T-404-LOKA Lokaverkefni

Marel

Cloud Analytics

Administration Manual

Students:
Gísli Rafn Guðmundsson
Gunnar Páll Gunnarsson
Jón Reginbald Ívarsson

Teacher:
Hallgrímur Arnalds
Instructor:
Elín Elísabet Torfadóttir
Examiner:
Haukur Kristinsson

May 13, 2016

Contents

1 Preface

Cloud Analytics is an *Innova* monitoring and analytics solution. It utilizes a few different tools and programs in order to be deployed correctly.

1.1 Purpose

The purpose of this document is to give a step-by-step guide for system administrators to be able to create and maintain an instance of the Cloud Analytics system.

1.2 Maintenance

This document is maintained by *Team Pretzel*.
If you have any issues or problems with this manual, please contact:

gunnar.gunnarsson2@marel.com

1.3 Version

Current document version is 1.0.0 as of 13.05.2016.

1.3.1 History

Date	Version	Comments	Author
13.05.2016	1.0.0	Initial version.	GPG

2 Tools and Programs

A few tools and programs are needed or recommended to be able to run the scripts in this manual.

2.1 PowerShell

PowerShell must be installed, preferably version 3. The ISE (Integrated Scripting Environment) editor is also recommended, which typically comes with PowerShell. Both tools should come with Windows, but can also be downloaded or updated at:
https://msdn.microsoft.com/en-us/powershell/mt173057.aspx

2.2 Azure PowerShell

Azure PowerShell adds Azure modules and is absolutely crucial to create the Azure services mentioned later in this document. Installation instructions and a download link can be found at:
https://azure.microsoft.com/en-us/documentation/articles/powershell-install-configure/

2.3 PowerShell Extensions for Microsoft SQL

PowerShell extensions for Microsoft SQL are needed to successfully create all the database tables in the system. They do not always seem to work as intended and may need to be executed in Powershell or PowerShell ISE with administrator privileges. In some cases it even seems to matter if Powershell is running in 32-bit or 64-bit mode. Newer versions of the extensions are expected to work, but have not been tested. Download using the link below. Click *Install Instructions* and find *Microsoft® Windows PowerShell Extensions for Microsoft® SQL Server® 2012*.
https://www.microsoft.com/en-us/download/details.aspx?id=29065

2.4 npm

npm is not needed in order to deploy the system, but it is needed to build the web application. Thus, npm should be installed on the build server or the computer that is used to build and deploy the web app. Node.js includes npm and can be found here:
https://nodejs.org/en/

2.5 Optional: PowerShell Tools

Install a set of tools for developing and debugging PowerShell scripts and modules in Visual Studio.
https://visualstudiogallery.msdn.microsoft.com/f65f845b-9430-4f72-a182-ae2a7b8999d7

2.6 Optional: Azure SDK

Install the Azure SDK for .NET Visual Studio 2013 using this link:
https://azure.microsoft.com/en-us/downloads/

3 Azure Provisioning

A suite of PowerShell scripts were made to make provisioning and other automation on Azure possible remotely and automatically. These scripts are used instead of manually creating and configuring services in the Azure portal. They reside in the Scripts directory in the AutomationScripts project in Visual Studio.

The PowerShell scripts can either be executed locally, or for added automation, executed on a dedicated build server, provided that all the necessary tools are installed.

3.1 Service Principal

In the authentication directory there is a script called *Azure.Rm.Service.Principal.ps1*. This script must be run on any new Azure account. It creates a user on the Azure portal to be able to run consecutive scripts without having to log in again for each one. This is paramount to running scripts on a head-less (one without a graphic interface) server where logging in may not even be possible. It may be a good idea to change some parameters to apply for the current case, such as the password of the principal.

After running the script the outcome **must** be noted, especially the "Values for Login" part. It is then used by the *Azure.Rm.Login.ps1* script. The output of the service provider script will look like the content in Figure 1.

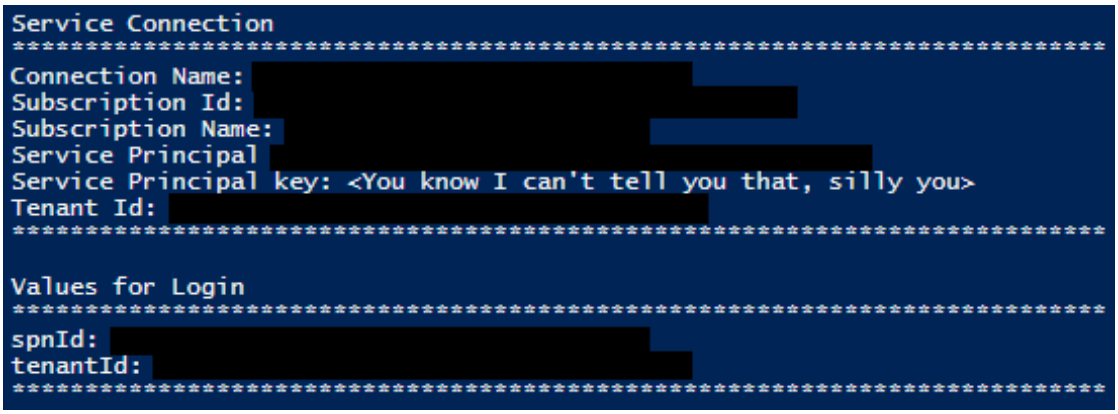

Figure 1: Service Principal script output

3.2 Login

After creating a service principal *Azure.Rm.Login.ps1* is opened and the *$spnId* and *$tenantId* should be replaced with the values from the service principal script. Save the newly updated script and it is ready to be run.

