6/26/2014

Azure Automation

Users guide

Santhosh Konka / Shital Savekar
KONKASOFT LLC

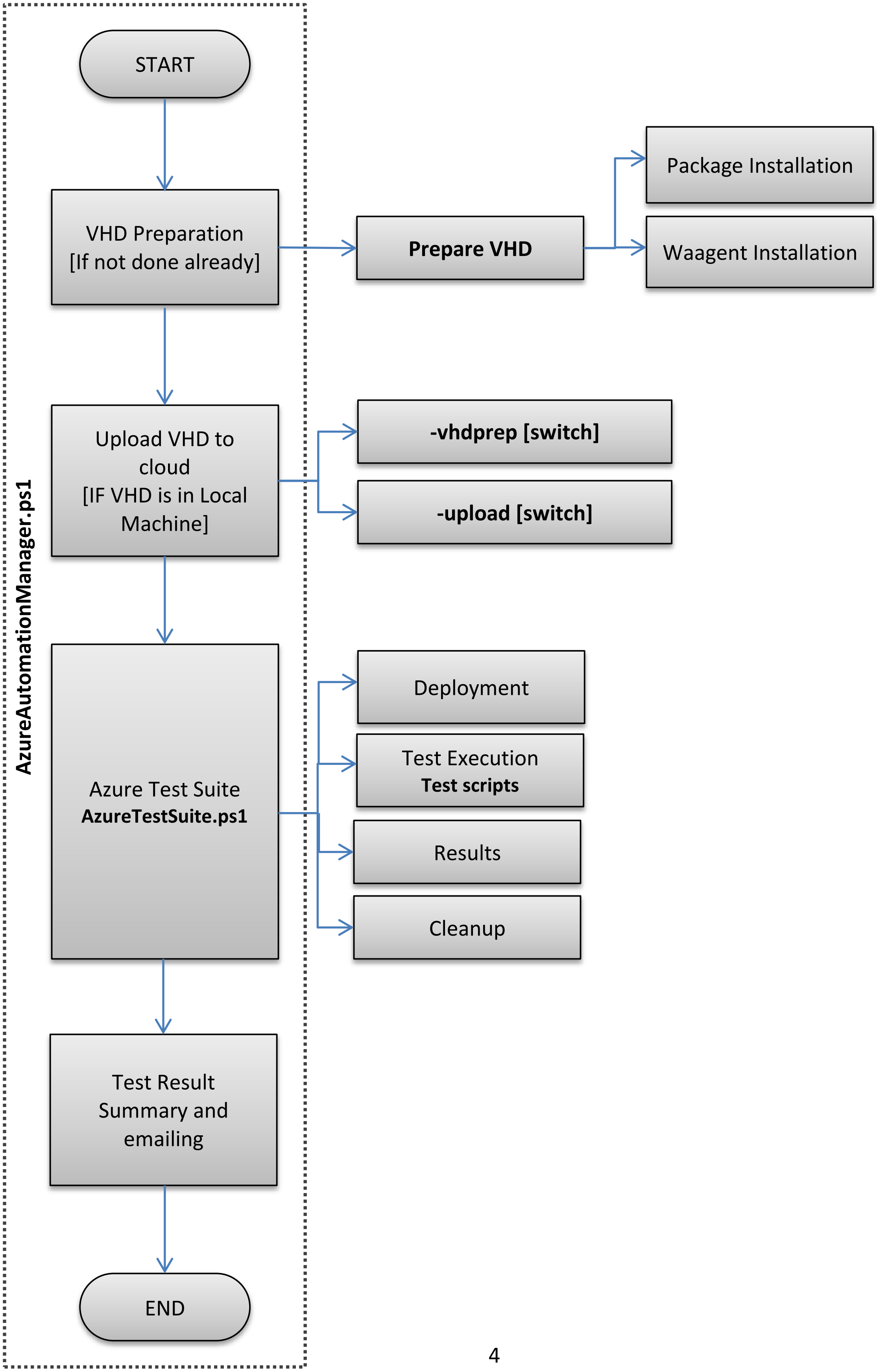
Table of Contents

Chapter 1 - Overview

Azure automation is the project for primarily running the Test Suite in the Windows Azure environment to test the Linux Agent for Windows Azure. Azure automation project is a collection of PowerShell, BASH and python scripts. The test ensures the functionality of Windows Azure Linux Agent and Windows Azure support for different Linux distributions. This test suite focuses on the Build Verification Tests (BVTs), Azure VNET Tests and Network tests. The test environment is composed of a Windows Machine (With Azure PowerShell SDK) and the Virtual Machines on Azure that perform the actual tests.

