Devleap
BridgetheGap

Deploying a SharePoint 2013 Farm Lab on Windows Azure VMs

Revision 4.0 – Update on 1 July 2013

Provisioning the VMs for Servers

In order to deploy a Microsoft SharePoint 2013 Farm Lab on a Windows Azure VMs (IaaS) environment, first of all you need to go to www.windowsazure.com and access the Portal using a valid LiveID account. Then, it is suggested to create a dedicated subscription to manage the Lab farm.

After having created the subscription, or using an already existing one, you need to follow the instructions illustrated on MSDN (http://msdn.microsoft.com/en-us/library/windowsazure/jj554332.aspx) in order to configure your environment for running PowerShell scripts and cmdlets against the Windows Azure environment. For instance, you will have to invoke the Get-AzurePublishSettingsFile cmdlet, in order to retrieve a .publishsettings file useful to manager the remote Azure environment.

Note: In the following PowerShell scripts and code excerpts you should replace the code and values highlighted in red color with values of your own.

As soon as you will have configured your workstation for running PowerShell scripts and cmdlets you should execute the following PowerShell script:

```
##############################
# Initial Azure Configuration #
##############################

Import-Module "C:\Program Files (x86)\Microsoft SDKs\Windows Azure\PowerShell\Azure\Azure.psd1"
Set-ExecutionPolicy RemoteSigned

Import-AzurePublishSettingsFile "D:\Azure\Your-credentials.publishsettings"

$AzureSubscriptionName = "The name of your subscription"
Select-AzureSubscription $AzureSubscriptionName
```

As you can see you simply configure the PowerShell environment in order to consume Windows Azure extensions, as well as to reference the subscription you want to work on.

Just after having configured the target subscription, you can start configuring and creating your environment. In the following code excerpt, you can see how to configure the network and the affinity groups where you will create the farm.

```
########################################
# Create the Azure Affinity Group & VNet #
########################################

# Execute the cmdlet "Get-AzureLocation | select Name" to
# retrieve all the available Azure locations

# Define some support variables for creating affinity group
$AFGLocation = "North Europe"
$AFGDescription = "SP2013-AG"
$AFGName = "SP2013-AG"
```

```
$AFGLabel = "SP2013-AG"

# Create the affinity group
New-AzureAffinityGroup -Location $AFGLocation -Description $AFGDescription -Name $AFGName -Label
$AFGLabel

# Load the network configuration from an external XML file and set it
$ConfigPath = "D:\Azure\SP2013-NetworkConfig.xml"
Set-AzureVNetConfig -ConfigurationPath $ConfigPath

# Executing the following cmdlet you can retrieve the XML configuration
# of an already existing network
#Get-AzureVNetConfig | select -ExpandProperty XMLConfiguration

# Define some support variables for referencing the target environment
$VNetName = "SP2013-Farm"
$DCSubnetName = "DC-Subnet"
$SPSubnetName = "SP-Subnet"
$SQLSubnetName = "SQL-Subnet"

# Define the names of the availability sets
$AvSetNameDC = "SP2013-DC-AV"
$AvSetNameSP = "SP2013-SP-AV"
$AvSetNameSQL = "SP2013-SQL-AV"
```

As you can argue from the previous code sample, the network is configured starting from an XML file, which can be like the following one.

```
<NetworkConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">
  <VirtualNetworkConfiguration>
    <Dns>
      <DnsServers>
        <DnsServer name="sp2013-dc1-ns" IPAddress="10.50.10.4" />
      </DnsServers>
    </Dns>
    <VirtualNetworkSites>
      <VirtualNetworkSite name="SP2013-Farm" AffinityGroup="SP2013-AG">
        <AddressSpace>
          <AddressPrefix>10.50.0.0/16</AddressPrefix>
        </AddressSpace>
        <Subnets>
          <Subnet name="DC-Subnet">
            <AddressPrefix>10.50.10.0/24</AddressPrefix>
          </Subnet>
          <Subnet name="SP-Subnet">
            <AddressPrefix>10.50.20.0/24</AddressPrefix>
          </Subnet>
          <Subnet name="SQL-Subnet">
            <AddressPrefix>10.50.30.0/24</AddressPrefix>
          </Subnet>
        </Subnets>
        <DnsServersRef>
          <DnsServerRef name="sp2013-dc1-ns" />
        </DnsServersRef>
      </VirtualNetworkSite>
    </VirtualNetworkSites>
  </VirtualNetworkConfiguration>
</NetworkConfiguration>
```

The network configuration file, in my example, defines that we have a DNS server named *sp2013-dc1-ns* and with an IP address of 10.50.10.4. Then, it states that the network name is SP2013-Farm and it belongs

to the affinity group named *SP2013-AG*. The global IP address space for the network – in the sample scripts - is 10.50.0.0/16, and there are three sub-networks:

- DC-Subnet: 10.50.10.0/24
- SP-Subnet: 10.50.20.0/24
- SQL-Subnet: 10.50.30.0/24

As you can argue from their names, these sub-networks define the IP subnet for domain controllers, SharePoint servers, and SQL servers.

Now you have to create a storage account, in case you do not have one, for storing VHDs of your VMs. Here you can see how to do that.

```
###################################
# Create the Azure Storage Account #
###################################

$StorageAccountName = "sp2013farmstorage"
$StorageAccountLabel = "SP2013 Storage Account"
New-AzureStorageAccount -StorageAccountName $StorageAccountName -Label $StorageAccountLabel
  -AffinityGroup $AFGName

$StorageUriBase = "https://" + $StorageAccountName + ".blob.core.windows.net/vhds/"

Set-AzureSubscription -SubscriptionName $AzureSubscriptionName
  -CurrentStorageAccount $StorageAccountName
```