3.3 Provision All

Apart from the log in scripts, *Provision.All.ps1* may be the only script that ever needs to be run to provision the Azure cloud. Provision All is a collection of every script used to provision all services needed to create the system. It also ties some of them together, by extracting connection keys, etc. Note when creating two instances of the system on the same Azure account, or to get different URLs and names for the services, some variables may need to be changed in the configuration section.

3.4 Delete All

The *Delete.All.ps1* script can be used to tear down the system. In the same way as the Provision All script is a collection of scripts to create the system, Delete All is a collection of scripts to take it back down. This script may be useful when setting up a scheduled setup and teardown of the system, where the entire system is rebuilt or taken down when some condition is met.

3.5 Individual Scripts

Every other provision and deletion script in the AutomationScripts directory is written in such a way that they can be executed individually of the provision/delete all scripts. However, some restrictions apply and to name a few:

- No provision script will run unless the resource group exists (which can be created using *Resource.Group.Provision.ps1*).
- A service can not be deleted, using a delete script, unless it exists.
- Some scripts, like *StreamAnalytics.Add.Input.ps1*, only run if the service exists. All scripts with *Provision* in the name should be safe.

It is not recommended to run individual scripts, except in special cases. Instead Provision All should be used, as it runs the scripts in the ideal order, as well as running some helpful utility scripts in between.

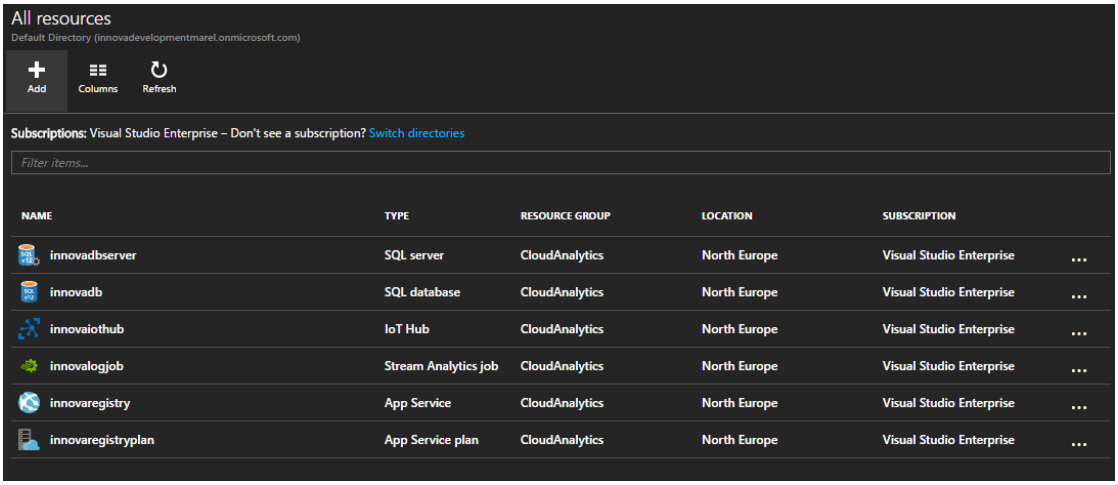

Figure 2: The All Resources screen in the Azure portal

3.6 After Running the Scripts

After running all the necessary scripts (Service Provider, Login and Provision All in most cases), it is a good idea to check if all the services were properly set up on the Azure portal website (*https://portal.azure.com/*). The *All Resources* page should look like the content in Figure 2. If any parameters or variables were changed, just make sure that all connections remain intact.

Each service can be clicked for additional configuring or to observe that everything is working correctly. For instance, it may be wise to check if all *StreamAnalytics* connections are correct. At the time of writing, StreamAnalytics has one input, IoT Hub, and three outputs to different tables in the same database. However, if all scripts ran successfully, there should not be any issues.

In Addition to ensuring that logs are able to travel through the system, it is recommended that the Node.js application in End2EndTests directory is executed using the command *node app.js*.

4 Web Application Deployment

A major component of the Cloud Analytics system is the web application. The best (and only properly tested) way to deploy it is to use a dedicated build server that is set up using continuous integration and deployment methods. A build is then defined in *Microsoft Visual Studio Team Services* and set to queue a new build on code change.

Figure 3 shows the recommended build definition setup. The definition is based on the deployment template found when the + symbol is clicked to create a new definition, with added PowerShell scripts to test and build front-end code.