Chapter 2 - Framework Details

Flow Chart



Framework Components.

Microsoft Tools

Power Shell script name	Description
Windows PowerShell	Windows PowerShell is powerful scripting language to automate tasks on Windows Environment.
Windows Azure PowerShell	Azure PowerShell provided Windows PowerShell cmdlets to do different task on Azure.

PowerShell Scripts (Framework Scripts):

Power Shell script name	Description
AzureAutomationManager.ps1	The main Power Shell wrapper script to initiate automation. This script accepts arguments from user, parses them accordingly and decides the next flow.
AzureTestSuite.ps1	AzureAutomationManager.ps1 will initiate this script file to start automation cycle. This script extracts data from AzureXML file like test cycles, test data etc.
AzureWinUtils.psm1	This PowerShell library contains functions related Azure specific tasks.
RDFELibs.psm1	This is library of PowerShell Functions. It contains functions related to Deployments, Test Cases etc.

Third Party tools

Power Shell script name	Description
pscp.exe	Windows console executable to execute command on Linux VM over SSH and return the output
plink.exe	Windows console executable to upload / download files to/from Linux VM over scp.

Other Files

Power Shell script name	Description
Azure_ICA_ALL.xml	This XML file contains all data required to run automation.

Component Explanation

AzureAutomationManager .ps1

This is an entry point script which manages initialization of the Azure Automation. It initially checks Basically Check for the User Inputs and Validates it. Automation manager accepts following parameters.

1. **-XmlConfigFile**
   Expected Value: Path to the Azure XML file.
2. **vhdprep**
   It is a switch, if used, then VHD preparation task will be performed.

Please see also - `OnCloud`

3. **OnCloud**
   It is a switch, it should be used in combination with `vhdprep`
4. **testCycle**
   Expected Value: Test Cycle Name.
   A test cycle must be defined in the XML file, with cycle name. This test cycle contains test case names which should be executed under that cycle name.
5. **runtests**
   It is a switch, if used, then tests will start to execute on give **testCycle.**
6. **Distro**
   You need to give a distro name on which tests should be conducted. Distro Name should be explicitly mentioned in the Azure XML file with the OS Image Name.
7. **email**
   It is a switch, if used, automation logs will be mailed to desired recipients.**.**
   Note – recipients list is should me mentioned in XML file. Please refer XML file documentation.

AzureTestSuite.ps1

Once the AzureAutomationManager.ps1 file accepts valid arguments from user, it passes proper parameter to AzureTestSuite.ps1 and executes it. AzureTestSuit.ps1, on receiving inputs gathers all the test cycle data from Azure_XML file and executes tests serially.

It takes result of each test and saves it for creating final test summary. After finishing the complete test cycle, this script passes test cycle summary, back to AzureAutomationManager.ps1.

Chapter 3 - Prepare Your Machine for automation cycle.

Prerequisites

1. You must have a Windows Machine with PowerShell.
   Tested Platforms:
   a. Windows 7x64
   b. Windows 8x64
   c. Server 2008
   d. Server 2012.
2. You must be connected to Internet.
3. You must have a valid Windows Azure Subscription.
   a. Subscription Name
   b. Subscription ID

Download Latest Automation Code.

1. Checkout from https://github.com/Azure/azure-linux-automation.git

Download Latest Azure PowerShell

2. Download Web Platform Installer from :
   http://go.microsoft.com/fwlink/p/?linkid=320376&clcid=0x409
3. Start Web Platform Installer and select Azure PowerShell and proceed for Azure PowerShell Installation.