Now you are ready to create the first VM in your farm, which should be the Domain Controller of the domain that will support you Lab Farm.

```
#################
# Create the DC #
#################

$Password = "Pass@w0rd!"
$DomainUsername = "MyAdministrator"
$DomainPassword = "Pass@w0rd!"

# In order to retrieve the list of available VMs models invoke the
# following command
#Get-AzureVMImage | Select ImageName

# Configure the image file for Windows Server 2012
$WindowsServer2012 = "a699494373c04fc0bc8f2bb1389d6106__Windows-Server-2012-Datacenter-201306.01-
en.us-127GB.vhd"

# Configure the image file for Windows Server 2012 + SharePoint 2013 Eval
$SharePoint2013 = "c6e0f177abd8496e934234bd27f46c5d__SharePoint-2013-Trial-4-13-2013"

# Configure the image file for Windows Server 2008 R2 + SQL Server 2012 Standard
$WindowsServer2008R2SP1WithSQL2012 = "fb83b3509582419d99629ce476bcb5c8__Microsoft-SQL-Server-
2012SP1-Standard-CY13SU04-SQL11-SP1-CU3-11.0.3350.0-B"

$DCServiceName = "SP2013-PaoloPi-DC"
$DCServiceLabel = "SP2013-PaoloPi-DC"
$DCServiceDescription = "SP2013 Domain Controllers"

$DC1Name = "SP2013DC1"
$DC1Label = "SP2013DC1"
$DC1Size = "Small"
$DC1DataDiskSize = "15"
$DC1DataDiskLabel = "SP2013DC1DataDisk"
```

```
$DC1DataDiskName = "SP2013DC1 Data Disk"
$DC1SysDiskMediaLocation = $StorageUriBase + "sp2013dc1systemdisk01.vhd"
$DC1DataDiskMediaLocation = $StorageUriBase + "sp2013dc1datadisk01.vhd"
$DC1 = New-AzureVMConfig -Name $DC1Name -AvailabilitySetName $AvSetNameDC -ImageName
$WindowsServer2012 -InstanceSize $DC1Size -Label $DC1Label -MediaLocation $DC1SysDiskMediaLocation |
        Add-AzureProvisioningConfig -Windows -Password $Password -AdminUsername $DomainUsername |
        Add-AzureDataDisk -CreateNew -DiskSizeInGB $DC1DataDiskSize -DiskLabel $DC1DataDiskLabel -
LUN 0 -MediaLocation $DC1DataDiskMediaLocation |
        Set-AzureSubnet $DCSubnetName

New-AzureVM -ServiceName $DCServiceName -ServiceLabel $DCServiceLabel -ServiceDescription
$DCServiceDescription -AffinityGroup $AFGName -VNetName $VNetName -VMs $DC1

$DNS1 = New-AzureDns -Name 'sp2013-dc1-ns' -IPAddress '10.50.10.4'
$DomainName = "sp2013.test"
```

As you can see the script defines the user credentials to use for initial configuration of the domain controller VM. Moreover, it defines a “Small” sized VM with a couple of virtual disks, one for the operating system and another one for the data, which will host the domain data files. Using the *New-AzureVMConfig* cmdlet the scripts defines the configuration for the VM, its availability set, as well as the target subnet. Then using the *New-AzureVM* cmdlet it creates a new cloud service that will host the just configured VM inside the target network and affinity group. Lastly, it defines the DNS server for the farm, using the *New-AzureDns* cmdlet.

Now, as soon as the domain controller VM will be provisioned and running, you will have to create an Active Directory domain, either using PowerShell or the graphical user interface provided by Windows. The current example creates a Windows Server 2012 domain controller and uses the new server management UI for installing the domain controller feature for a domain named *sp2013.test*.

Note: consider that you can access the VMs via PowerShell from remote sites. Thus, a suitable option is to install the AD forest, as well as all the other stuff from a remote management station.

Just after having provisioned and configured the domain controller, you can define an Organizational Unit to hold all the SharePoint related active directory objects. In the current sample Lab Farm it is called *SharePoint Farm*, and it is defined also within a PowerShell variable. Here you can see the variable declaration.

```
$SPDomainOU = "OU=SharePoint Farm,DC=sp2013,DC=test"
```

Now you are ready to create the other servers of the farm. Here you can see how to provision the SharePoint servers.

```
#########################
# Create the SP Servers #
#########################

# SP Servers Service
$SPServersName = "SP2013-PaoloPi-SP"
$SPServersLabel = "SP2013-PaoloPi-SP"
$SPServersDescription = "SP2013 SharePoint Servers"
$SPServers = @()

# Change the following array to change the number of SP servers
# to provision
$SPServersArray = (1,2)
Foreach ($SPServer in $SPServersArray) {
    # SP2013SPSRVn
```

```
    $SPAPPnName = "SP2013SRV" + $SPServer
    $SPAPPnLabel = "SP2013SRV" + $SPServer
    $SPAPPnSize = "Medium"
    $SPAPPnSysDiskMediaLocation = $StorageUriBase + "sp2013srv" + $SPServer + "systemdisk01.vhd"
    $SPAPPn = New-AzureVMConfig -Name $SPAPPnName -AvailabilitySetName $AvSetNameSP -ImageName
$SharePoint2013 -InstanceSize $SPAPPnSize -Label $SPAPPnLabel -MediaLocation
$SPAPPnSysDiskMediaLocation |
    Add-AzureProvisioningConfig –DisableWinRMHttps -WindowsDomain -Password $Password
    -AdminUsername $DomainUsername -Domain $DomainName -DomainUserName $DomainUsername
    -DomainPassword $DomainPassword -JoinDomain $DomainName -MachineObjectOU $SPDomainOU |
    Set-AzureSubnet $SPSubnetName
    $SPServers += $SPAPPn
}

New-AzureVM -ServiceName $SPServersName -ServiceLabel $SPServersLabel -ServiceDescription
$SPServersDescription -AffinityGroup $AFGName -VNetName $VNetName –DnsSettings $DNS1 -VMs $SPServers
```

In the previous code excerpt, you can see the definition of an array of items, which are the servers to provision. The example provisions two servers based on Windows Server 2012 + SharePoint 2013 Eval (Expires on 16 Dec 2013). Later you will configure SharePoint 2013 on them. Notice the command highlighted in bold, which provisions the VMs inside the target Active Directory domain, within the Organization Unit we declared early. Notice also the *New-AzureVM* cmdlet invocation, passing the array of server as the input for argument named VMs. This command will provision all the servers within the same cloud service.

