico Instances

As explained in the Dev Guide, Kentico web farm servers are registered in two steps:

1. Adding a unique server name as an application setting to the web.config file,
2. Creating a new web farm server in Kentico Site manager.

The above steps need to be completed to get the Kentico web farm working properly. Registration needs to happen as soon as possible once the new EC2 instance is up and running. Registration should be completed before any load is directed towards this new instance.

Since the actual registration needs to happen automatically, you can develop a PowerShell script as part of the routine, and execute it when the EC2 instance status reaches a certain level.

Updating the web.config

Each Kentico instance needs to have a unique name defined in its web.config before you include it in a web farm.

- `<add key="CMSWebFarmServerName" value="<servername>"/>`

The servername has to be a unique ID of the Kentico instance within the web farm.
For v6.x you also need to enable web farm support for new instances by adding the following key:

- `<add key="CMSWebFarmEnabled" value="true"/>`

The part of the PowerShell script executed after the EC2 instance starts needs to update the web.config file to include the appropriate settings.

Kentico v7
In v7, the web farm is enabled through the settings in *Site manager → Settings → Versioning & Synchronization → Web farm*. Since configuration using settings in Site manager is stored in the DB, and all new instances point to the same DB, you only

need to enable this setting once. All new instances will automatically enable web farm support.

If you are running v7, you do not need to explicitly specify the CMSWebFarmServerName at all. The system uses the machine name as the web farm server name when auto-creating servers on application start. That means no updates are necessary when spinning new Kentico v7 instances.

Creating New Kentico Web Farm Servers

When the web.config is updated, the second step in the registration process takes place. You need to ensure that a new Kentico web farm server is created in the *Site manager → Administration → Web farm*. When registering a new server, you have to provide a display name, code name (must match the server name specified in the web.config), and server root URL (an actual URL to reach the current Kentico instance from other web farm servers).

Kentico v7

With v7 you can take advantage of the web farm server auto-create feature. The system automatically creates new servers during the application start event, using the server name specified in the web.config. The server root URL is automatically recognized based on the URL that invokes the application start. In order for auto-create to work, you have to enable the *Versioning & Synchronization → Web farm → Generate servers dynamically* setting. Again, since configuration is stored in the DB, new instances automatically recognize this setting, and create the web farm server.

It is recommended to use database notifications instead of URL notifications when using the web farm server auto-create feature.

Kentico v6

Since v6 does not have the auto-create feature available yet, you have to register web farm servers manually using the Kentico API.

You can create new web farm servers using the sample code below:

```
private void CreateWebFarmServer()
{
    // Create new web farm server object
    WebFarmServerInfo newServer = new WebFarmServerInfo();

    // Set the properties
    newServer.ServerDisplayName = "My new server";
    newServer.ServerName = "MyNewServer";
    newServer.ServerEnabled = true;
    newServer.ServerURL = "http://localhost/KenticoCMS";

    // Save the web farm server
    WebFarmServerInfoProvider.SetWebFarmServerInfo(newServer);
}
```

The `ServerName` property must match the unique server ID specified in the web.config file.

Deleting Kentico Web Farm Servers

After an existing EC2 instance is destroyed, you should remove all related web farm servers in the system. This way, the system does not produce and maintain synchronization tasks for servers that are no longer on-line and/or available.

Kentico v7
With v7 you can let the system automatically remove web farm servers on application end. If the application being shut down has created a custom web farm server, it can remove itself from the list automatically. You just need to enable the setting *Site manager → Settings → Versioning & Synchronization → Delete generated servers on application end*.

Odds are that when EC2 instances are destroyed, the application will not have enough time to complete the remove operation. This depends on Amazon's destroy routine and how fast instances are recycled (it can easily be a matter of milliseconds before the instance is gone). In such cases, you can rely on an additional custom clean-up mechanism as described below.

Kentico v6
There is no auto-remove functionality for removing old servers during the application end event in v6 and older versions. That means you need to build a

custom mechanism for removing old servers. One of the possible options is to build a custom scheduled task, which periodically checks whether the web farm servers reply, and removes the ones that are dead or idle for longer than a specified period.

The following code is a sample implementation of a custom task that goes through the list of servers, checks them and logs the results for later evaluation. If a server fails to respond 3 times in a row, it is automatically removed. The frequency of the check depends on the task configuration in *Site manager → Administration → Scheduled tasks*.

```
    private static Hashtable mUnresponsiveServers = null;

    /// </summary>
    /// <param name="ti">Info object representing the scheduled task</param>
    public string Execute(TaskInfo ti)
    {
        // Get the data
        InfoDataSet<WebFarmServerInfo> servers =
WebFarmServerInfoProvider.GetAllServers();
        if (!DataHelper.DataSourceIsEmpty(servers))
        {
            // Loop through the individual items
            foreach (WebFarmServerInfo wfsi in servers)
            {
                try
                {
                    string serverName = wfsi.ServerName;

                    // Ping the server URL
                    Ping ping = new Ping();

                    // If returned anything else than 200 OK
                    PingReply reply = ping.Send(wfsi.ServerURL);
                    if (reply.Status != IPStatus.Success)
                    {
                        if (mUnresponsiveServers == null)
                        {
                            mUnresponsiveServers = new Hashtable();
                        }

                        // Updated failed counter
                        int failedTimes =
ValidationHelper.GetInteger(mUnresponsiveServers[serverName], 0);
                        mUnresponsiveServers[serverName] = ++failedTimes;

                        // Remove server if failed to response at least 3
times
                        if (failedTimes >= 3)
                        {
                            wfsi.Delete();
```

```
                        mUnresponsiveServers.Remove(serverName);
                    }
                }
            }
            catch { // TODO: Error handling }
        }
    }

    return null;
}
```