HOW TO LIFT-AND-SHIFT A LINE OF BUSINESS APPLICATION ONTO GOOGLE CLOUD PLATFORM

GitHub Source Url:
https://github.com/Magenic/GCELiftAndShift

March, 2017
Scenario Description

Magenicons, a fictional comic book publishing company, has decided that their IT infrastructure needs to go through a modernization effort in order to increase system reliability while providing cost savings. Their IT staff believes leveraging cloud computing will help the company gain better agility for their IT infrastructure and applications. As part of an evaluation process, the company has selected an existing intranet-based expense reporting application as the proof-of-concept for their cloud migration strategy.

Background Information

The expense reporting application is a standard two tier web-based application that currently relies on an on-premise Internet Information Server (IIS) server with data storage on a separate on-premise SQL Server. A second on-premise IIS server is also utilized to provide auditing services. Access to the application is secured by authenticating against an on-premises Active Directory (AD) while data access is secured by using SQL Server Authentication with an application service account.

Project Approach

In order to minimize risk while gradually ramping up their teams' cloud knowledge and experience, Magenicons wants to execute the project in phases, with each phase having a defined objective to achieve. These objectives will be used at the end of the project for evaluation of the long-term viability of Cloud Computing for the company. Google Cloud Platform (GCP) was selected as the cloud provider due to the robust capabilities of the platform and Google's excellent technical reputation.

• **Phase one** – Migrate the application to the cloud using Google's Infrastructure-as-a-Service (IaaS) offering: Google Compute Engine (GCE).
  - **Objective:** Be able to lift-and-shift the expense reporting application by leveraging GCE with minimal cost. In particular, the company would like to execute the move with minimal to no code change to the existing application.
  - **Prerequisite:** Environment setup, such as establishing a network connection between Magenicons' local network and GCP, will be required in this phase to support the lift-and-shift of the application.

• **Phase two** – Leverage the cloud for high availability (HA).
  - **Objective:** Once the application is properly operating in the cloud, Magenicons would like to reduce the risk of potential downtimes by adding HA to SQL Server used by the application. AlwaysOn Availability Groups is SQL Server’s recommended solution, allowing users to configure replicas for automatic failover in case of failure. GCP supports Windows Server Failover Clustering (WSFC) and SQL Server AlwaysOn Availability Groups.
• **Phase three** – Leverage the cloud for disaster recovery (DR)
  - **Objective:** Magenicons would like to then further enhance application availability and improve their DR plan by extending their on-premise AD into the cloud. This provides a cost-effective option for protecting AD in DR scenarios. In the event of a physical disaster or outage at company’s data center, a virtual machine (VM) running as an Active Directory Domain Controller (AD DC) in the GCP can provide uninterrupted access to AD for cloud-based applications and any on-premise AD-integrated applications unaffected by the outage. As an added benefit, having an AD hosted in cloud alongside the application will significantly shorten the network latency and thus improve system response time.

**Current On-Premise System Architecture**
The expense report system uses a standard ASP.NET MVC application architecture for an intranet environment. The application is deployed onto an IIS webserver hosted in Windows Server and joined to the AD domain. The system is secured by leveraging Windows Integrated Security for all access to the application. Connection to SQL Server is also quite standard by using SQL Server Authentication with a domain service account user id and password.
Final Targeted On-Cloud System Architecture
The final targeted system architecture should look similar to the original on-premise system architecture, as it is treating GCP as an extension to the on-premise data center via a virtual private network (VPN) with additional features for SQL Server HA and DR for AD.

The How-To’s

Phase One Implementation
For the phase one objective of lifting-and-shifting the expense reporting application to the cloud, four major tasks were identified as requirements:
1. Create a GCP network suitable for the project
2. Create a VPN from Magenicons corporate network to Google Cloud Platform
3. Create the VM instances that are necessary to support the application
4. Make any necessary configuration and or code changes to support the lift-and-shift

Creating a GCP network
GCP networks connect VM instances to each other and to the Internet. It allows users to segment their networks, create firewall rules for access control, as well as creating static routes to forward traffic to specific destinations. All of these capabilities will be needed as the project move along its various phases. Tutorial on the particulars of GCP networking can be found here.