The last minimal step to execute is installing the SQL Server environment. The PowerShell code for this task is almost the same as the previous one. The only significant difference is that you need to reference a different virtual machine model, in order to provision a Windows Server 2008 R2 machine with SQL Server 2012 on-board.

```
#########################
# Create the SQL Server #
#########################

$WindowsServer2008R2SP1WithSQL2012 = "MSFT__Sql-Server-11EVAL-11.0.2215.0-08022012-en-us-30GB.vhd"

# SQL Servers Service
$SQLServersName = "SP2013-PaoloPi-SQL"
$SQLServersLabel = "SP2013-PaoloPi-SQL"
$SQLServersDescription = "SP2013 SQL Servers"
$SQLServers = @()

$SQLDiskSize = 100
$SQLDataDiskLabel = "DataDisk"
$SQLDataDiskName = "Data Disk"
$SQLLogDiskLabel = "LogDisk"
$SQLLogDiskName = "Log Disk"
$SQLTempDbDiskLabel = "TempDbDisk"
$SQLTempDbDiskName = "TempDb Disk"

$SQLServersArray = (1)
Foreach ($SQLServer in $SQLServersArray) {
    # SQLn
    $SQLnDataDiskMediaLocation = $StorageUriBase + "sp2013sql" + $SQLServer + "datadisk01.vhd"
    $SQLnLogDiskMediaLocation = $StorageUriBase + "sp2013sql" + $SQLServer + "logdisk01.vhd"
    $SQLnTempDbDiskMediaLocation = $StorageUriBase + "sp2013sql" + $SQLServer + "tempdbdisk01.vhd"

    $SQLnName = "SP2013SQL" + $SQLServer
    $SQLnLabel = "SP2013SQL" + $SQLServer
    $SQLnSize = "Large"
    $SQLnSysDiskMediaLocation = "https://" + $StorageAccountName +
".blob.core.windows.net/vhds/sp2013sql" + $SQLServer + "systemdisk01.vhd"
```

```
    $SQLnDataDiskLabel = $SQLnLabel + $SQLDataDiskLabel
    $SQLnLogDiskLabel = $SQLnLabel + $SQLLogDiskLabel
    $SQLnTempDbDiskLabel = $SQLnLabel + $SQLTempDbDiskLabel
    $SQLn = New-AzureVMConfig -Name $SQLnName -AvailabilitySetName $AvSetNameSQL -ImageName
$WindowsServer2008R2SP1WithSQL2012 -InstanceSize $SQLnSize -Label $SQLnLabel -MediaLocation
$SQLnSysDiskMediaLocation |
        Add-AzureProvisioningConfig -DisableWinRMHttps -WindowsDomain -Password $Password
    -AdminUsername $DomainUsername -Domain $DomainName -DomainUserName $DomainUsername
    -DomainPassword $DomainPassword -JoinDomain $DomainName -MachineObjectOU $SPDomainOU |
        Add-AzureDataDisk -CreateNew -DiskSizeInGB $SQLDiskSize -DiskLabel $SQLnDataDiskLabel -LUN 0
-MediaLocation $SQLnDataDiskMediaLocation |
        Add-AzureDataDisk -CreateNew -DiskSizeInGB $SQLDiskSize -DiskLabel $SQLnLogDiskLabel -LUN 1
-MediaLocation $SQLnLogDiskMediaLocation |
        Add-AzureDataDisk -CreateNew -DiskSizeInGB $SQLDiskSize -DiskLabel $SQLnTempDbDiskLabel -LUN
2 -MediaLocation $SQLnTempDbDiskMediaLocation |
        Set-AzureSubnet $SQLSubnetName

    $SQLServers += $SQLn
}

New-AzureVM -ServiceName $SQLServersName -ServiceLabel $SQLServersLabel -ServiceDescription
$SQLServersDescription -AffinityGroup $AFGName -VNetName $VNetName –DnsSettings $DNS1 -VMs
$SQLServers
```

In the previous code sample, notice also that the SQL machine is created using the “Large” model and a set of three virtual disks: system disk, data disk, and temporary data disk.

Provisioning Service Accounts and Sample Users

In order to correctly configure you environment you should create a set of service accounts, as well as some sample users for playing with User Profile service and some other functionalities.

Here you can see a PowerShell script to automatically create all of this users and service accounts.

```
Import-Module ActiveDirectory

function createUser
{
    param ([string] $ou, [string] $firstName, [string] $lastName, [string] $password, [string]
$emailDomain)

    $userName = $firstName + "." + $lastName
    $fullName = $firstName + " " + $lastName
    $emailAddress = $username + "@" + $emailDomain
    New-ADUser -SamAccountName $userName -Name $fullName -DisplayName $fullName -GivenName
$firstName -Surname $lastName -Path $ou -ChangePasswordAtLogon $false -AccountPassword (ConvertTo-
SecureString -AsPlainText -String $password -Force) -Description $fullName -Enabled $true -
EmailAddress $emailAddress -PasswordNeverExpires $true -UserPrincipalName $emailAddress
}

function createServiceUser
{
    param ([string] $ou, [string] $userName, [string] $password, [string] $emailDomain)

    $emailAddress = $username + "@" + $emailDomain
    New-ADUser -SamAccountName $userName -Name $userName -DisplayName $fullName -Path $ou -
ChangePasswordAtLogon $false -AccountPassword (ConvertTo-SecureString -AsPlainText -String $password
-Force) -Description $userName -Enabled $true -EmailAddress $emailAddress -PasswordNeverExpires
$true -UserPrincipalName $emailAddress
}

$domain = [ADSI] "LDAP://dc=sp2013,dc=test"
```

```
$ouServices = $domain.Create("organizationalUnit", "ou=Services")
$ouServices.SetInfo()

$ouUserProfiles = $domain.Create("organizationalUnit", "ou=SharePoint Users")
$ouUserProfiles.SetInfo()

$ouUserProfilesEmployees = $ouUserProfiles.Create("organizationalUnit", "ou=Employees")
$ouUserProfilesEmployees.SetInfo()

$services = [ADSI] "LDAP://ou=Services,dc=sp2013,dc=test"
$employees = [ADSI] "LDAP://ou=Employees,ou=SharePoint Users,dc=sp2013,dc=test"

$fakePassword = "Passw0rd"
$emailDomain = "sp2013.test"

createServiceUser -ou $services.distinguishedName -userName "SPFarm" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SPService" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SPContent" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SPSearch" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SPSandbox" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SPUPS" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SQLService" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SQLAgent" -password $fakePassword -
emailDomain $emailDomain
createServiceUser -ou $services.distinguishedName -userName "SQLReporting" -password $fakePassword -
emailDomain $emailDomain

$c = 1

do {
    $firstName = "fn" + $c
    $lastName = "ln" + $c
    createUser -ou $employees.distinguishedName -firstName $firstName -lastName $lastName -password
$fakePassword -emailDomain $emailDomain
    $c++
}
while ($c -le 500)
```