Authenticate your machine with your Azure Subscription

There are two ways to authenticate your machine with your subscription.

1. **Azure AD method**
   This creates a 12 Hours temporary session in PowerShell, in that session, you are allowed to run Windows Azure Cmdlets to control / use your subscription. After 12 hours you will be asked to enter username and password of your subscription. This may create problems long running automations, hence we use certificate method.
2. **Certificate Method.**
   To learn more about how to configure your PowerShell with your subscription, please visit ***here.***

Download 3rd Party utilities.

1. Download Putty executables from http://www.putty.org and keep them in `.\AutomationDirectory\tools` You should have the following utilities:
   - plink.exe
   - pscp.exe
   - putty.exe
   - puttygen.exe

Update Azure_ICA_all.xml file.

1. Setup Subscription details.
   Go to Config > Azure > General and update following fields :
   a. SubscriptionID
   b. SubscriptionName

c. CertificateThumbprint (Make sure you have installed a management certificate and can access it via the Azure Management Portal (SETTINGS->MANAGEMENT CERTIFICATES).)
d. StorageAccount
e. Location
f. AffinityGroup (Make sure that you either use <Location> or <AffinityGroup>. Means, if you want to use Location, then AffinityGroup should be blank and vice versa)

Example :

```
<General>
    <SubscriptionID>Your Subscription ID</SubscriptionID>
    <SubscriptionName>Your Subscription Name</SubscriptionName>
    <CertificateThumbprint>Certificate associated with your
subscription</CertificateThumbprint>
    <ManagementEndpoint>https://management.core.windows.net</ManagementEndpoint>
    <StorageAccount>your current storage account</StorageAccount>
    <Location>Your preferred location</Location>
    <AffinityGroup></AffinityGroup>
</General>
```

2. Add VHD details in XML File.
   Go to Config > Azure > Deployment > Data.
   Make sure that your "VHD under test" should be present here in one of <Distro>..</Distro> entries.
   If your VHD is not listed here. Create a new Distro element and add your VHD details.
   Example.

```
<Distro>
    <Name>Distro_Name</Name>
    <OsImage>Distro_OS_Image_Name_As_Appearing_under_Azure_OS_Images</OsImage>
</Distro>
```

3.Save file.

Prepare VHD to work in Azure:

[Applicable if you are uploading your own VHD with Linux OS to Azure.]

A VHD with Linux OS must be made compatible to work in Azure environment. This includes –

1. Installation of Linux Integration Services to Linux VM (if already not present)
2. Installation of Windows Azure Linux Agent to Linux VM (if already not installed.)
3. Installation of minimum required packages. (*Applicable if you want to run Tests using Automation code*)

Please follow the steps mentioned at:
http://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-create-upload-vhd/

Prepare VHD to work with Automation code.

[Applicable if you are using already uploaded VHD / Platform Image to run automation]

Minimum Required Packages:

To run automation code successfully, you need have following packages installed in your Linux VHD.

1. iperf
2. mysql-server
3. mysql-client
4. gcc
5. gcc-c++
6. bind
7. bind-utils

8. bind9
9. python
10. python-pyasn1
11. python-argparse
12. python-crypto
13. python-paramiko
14. libstdc++6
15. psmisc
16. nfs-utils
17. nfs-common
18. tcpdump

Installation of Minimum Required Packages

This task can be automatically done with following steps:

1. Create a new `Distro` XML tag in `$xmlConfigFile.config.Azure.Deployment.Data.Distro` with unique distro name to `Name` property.
2. Assign your OsImage name to the <OsImage>.
3. Now, you have done all prerequisites. Now run the following command : .\AzureAutomationManager.ps1 -xmlConfigFile <FullPathToXMLFile> -Distro <UniqueDistro> -cycleName autosetup

This will do following procedure:

1. Deploy A VM with your OS Image.
2. Install the minimum required packages
3. Create and Public an OS image which can be used for executing tests.