Important Note: Any type of supported subnet network mode (auto or custom) can be used to achieve phase one’s objectives. However, as detailed in phase two below, in order to install SQL Server AlwaysOn Availability Group a custom subnet must be used. Therefore, if one has the desire to eventually install this feature, it is highly recommended that a custom subnet be created for the project from the beginning to avoid any unnecessary rework down the road.
Creating the Site-to-Site VPN
Creating the VPN was a straightforward exercise and the project team did not run into any issues of note. They simply followed the documentation from Google and the VPN was up and running within a day.

Creating the VMs to Support the On-Cloud System Architecture
Creating VM instances in GCE can be done in 3 ways:
1. The point-and-click interface: Google Cloud Console in the subscriber’s portal
2. REST API
3. Command Line Interface (CLI), which in Google’s case is called gcloud

Since the team understands that automation is a key ingredient to long term sustainability for cloud computing, it was decided that a code based approach using the gcloud CLI would be the preferred choice.

Using the CLI is quite simple, just download the gcloud SDK from Google and follow the documentation. Below is a screenshot of the experience when creating a Windows Server Instance:

Below are the VM instances and their roles needed for the various phases of the project:
• Phase 1
  - IIS Web Server
  - SQL Server
• Phase 2
  - Additional SQL Server instance used as part of SQL Server Availability Group replica
• Phase 3
  - An AD Domain Controller instance running in GCP with full replication to the on-premise instance of AD Domain Controller
In addition to the above mentioned VM creation methods, GCP offers another great time saving alternative for creating VM instance: Cloud Launcher. Google Cloud Launcher is a marketplace for ready-to-go development stacks, solutions and services. If the workload type fits what’s available, then one can create the needed VM with simply a few clicks.

For Phase 1 development, the staff needed an IIS Server with ASP.NET 4.6 and its supporting .NET Framework installed. With the usual VM creation method, one would have to create the OS VM and manually install all the various .NET Framework & ASP.NET components. With Cloud Launcher for ASP.NET (currently in Beta), however, the process is dramatically simplified:

1. Go to the following url: [https://console.cloud.google.com/launcher](https://console.cloud.google.com/launcher)
2. Type in asp.net in the search input, and click on search result to initiate the Cloud Launcher process for ASP.NET Framework
3. Fill in the deployment specifics such as machine name, disk and cpu size:

4. Within minutes, the desired VM, along with all the needed components, will be created:
For the creation of the SQL Server instance, the team uses the following `gcloud` CLI to create the image:

```
gcloud compute instances create "magcustom-sql1" --machine-type "n1-standard-4" --zone "us-central1-a" --subnet "wsfcsubnet1" --image-project windows-sql-cloud --image-family sql-ent-2016-win-2016 --boot-disk-size "200" --boot-disk-type "pd-ssd" --private-network-ip=10.201.1.3 --can-ip-forward
```

**Note:** During phase 2 of the project, when the team is going to setup SQL Server HA, specific network routes will need to be created (see section on phase 2 for details). Therefore, this VM instance's internal IP Address will need to conform to the network scheme described in phase 2. This is reason the 'private-network-ip' parameter was used to specify a preferred internal IP address at instance creation time. If this parameter is used to specify a specific IP address, then one can not change it to different static IP address afterwards without the risk of losing access to the instance.

**Code Deployment**

Once the IIS VM instance is provisioned and properly set up, the next task for the team is to deploy the code onto the instance. This is accomplished by leveraging one of Microsoft's many offerings in this space: Web Deploy.

Web Deploy is a mature, extensible client-server tool for publishing website content between a developer’s or SysOps’ workspace onto an IIS instance. The actual mechanism is well documented in the ASP.NET community and is out of scope for this document, but an overview of this technology can be found [here](#).