As you can see the code creates some Organizational Units:

- *Services*: for service accounts
- *SharePoint Users*: for SharePoint users
- *Employees*: child of SharePoint Users, for sample employees to synchronize with UPS

Moreover, the app creates 500 fake users under the Employees OU, as well the following service account:

- *SPFarm*: SharePoint farm account
- *SPService*: default account for SharePoint services
- *SPContent*: default content account for web applications
- *SPSearch*: SharePoint search account
- *SPSandbox*: Sandboxed code execution account
- *SPUPS*: UPS replication account
- *SQLService*: SQL Server engine account
- *SQLAgent*: SQL Server Agent account

- *SQLReporting*: SQL Server reporting services account

You should also configure SQL Server 2012 services (engine, agent, reporting) in order to use these domain accounts and not those configured by defaults. Moreover, you should also configure the *SPFarm* account in roles securityadmin and dbcreator onto the SQL Server engine.

Provisioning the SharePoint 2013 Farm

Now you are ready to deploy the SharePoint 2013 farm. Because you created the SharePoint VMs using the trial version of the product, you don't need to download the SharePoint bits, but you simply need to configure the farm. To achieve the result you have to setup the product onto the target SharePoint servers. **Do not** execute the setup and configuration wizard. It is better that you install the SharePoint farm using PowerShell, which provides you with a more flexible set of tools and choices.

Before executing the farm configuration, you may like to configure support for Service Connection Points, as like as Bill Baer illustrated here: http://blogs.technet.com/b/wbaer/archive/2010/04/28/service-connection-points-and-governance-with-sharepoint-server-2010.aspx. The blog post is about SharePoint 2010, but it works the same on SharePoint 2013.

Here you can see the PowerShell script to deploy the farm onto the first server (let's say *SP2013SRV1*). Thanks to Roger Cormier (http://gallery.technet.microsoft.com/office/Create-a-New-SharePoint-81f70350) for his sample code on TechNet code gallery.

```
#############################################################
# Configures a SharePoint 2013 farm with custom:
# Configuration Database
# Central Administation Database
# Central Administation web application and site
#############################################################

Add-PSSnapin Microsoft.SharePoint.PowerShell -erroraction SilentlyContinue

## Settings ##
$configDatabaseName = "SP2013_Farm_SharePoint_Config"
$sqlServer = "SP2013SQL1"
$sqlServerAlias = "SP2013SQL"
$caDatabaseName = "SP2013_Farm_Admin_Content"
$caPort = 2222
$caAuthN = "NTLM"
$passphrase = "pass@word1"
$sPassphrase = (ConvertTo-SecureString -String $passphrase -AsPlainText -force)

#######################################
# Create the SQL Alias
#######################################

$x86 = "HKLM:\Software\Microsoft\MSSQLServer\Client\ConnectTo"
$x64 = "HKLM:\Software\Wow6432Node\Microsoft\MSSQLServer\Client\ConnectTo"


if ((test-path -path $x86) -ne $True)
{
     write-host "$x86 doesn't exist"
    New-Item $x86
}

if ((test-path -path $x64) -ne $True)
{
     write-host "$x64 doesn't exist"
    New-Item $x64
```

```
}

$TCPAlias = "DBMSSOCN," + $SQLServer

New-ItemProperty -Path $x86 -Name $sqlServerAlias -PropertyType String -Value $TCPAlias
New-ItemProperty -Path $x64 -Name $sqlServerAlias -PropertyType String -Value $TCPAlias

#######################################
# Create the farm
#######################################
Write-Output "Creating the configuration database $configDatabaseName"

New-SPConfigurationDatabase –DatabaseName $configDatabaseName –DatabaseServer $sqlServerAlias –
AdministrationContentDatabaseName $caDatabaseName –Passphrase $sPassphrase –FarmCredentials (Get-
Credential)

$farm = Get-SPFarm
if (!$farm -or $farm.Status -ne "Online") {
       Write-Output "Farm was not created or is not running"
       exit
}

# Perform the config wizard tasks
Write-Output "Initialize security"
Initialize-SPResourceSecurity

Write-Output "Install services"
Install-SPService

Write-Output "Register features"
Install-SPFeature -AllExistingFeatures

Write-Output "Create the Central Administration site on port $caPort"
New-SPCentralAdministration -Port $caPort -WindowsAuthProvider $caAuthN

Write-Output "Install Help Collections"
Install-SPHelpCollection -All

Write-Output "Install Application Content"
Install-SPApplicationContent

New-ItemProperty HKLM:\System\CurrentControlSet\Control\Lsa -Name "DisableLoopbackCheck" -value "1"
-PropertyType dword

$ServiceConnectionPoint = get-SPTopologyServiceApplication | select URI
Set-SPFarmConfig -ServiceConnectionPointBindingInformation $ServiceConnectionPoint -Confirm:$False
```

Notice the command to add the SharePoint extensions for PowerShell. Then the script simply defines some environmental variables to hold the general configuration of the farm to provision. Then it is configured a SQL alias for the SQL server in the back-end, which is a good habit in a SharePoint Farm deployment. Lastly it is invoked the *New-SPConfigurationDatabase* cmdlet, which corresponds to the first step of the psconfig tool. Moreover, are invoked all the cmdlets to provision all the other stuff (security settings, services, features, SharePoint Central Administration site, help, and application content). The last cmdlet invoked creates a service connection point into the active directory of the current domain, in case you configured support for service connection points, as referenced before in the script excerpt.