Create SSH Key Pair

[PublicKey.cer – PrivateKey.ppk]

A Linux Virtual machine login can be done with Password authentication or SSH key pair authentication. You must create a Public Key and Private key to run automation successfully. To learn more about how to create SSH key pair, please visit ***here.***

After creating Public Key (*.cer) and putty compatible private key (*.ppk), you must put it in your `automation_root_folder\ssh\` folder and mention their names in Azure XML file.

VNET Preparation (required for executing Virtual Network Tests)

Create a Virtual Network in Azure

A virtual network should be created and connected to Customer Network before running VNET test cases. To learn about how to create a virtual network on Azure, please visit ***here.***

Create A customer site using RRAS

Apart from Virtual Network in Azure, you also need a network (composed of Subnets and DNS server) to work as Customer Network. If you don't have separate network to run VNET, you can create a virtual customer network using RRAS. To learn more, please visit ***here.***

Chapter – 4 How to Start Automation

[Before starting Automation, make sure that you have completed steps in Chapter 3.]

1. Start PowerShell with Administrator privileges.
2. Navigate to folder where automation code exists
3. Issue automation command.

Automation Cycles Available –

1. BVT
2. NETWORK
3. VNET
4. E2E
5. E2E-TIMESYNC
6. E2E-TIMESYNC-KERNBANCH
7. WORDPRESS-NOLB
8. WORDPRESS-LB
9. DAYTRADER-NOLB
10. DAYTRADER-LB

[These cycles are illustrated in next chapter.]

Command to Start any of the Automation Cycle:

```
.\AzureAutomationManager.ps1 -xmlConfigFile .\Azure_ICA_ALL.xml -
runtests -email –Distro <DistroName> -cycleName <TestCycleToExecute>
```

Test Cycles

BVT

This automation cycle checks VHD prerequisites. It checks

1. Windows Azure Linux Agent
2. If VHD prerequisites are fulfilled or not and other sanity checks

For complete list of tests included, please visit here.

NETWORK

This automation cycle tests data transfer between different Linux VMs in distinct scenarios. For E.g.

1. Data transfer between two Azure Linux VM
2. Data transfer between an Azure Linux VM and VM on internet.

For complete list of tests included, please visit here.

VNET

VNET stands for Virtual Network. To learn more about virtual network, please visit ***here.***

This automation cycle tests connectivity, functionality of Virtual Network.

Note: Before running Virtual Network Tests, you virtual network must be connected to on premises network and you need to setup a DNS server in your premises.

For complete list of tests included, please visit here.

E2E (End to End tests)

This automation cycle checks some important tasks like – creation, deletion of user to new Linux VM. Attaching and Removing disks from live Linux Machine.

E2E-TimeSync

This automation cycle checks only one test – Time Sync. It creates a virtual machine and verifies that VM is consistently in sync with NTP servers for 2 days. (This time can be changed by editing the test case)

E2E-TimeSync-Kernbench

This is same is E2E-Timesynch except, while running the test, a high CPU load will be generated using Kernbench software. We expect that, Time sync should not be deflected even in High CPU load.

WordPress

This automation cycle checks working of LAMP in Azure Environment. It deploys a WordPress application on Linux VM and verifies it.

DayTrader

This automation cycle checks working of Web Server and Database transitions in Azure Environment. It deploys a Daytrader app on top of IBM WebSphere application on Linux VM and verifies it.