Security best practices call for always encrypting sections of a configuration file to protect sensitive information, e.g. database connection strings. This improves security by making it difficult for unauthorized access even if an attacker gains access to your configuration file. The same principle applies to this application.

The .NET Framework includes two protected built in configuration providers that can be used to encrypt sections of a configuration file. The RsaProtectedConfigurationProvider class uses the RSACryptoServiceProvider to encrypt configuration sections. The DpapiProtectedConfigurationProvider class uses the Windows Data Protection API (DPAPI) to encrypt configuration sections. However, given the expense reporting application's required usage of integrated security and impersonation, RSACryptoServiceProvider is not a suitable choice as it would require granting access to the RSA Key Container used for encryption to a large group of users.
Encryption of Sensitive Data as part of Deployment Process

One of the major downsides of using the DpapiProtectedConfigurationProvider is the fact it’s not the default Configuration Provider used by Web Deploy, and therefore it is not able to automatically encrypt sensitive data as part of the deployment process. After a bit of research, the Magenicons development staff comes up with a solution that will be able to build, deploy and encrypt sensitive data (on the deployed server) with one single call to MSBuild. The solution calls for leveraging PowerShell’s Invoke-command cmdlet, which has the ability to run commands on local or remote computers. Combining this capability with MSBuild’s extensible feature of embedding scripts for various build and deployment events (in this case, after ‘MSDeployPublish’), the team is able to optimize the build/deployment process while enhancing the security of the application.

Creating the SQL Server AlwaysOn Availability Groups

Enterprise SQL Server workloads require support for HA and DR. AlwaysOn Availability Groups is SQL Server’s flagship HA/DR solution. This technology provides hot-standby for the servers and duplicate data for the database. AlwaysOn can also provide read-only access to one or more secondary replicas, alleviating load from the primary database in reporting and other read-only scenario.

For these reasons, Magenicons’ IT staff selects this technology for achieving the project’s HA requirement. Coincidentally, Google has recently added the support for SQL Server AlwaysOn Availability Group on GCE (it’s currently in beta with GA expected by end of Q2 2017).

In planning for the installation for AlwaysOn Availability Groups, there are several requirements one needs to pay special attention to.

1. At the current time, AlwaysOn Availability Groups can only be installed and supported in a GCP subnet network type. It cannot be installed in a legacy network, moreover, the subnet network must be in custom mode and not the default auto mode (details on the difference in these network types and subnet modes can be found here).
2. Each node in the AlwaysOn Availability Group must reside on a different subnetwork, therefore one would need a minimum of 2 subnetworks for the setup.
3. Each database replica is hosted by an instance of SQL Server on a different node of the Windows Server Failover Cluster (WSFC) cluster.
4. To implement a 2 nodes failover cluster, 4 IP addresses must be provisioned for the cluster itself as well as the Availability Group Listener. It’s important to note these designated IP Addresses must fall outside of the actual subnetwork IP Address range of the cluster nodes, but still be addressable with an appropriate subnet mask.
Let’s walk through a quick example for clarification:

If a subnetwork is defined as 10.0.1.0/24, the VM’s static IP and subnetmask are setup as 10.0.1.4 and 255.255.0.0 (/16). From the VM’s perspective, the addressable subnet is 10.0.0.0/16. Therefore, one should pick an IP address such as 10.0.2.4 for the listener, which is outside 10.0.1.0/24 subnetwork the VM resides in, but still addressable from the guest OS’s perspective due to its wider subnet mask. One needs to apply this requirement for all the IP Addresses needed for WSFC and Availability Group Listener purpose.

See table below for a sample network address scheme needed for entire setup. A step by step tutorial is also available here.