As soon as the script will complete execution, you will have a farm ready to go. Now you can add the second SharePoint server, and all the other SharePoint servers, to the farm. In order to do that you will need to invoke the *Connect-SPConfigurationDatabase* cmdlet on every additional server. Here you can see the code excerpt to execute on the additional servers.

```
#############################################################
# Joins a SharePoint 2013 Farm
#############################################################

Add-PSSnapin Microsoft.SharePoint.PowerShell -erroraction SilentlyContinue

## Settings ##
$configDatabaseName = "SP2013_Farm_SharePoint_Config"
$sqlServer = "SP2013SQL1"
$sqlServerAlias = "SP2013SQL"
$passphrase = "pass@word1"
$sPassphrase = (ConvertTo-SecureString -String $passphrase -AsPlainText -force)

########################################
# Create the SQL Alias
########################################

$x86 = "HKLM:\Software\Microsoft\MSSQLServer\Client\ConnectTo"
$x64 = "HKLM:\Software\Wow6432Node\Microsoft\MSSQLServer\Client\ConnectTo"


if ((test-path -path $x86) -ne $True)
{
     write-host "$x86 doesn't exist"
    New-Item $x86
}

if ((test-path -path $x64) -ne $True)
{
     write-host "$x64 doesn't exist"
    New-Item $x64
}

$TCPAlias = "DBMSSOCN," + $SQLServer

New-ItemProperty -Path $x86 -Name $sqlServerAlias -PropertyType String -Value $TCPAlias
New-ItemProperty -Path $x64 -Name $sqlServerAlias -PropertyType String -Value $TCPAlias

########################################
# Connect to the farm
########################################
Write-Output "Connecting to the configuration database $configDatabaseName"

# psconfig -cmd upgrade -inplace b2b -wait -force
Connect-SPConfigurationDatabase -DatabaseServer $sqlServerAlias -DatabaseName $configDatabaseName -
Passphrase $sPassphrase

$farm = Get-SPFarm
if (!$farm -or $farm.Status -ne "Online") {
       Write-Output "Farm was not connected or is not running"
       exit
}

# Perform the config wizard tasks
Write-Output "Initialize security"
Initialize-SPResourceSecurity

Write-Output "Install services"
Install-SPService

Write-Output "Register features"
Install-SPFeature -AllExistingFeatures

New-ItemProperty HKLM:\System\CurrentControlSet\Control\Lsa -Name "DisableLoopbackCheck" -value "1"
-PropertyType dword
```

Now you can check, using the SharePoint Central Administration or PowerShell, the list of servers available in the current Farm.

Provisioning the SharePoint Services

In case you have correctly provisioned the farm, you are now ready to configure services and service applications. You can for example execute the following PowerShell scripts on the main server (*SP2013SRV1*).

Let's start importing PowerShell extensions for SharePoint and declaring some useful variables, as well as creating or referencing the accounts and application pools.

```
################################
## Farm Initial Configuration ##
################################

Add-PSSnapin Microsoft.SharePoint.PowerShell -erroraction SilentlyContinue

## Settings ##
$databaseServerName = "SP2013SQL"
$saAppPoolName = "SharePoint Web Services"
$appPoolUserName = "SP2013\SPService"

# Retrieve or create the services application pool and managed account
$saAppPool = Get-SPServiceApplicationPool -Identity $saAppPoolName -EA 0
if($saAppPool -eq $null)
{
  Write-Host "Creating Service Application Pool..."

  $appPoolAccount = Get-SPManagedAccount -Identity $appPoolUserName -EA 0
  if($appPoolAccount -eq $null)
  {
      Write-Host "Please supply the password for the Service Account..."
      $appPoolCred = Get-Credential $appPoolUserName
      $appPoolAccount = New-SPManagedAccount -Credential $appPoolCred -EA 0
  }

  $appPoolAccount = Get-SPManagedAccount -Identity $appPoolUserName -EA 0

  if($appPoolAccount -eq $null)
  {
    Write-Host "Cannot create or find the managed account $appPoolUserName, please ensure the
account exists."
    Exit -1
  }
  New-SPServiceApplicationPool -Name $saAppPoolName -Account $appPoolAccount -EA 0 > $null
}
```

Then you can start provisioning the Web Analytics and Health Data Collection service, together with the Usage service, and the State service.

```
$usageSAName = "Usage and Health Data Collection Service"
$stateSAName = "State Service"
$stateServiceDatabaseName = "SP2013_Farm_StateDB"

# Configure the web analytics and health data collection service before creating the service
Set-SPUsageService -LoggingEnabled 1 -UsageLogLocation "C:\Program Files\Common Files\Microsoft
Shared\Web Server Extensions\15\LOGS\" -UsageLogMaxSpaceGB 2

# Usage Service
Write-Host "Creating Usage Service and Proxy..."
```

```
$serviceInstance = Get-SPUsageService
New-SPUsageApplication -Name $usageSAName -DatabaseServer $databaseServerName -DatabaseName
"SP2013_Farm_UsageDB" -UsageService $serviceInstance > $null

# State Service
$stateServiceDatabase = New-SPStateServiceDatabase -Name $stateServiceDatabaseName
$stateSA = New-SPStateServiceApplication -Name $stateSAName -Database $stateServiceDatabase
New-SPStateServiceApplicationProxy -ServiceApplication $stateSA -Name "$stateSAName Proxy" -
DefaultProxyGroup
```