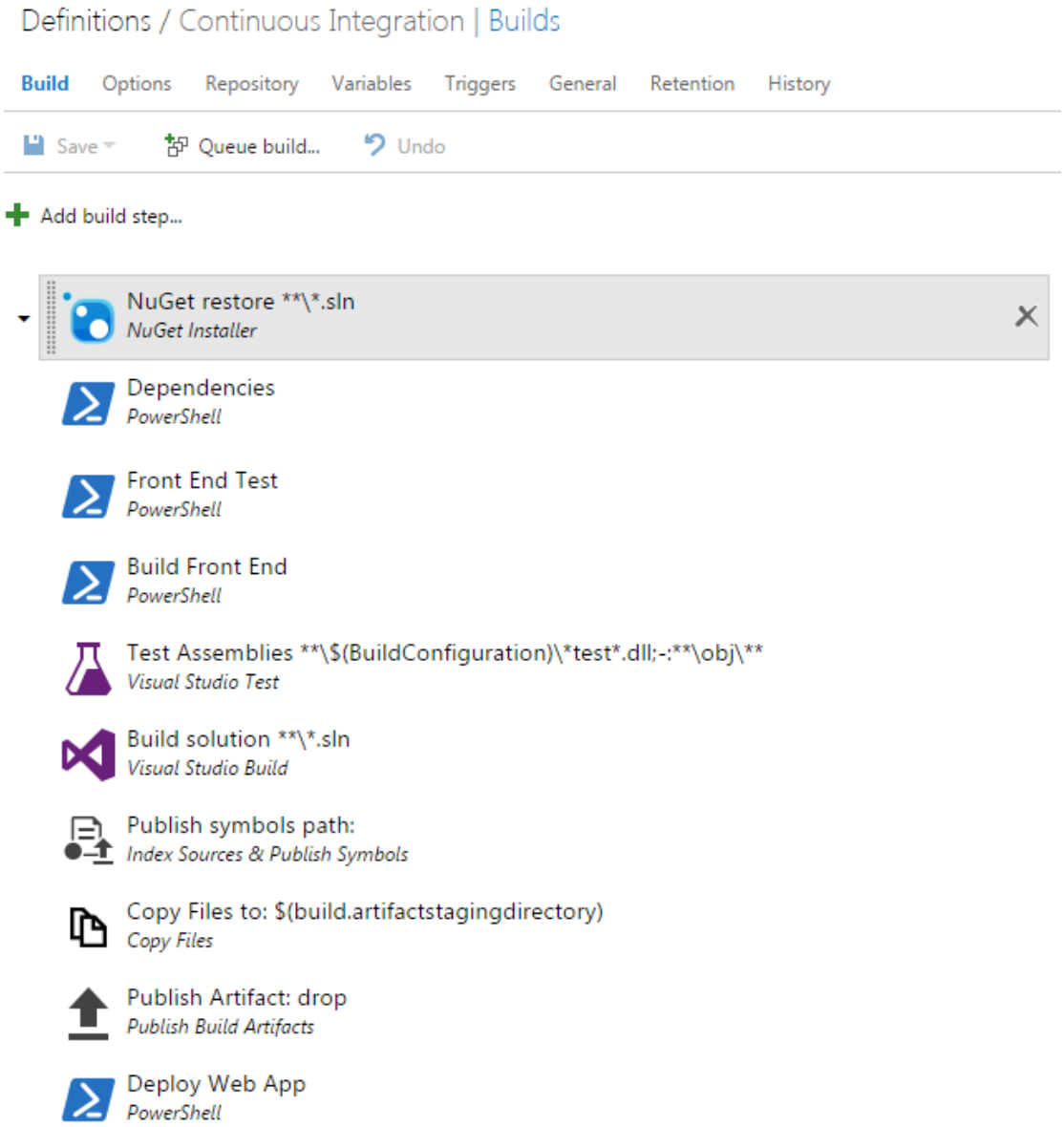

Figure 3: Preferred continuous integration build definition

4.1 One-time PowerShell Install Script

As explained in the *Tools and Programs* section of this document, npm must be installed on the build server/development computer. If npm (and Node.js) is correctly installed, the script located at *Misc/Npm.Install.Globally.ps1* can be run to install all required applications globally on the build server, before building the front-end of the web application. **Note:** this should be done **before** attempting to run the scripts below.

4.2 PowerShell Deployment Scripts

All the PowerShell scripts seen in the build definition should be added in the correct order as depicted in Figure 3. All these scripts are located in the Scripts directory with the Azure provisioning scripts under these name:

1. Dependencies: *Web App/Web.App.Dependencies.ps1*
2. Front End Test: *Web App/Web.App.Test.ps1*
3. Build Front End: *Web App/Web.App.Build.ps1*
4. Deploy Web App: *Deployment/Publish.Package.ps1*

Scripts 1-3 are used to build the front-end of the web application. The automation tool *Gulp* is used to run the builds and *Karma* to run the tests. What these tools are and how they are used is described in more detail in the *Development Manual*. The last script is only a workaround solution to avoid some known issues when publishing the web application.