Example Network address Scheme for the AlwaysOn Installation:

<table>
<thead>
<tr>
<th>SUBNETWORKS</th>
<th>IP ADDRESSES RANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsfcsubnet1</td>
<td>10.201.32/29</td>
</tr>
<tr>
<td>wsfcsubnet2</td>
<td>10.202.32/29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NODE INSTANCES</th>
<th>SUBNETWORKS</th>
<th>WSFC</th>
<th>AVAILABILITY GROUP LISTENER</th>
</tr>
</thead>
<tbody>
<tr>
<td>magcustom-sql1</td>
<td>10.201.34</td>
<td>10.201.1.50</td>
<td>10.201.1.51</td>
</tr>
<tr>
<td>magcustom-sql2</td>
<td>10.202.34</td>
<td>magcustom-wsfc</td>
<td>magcustom-as</td>
</tr>
</tbody>
</table>

5. Lastly, as noted in the step by step tutorial, network routes for the cluster and the availability group listener are needed in order for the listener and cluster to be able to reach the node instances. To create the routes, simply follow the example commands provided in the tutorial (modify them to fit one’s own networking scheme as needed).

Phase Three

**Setting up the AD Replication for Redundancy in the Cloud**

As an underlying goal for the entire project, the network in GCP should be treated as an extension of the Magenicons on-premises network so that applications can move from one site to the other seamlessly. Once the VPN connectivity between the two sites is setup, one can create an AD DC and Domain Name Server (DNS) in GCP as if it’s just another branch office. Once this is set up and running, application(s) running in GCP will not have to traverse the internet for authentication and lookup purposes thus improving system performance.
The following diagram depicts the traffic flow:

![Diagram](image)

The process for setting up an AD DC with GCE is similar to other VM roles. The first task is to provision a VM instance by running a gcloud command:

```bash
  gcloud compute instances create your-dc-machine-name --machine-type n1-standard-1 \
  --boot-disk-type pd-ssd --image-project windows-cloud \
  --image-family windows-2016 --boot-disk-size 200GB \
  --zone us-central1-a --subnet wsfcsubnet3 --private-network-ip=10.2.0.100
```
Once the VM is provisioned, the following tasks are performed in order to set up the site to site or inter-Site Domain replication:

**On the on-premise DC**
- Create a new site for GCP:

- Add a GCP subnet in Active Directory Sites and Services:
On the GCP AD DC

- Change the default DNS IP to point to the existing DC on-prem:

- Join the VM to the on-premise domain:
Computer Name/Domain Changes
Enter the name and password of an account with permission to join the domain.

- **Domain**: magenicons.net

Connect a smart card

Welcome to the magenicons.net domain.

OK
• Install Active Directory

![Active Directory Installation](image)

[Add Roles and Features Wizard](image)

Select server roles

Choose "Active Directory Domain Services" to facilitate secure user access.

Description:

Active Directory Domain Services (AD DS) stores information about objects on the network and makes this information available to users and network administrators. AD DS uses domain controllers to give network users access to permitted resources anywhere on the network through a single logon process.
• Promote the VM to a domain controller
• After it reboots, configure DNS settings so that the VM is pointing to itself for DNS queries

For a detailed explanation of how AD Replication works in various network topology (including the one used here), one can reference this documentation.
Final Thoughts and Conclusion
At the completion of the project, all 3 phases were successfully delivered while achieving their respective objectives. Magenicons IT staff found their experience with the cloud to be both an intuitive and efficient one. The GCP portal’s simplicity was a joy to use. Documentation on the topics needed to carry out the various tasks was plentiful on Google’s site and support (available in both paid and free format) was easy to use. Other than some technical requirements needed for SQL Server AlwaysOn that needed a bit of experimentation and time to digest and implement, the staff did not run into anything that would hindered their project. Most impressive of all, other than changing the application connection string in the web.config (to connect to the new HA SQL Server instance), not a single line of code needed to be changed in order to make the system properly hosted in the cloud!

About Magenic
Magenic is a leader in business technology consulting. We understand the barriers to innovation companies are facing and apply the right technology to transform their business.

Visit us at magenic.com or call us at 877.277.1044 to learn more or to engage Magenic today.