Then you can provision the Managed Metadata service application.

```
$metadataSAName = "Managed Metadata Service"

# Managed Metadata Service
Write-Host "Creating Metadata Service and Proxy..."
$mmsApp = New-SPMetadataServiceApplication -Name $metadataSAName -ApplicationPool $saAppPoolName -
DatabaseServer $databaseServerName -DatabaseName "SP2013_Farm_MetadataDB" > $null
New-SPMetadataServiceApplicationProxy -Name "$metadataSAName Proxy" -DefaultProxyGroup -
ServiceApplication $metadataSAName > $null
Get-SPServiceInstance | where-object {$_.TypeName -eq "Managed Metadata Web Service"} | Start-
SPServiceInstance > $null
```

Now it's time to provision the Search service application. The following excerpt provisions the search services on both the SharePoint server (*SP2013SRV1* and *SP2013SRV2*).

```
###########################
# Search Service - START #
###########################
$searchMachines = @("SP2013SRV1", "SP2013SRV2")
$searchAdminComponentMachines = @("SP2013SRV1", "SP2013SRV2")
$searchAnalyticsMachines = @("SP2013SRV1", "SP2013SRV2")
$searchContentProcessingMachines = @("SP2013SRV1", "SP2013SRV2")
$searchQueryMachines = @("SP2013SRV1", "SP2013SRV2")
$searchCrawlerMachines = @("SP2013SRV1", "SP2013SRV2")
$searchIndexMachines = @("SP2013SRV1", "SP2013SRV2")

$searchSAName = "Search Service"
$saAppPoolName = "SharePoint Web Services"
$databaseServerName = "SP2013SQL"
$searchDatabaseName = "SP2013_Farm_Search"
$indexLocation = "C:\SearchIndex"

Write-Host "Creating Search Service and Proxy..."
Write-Host "  Starting Services..."

foreach ($machine in $searchMachines)
{
    Write-Host "    Starting Search Services on $machine"
    Start-SPEnterpriseSearchQueryAndSiteSettingsServiceInstance $machine -ErrorAction
SilentlyContinue
    Start-SPEnterpriseSearchServiceInstance $machine -ErrorAction SilentlyContinue
}

Write-Host "  Creating Search Application..."
$searchApp = Get-SPEnterpriseSearchServiceApplication -Identity $searchSAName -ErrorAction
SilentlyContinue
if (!$searchApp)
{
    $searchApp = New-SPEnterpriseSearchServiceApplication -Name $SearchSAName -ApplicationPool
$saAppPoolName -DatabaseServer $databaseServerName -DatabaseName $searchDatabaseName
}
$searchInstance = Get-SPEnterpriseSearchServiceInstance -Local
```

```
# Define the search topology
Write-Host "  Defining the Search Topology..."
$initialSearchTopology = $searchApp | Get-SPEnterpriseSearchTopology -Active
$newSearchTopology = $searchApp | New-SPEnterpriseSearchTopology

# Create search components
Write-Host "  Creating Admin Component..."
foreach ($machine in $searchAdminComponentMachines)
{
    New-SPEnterpriseSearchAdminComponent -SearchTopology $newSearchTopology -SearchServiceInstance
$machine
}

Write-Host "  Creating Analytics Component..."
foreach ($machine in $searchAnalyticsMachines)
{
    New-SPEnterpriseSearchAnalyticsProcessingComponent -SearchTopology $newSearchTopology -
SearchServiceInstance $machine
}

Write-Host "  Creating Content Processing Component..."
foreach ($machine in $searchContentProcessingMachines)
{
    New-SPEnterpriseSearchContentProcessingComponent -SearchTopology $newSearchTopology -
SearchServiceInstance $machine
}

Write-Host "  Creating Query Processing Component..."
foreach ($machine in $searchQueryMachines)
{
    New-SPEnterpriseSearchQueryProcessingComponent -SearchTopology $newSearchTopology -
SearchServiceInstance $machine
}

Write-Host "  Creating Crawl Component..."
foreach ($machine in $searchCrawlerMachines)
{
    New-SPEnterpriseSearchCrawlComponent -SearchTopology $newSearchTopology -SearchServiceInstance
$machine
}

Write-Host "  Creating Index Component..."
foreach ($machine in $searchIndexMachines)
{
    New-SPEnterpriseSearchIndexComponent -SearchTopology $newSearchTopology -SearchServiceInstance
$machine -RootDirectory $indexLocation
}

Write-Host "  Activating the new topology..."
$newSearchTopology.Activate()

Write-Host "  Creating Search Application Proxy..."
$searchProxy = Get-SPEnterpriseSearchServiceApplicationProxy -Identity "$searchSAName Proxy" -
ErrorAction SilentlyContinue
if (!$searchProxy)
{
    New-SPEnterpriseSearchServiceApplicationProxy -Name "$searchSAName Proxy" -SearchApplication
$searchSAName
}

########################
# Search Service - END #
########################
```

As you can see, the script allows provisioning the Search service topology on multiple machines, using an array of servers. Notice the path for the search index, which should be properly configure, eventually leveraging a dedicated virtual VHD.

In case you want to provision the User Profile Service, you can follow the great instructions provided by Spencer Harbar on his site (www.harbar.net). Here you can find just a quick code excerpt to create the UPS service with a standard configuration.

```
# User Profile Service
Write-Host "Creating User Profile Service and Proxy acting as Farm account..."
$sb = {
       Add-PSSnapin Microsoft.SharePoint.PowerShell -erroraction SilentlyContinue

       Write-Host "Creating User Profile Service and Proxy acting as Farm account..."
       $saAppPoolNameForUPS = "SharePoint Web Services"
       $saAppPoolForUPS = Get-SPServiceApplicationPool -Identity $saAppPoolNameForUPS -EA 0
       $userUPSName = "User Profile Service"
       $databaseServerNameForUPS = "SP2013SQL"

       $userProfileService = New-SPProfileServiceApplication -Name $userUPSName -ApplicationPool
$saAppPoolNameForUPS -ProfileDBServer $databaseServerNameForUPS -ProfileDBName
"SP2013_Farm_ProfileDB" -SocialDBServer $databaseServerNameForUPS -SocialDBName
"SP2013_Farm_SocialDB" -ProfileSyncDBServer $databaseServerNameForUPS -ProfileSyncDBName
"SP2013_Farm_SyncDB"

       New-SPProfileServiceApplicationProxy -Name "$userUPSName Proxy" -ServiceApplication
$userProfileService -DefaultProxyGroup > $null
}

$farmAccount = (Get-SPFarm).DefaultServiceAccount
$farmCredential = Get-Credential $farmAccount.Name
$job = Start-Job -Credential $farmCredential -ScriptBlock $sb | Wait-Job

Get-SPServiceInstance | where-object {$_.TypeName -eq "User Profile Service"} | Start-
SPServiceInstance > $null
```

The configuration of users' profiles synchronization is out of scope for this article, and you can surely reference a more authoritative source, like the Spence's blog.