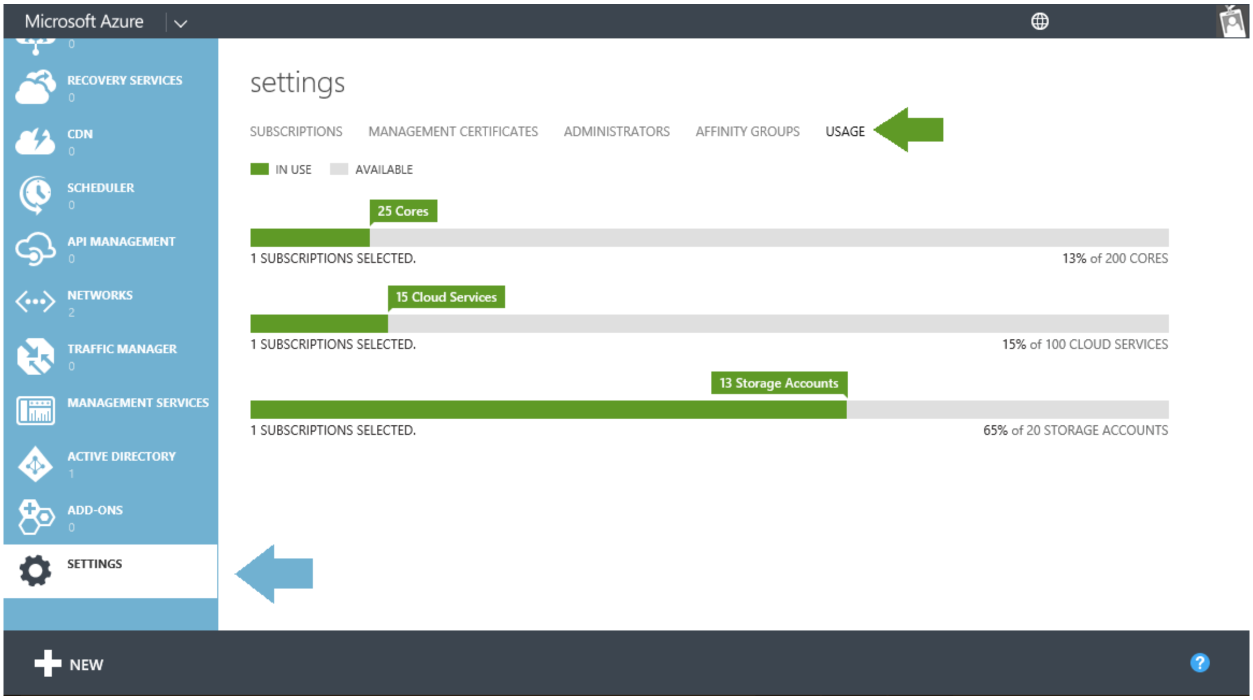
Troubleshooting

Sr. No.	Error Line	Solution
1	File AzureAutomation.ps1 cannot be loaded because running scripts is disabled on this system.	**Make sure that your PowerShell Execution policy is set to "Unrestricted"** Execute with Administrator privileges: `Set-ExecutionPolicy -ExecutionPolicy Unrestricted`
2	Unable to execute Azure PowerShell Command	**Make sure that your subscription is successfully added to your PowerShell.** Execute – Get-AzureSubscription command and check if your subscription is listed in the output. **Make sure that, subscription details are correctly added to Azure XML File.** 1. Your Azure PowerShell must be able to access your Azure Subscription. To access your Azure Subscription via PowerShell, you need to have a common certificate which is uploaded to your subscription and it also must be installed in your local machine.
3	Deployments are failing	**1.Unable to Create Service** Make Sure that, your service count is not exceeded. (Your Azure subscription have an upper limit of hosted service count, virtual machines and storage accounts.) Make sure that your quota is not full. Refer Figure 1 from References

Definitions

- **Azure** – Cloud Service by Microsoft. More information at http://azure.microsoft.com/en-us/
- **Deployment** – Creation of VM in a hosted service.
- **Hosted Service** – A hosted service is cloud endpoint, where you can create all your VMs. A VM must be placed in Hosted service
- **PowerShell** – A scripting language
- **VM** – Virtual Machine
- **VNET** – Virtual Network.

References:



settings

🟩 IN USE ▨ AVAILABLE

25 Cores

1 SUBSCRIPTIONS SELECTED. 13% of 200 CORES

15 Cloud Services

1 SUBSCRIPTIONS SELECTED. 15% of 100 CLOUD SERVICES

13 Storage Accounts

1 SUBSCRIPTIONS SELECTED. 65% of 20 STORAGE ACCOUNTS

Figure 1 Check Subscription Usage