Now, with SharePoint 2013 you can also configure the new service applications like the App Management service, which requires the configuration of the Subscription Settings service, too.

```
$subSettingstName = "Subscription Settings Service"
$subSettingstDatabaseName = "SP2013_Farm_SubSettingsDB"
$appManagementName = "App Management Service"
$appManagementDatabaseName = "SP2013_Farm_AppManagementDB"

Write-Host "Creating Subscription Settings Service and Proxy..."
$subSvc = New-SPSubscriptionSettingsServiceApplication –ApplicationPool $saAppPoolName –Name
$subSettingstName –DatabaseName $subSettingstDatabaseName
$subSvcProxy = New-SPSubscriptionSettingsServiceApplicationProxy –ServiceApplication $subSvc
Get-SPServiceInstance | where-object {$_.TypeName -eq $subSettingstName} | Start-SPServiceInstance >
$null

Write-Host "Creating App Management Service and Proxy..."
$appManagement = New-SPAppManagementServiceApplication -Name $appManagementName -DatabaseServer
$databaseServerName -DatabaseName $appManagementDatabaseName –ApplicationPool $saAppPoolName
$appManagementProxy = New-SPAppManagementServiceApplicationProxy -ServiceApplication $appManagement
-Name "$appManagementName Proxy"
Get-SPServiceInstance | where-object {$_.TypeName -eq $appManagementName} | Start-SPServiceInstance
> $null
```

Finally yet importantly, you can configure one or more custom service application proxy groups to reference the services you have just configured. In order to do that you can use the *New-SPServiceApplicationProxyGroup* and *Add-SPServiceApplicationProxyGroupMember* cmdlets.

Now you are ready to create a new web application, with a root site collection. Here is the script.

```
Add-PSSnapin Microsoft.SharePoint.PowerShell -erroraction SilentlyContinue

$PortalAppName = "Sample Portal"
$PortalAppPort = 80
$PortalAppHostHeader = "portal.sp2013.test"
$PortalAppPool = "SamplePortalAppPool"
$PortalAppPoolUsername = "SP2013\SPContent"
$PortalAppDatabaseName = "SP2013_Farm_WSS_Content_Sample_Portal"

$appPoolAccount = Get-SPManagedAccount -Identity $PortalAppPoolUsername -EA 0
if($appPoolAccount -eq $null)
{
    Write-Host "Please supply the password for the Service Account..."
    $appPoolCred = Get-Credential $PortalAppPoolUsername
    $appPoolAccount = New-SPManagedAccount -Credential $appPoolCred -EA 0
}

$appPoolAccount = Get-SPManagedAccount -Identity $PortalAppPoolUsername -EA 0

# Creates a new claims-based NTLM (default) authentication provider
$ap = New-SPAuthenticationProvider

# Create the Portal Host
New-SPWebApplication -Name $PortalAppName -Port $PortalAppPort -HostHeader $PortalAppHostHeader -URL
("http://" + $PortalAppHostHeader) -ApplicationPool $PortalAppPool -ApplicationPoolAccount
$appPoolAccount -DatabaseName $PortalAppDatabaseName -AuthenticationProvider $ap -
ServiceApplicationProxyGroup "IntranetProxyGroup"

# Create the Portal Root Site Collection
$PortalRootOwner = "SP2013\Administrator"
$PortalRootName = "Sample Portal"

$PortalRootTemplate = Get-SPWebTemplate "STS#0"
New-SPSite -Url ("http://" + $PortalAppHostHeader) -OwnerAlias $PortalRootOwner -Template
$PortalRootTemplate -Name $PortalRootName
```

The previous code creates a sample portal, with an endpoint URL of *portal.sp2013.test*, and a root site collection with a Team Site (*STS#0*) template.

Now browse with your browser to the URL you defined for your portal (remember to configure the corresponding host name in the DNS on the domain controller) and enjoy your SharePoint 2013 Lab Farm hosted on Windows Azure.

Supporting Apps

In case you want to support apps, you will need to create another Web Application for hosting them. Here is a sample PowerShell code excerpt.

```
Add-PSSnapin Microsoft.SharePoint.PowerShell -erroraction SilentlyContinue

$PortalAppName = "Apps Portal"
$PortalAppPort = 80
$PortalAppPool = "AppsAppPool"
$PortalAppPoolUsername = "SHAREPOINT\SPContent"
```

```
$PortalAppDatabaseName = "SP2013_Farm_WSS_Content_DevApps_Portal"

$appPoolAccount = Get-SPManagedAccount -Identity $PortalAppPoolUsername -EA 0
if($appPoolAccount -eq $null)
{
    Write-Host "Please supply the password for the Service Account..."
    $appPoolCred = Get-Credential $PortalAppPoolUsername
    $appPoolAccount = New-SPManagedAccount -Credential $appPoolCred -EA 0
}

$appPoolAccount = Get-SPManagedAccount -Identity $PortalAppPoolUsername -EA 0

# Creates a new claims-based NTLM (default) authentication provider
$ap = New-SPAuthenticationProvider

# Create the Portal Host
New-SPWebApplication -Name $PortalAppName -Port $PortalAppPort -ApplicationPool $PortalAppPool -
ApplicationPoolAccount $appPoolAccount -DatabaseName $PortalAppDatabaseName -AuthenticationProvider
$ap -ServiceApplicationProxyGroup "IntranetProxyGroup"

# Create the Portal Root Site Collection
$PortalRootOwner = "SHAREPOINT\Administrator"
$PortalRootName = "Apps Portal"

$PortalRootTemplate = Get-SPWebTemplate "STS#1"
New-SPSite -Url "http://sp2013srv01/" -OwnerAlias $PortalRootOwner -Template $PortalRootTemplate -
Name $PortalRootName
```

The Web Application endpoint will be the server host name, and you simply need to provision a Site Collection at the root, for example an empty one. You should also configure the DNS to support a subdomain or a secondary domain for hosting your apps. You can find more details about configuring the DNS and the App Management service on this blog post from Mirjam van Olst: http://sharepointchick.com/archive/2012/07/29/setting-up-your-app-domain-for-sharepoint-2013.aspx.

To finalize the configuration of the App Management service, you should execute the following script:

```
Set-SPAppDomain sp2013Apps.local
Set-SPAppSiteSubscriptionName -Name "apps" -Confirm:$false
```

These two cmdlets configure the domain and the prefix for the URL of your apps in the App Management service.

Provisioning the Workflow Infrastructure

In order to provision the workflow infrastructure you need to download and install the Workflow Manager 1.0 component, together with its dependencies, using the Web Platform Installer. To install Workflow Manager 1.0, you need the following:

- .NET Framework 4.5
- Service Bus 1.0 (Plus February 2013 CU)
- Workflow Client 1.0 (Plus February 2013 CU)
- Windows PowerShell 3.0
- SQL Server 2008 R2 Service Pack 1 (SP1), SQL Server Express 2008 R2 SP1, or SQL Server 2012

Remember that installing Workflow Manager on a Domain Controller is a not supported scenario. Thus, you should install it on one or more of the SharePoint servers of the farm, for example. In this document, I will use SP2013SRV2.

Then you will have to configure the Workflow Manager in order to create a new farm. You can use a wizard-like GUI, or you can use a PowerShell script like the following one, which can be created using the wizard.

```
# To be run in Workflow Manager PowerShell console that has both Workflow Manager and Service Bus
installed.

# Create new SB Farm
$SBCertificateAutoGenerationKey = ConvertTo-SecureString -AsPlainText  -Force  -String '*****
Replace with Service Bus Certificate Auto-generation key ******' -Verbose;


New-SBFarm -SBFarmDBConnectionString 'Data Source=SP2013SQL;Initial
Catalog=SbManagementDB;Integrated Security=True;Encrypt=False' -InternalPortRangeStart 9000 -TcpPort
9354 -MessageBrokerPort 9356 -RunAsAccount 'SP2013\WFManagerUser' -AdminGroup
'BUILTIN\Administrators' -GatewayDBConnectionString 'Data Source=SP2013SQL;Initial
Catalog=SbGatewayDatabase;Integrated Security=True;Encrypt=False' -CertificateAutoGenerationKey
$SBCertificateAutoGenerationKey -MessageContainerDBConnectionString 'Data Source=SP2013SQL;Initial
Catalog=SBMessageContainer01;Integrated Security=True;Encrypt=False' -Verbose;

# To be run in Workflow Manager PowerShell console that has both Workflow Manager and Service Bus
installed.

# Create new WF Farm
$WFCertAutoGenerationKey = ConvertTo-SecureString -AsPlainText  -Force  -String '***** Replace with
Workflow Manager Certificate Auto-generation key ******' -Verbose;


New-WFFarm -WFFarmDBConnectionString 'Data Source=SP2013SQL;Initial
Catalog=WFManagementDB;Integrated Security=True;Encrypt=False' -RunAsAccount 'SP2013\WFManagerUser'
-AdminGroup 'BUILTIN\Administrators' -HttpsPort 12290 -HttpPort 12291 -InstanceDBConnectionString
'Data Source=SP2013SQL;Initial Catalog=WFInstanceManagementDB;Integrated
Security=True;Encrypt=False' -ResourceDBConnectionString 'Data Source=SP2013SQL;Initial
Catalog=WFResourceManagementDB;Integrated Security=True;Encrypt=False' -CertificateAutoGenerationKey
$WFCertAutoGenerationKey -Verbose;

# Add SB Host
$SBRunAsPassword = ConvertTo-SecureString -AsPlainText  -Force  -String '***** Replace with RunAs
Password for Service Bus ******' -Verbose;


Add-SBHost -SBFarmDBConnectionString 'Data Source=SP2013SQL;Initial
Catalog=SbManagementDB;Integrated Security=True;Encrypt=False' -RunAsPassword $SBRunAsPassword -
EnableFirewallRules $true -CertificateAutoGenerationKey $SBCertificateAutoGenerationKey -Verbose;

Try
{
    # Create new SB Namespace
    New-SBNamespace -Name 'WorkflowDefaultNamespace' -AddressingScheme 'Path' -ManageUsers
'SP2013\WFManagerUser','Administrator@SP2013' -Verbose;

    Start-Sleep -s 90
}
Catch [system.InvalidOperationException]
{
}

# Get SB Client Configuration
$SBClientConfiguration = Get-SBClientConfiguration -Namespaces 'WorkflowDefaultNamespace' -Verbose;

# Add WF Host
```

```
$WFRunAsPassword = ConvertTo-SecureString -AsPlainText  -Force  -String '***** Replace with RunAs
Password for Workflow Manager ******' -Verbose;


Add-WFHost -WFFarmDBConnectionString 'Data Source=SP2013SQL;Initial
Catalog=WFManagementDB;Integrated Security=True;Encrypt=False' -RunAsPassword $WFRunAsPassword -
EnableFirewallRules $true -SBClientConfiguration $SBClientConfiguration -EnableHttpPort  -
CertificateAutoGenerationKey $WFCertAutoGenerationKey -Verbose;
```

Now you simply need to configure, on the SharePoint side, the address of the Workflow Manager farm endpoint, for a specific target site. Here is the PowerShell script.

```
Register-SPWorkflowService -SPSite 'http://portal.sp2013.test/' -WorkflowHostUri
'https://sp2013srv2:12290/' -AllowOAuthHttp -Force
```

Now, you are done! Enjoy!

Note: in case of any issues in this script, please let me know (paolo@pialorsi.com) and I will be happy to fix it or improve it. Thanks.