**Azure Jenkins DevTest Environment**

**ISV Dev Test Scenario**

The cloud presents new opportunities for ISVs to optimize their dev/test cycles by enabling dev-ops teams to easily create test environments on demand. Test environments are often something that developers and IT pros like to be able to recreate often in a repeatable fashion as part of the software dev/test cycle. The Azure cloud provides capabilities to define infrastructure “resources as code” via ARM templates and the Azure CLI. Having infrastructure resources defined as code enables Azure admins to recreate those resources in a repeatable fashion.

For some ISVs, their customers are often an integral part of the test & QA cycle. These ISV customers might want to have access to the test environment in order to verify the ISV’s testing, or perform additional testing themselves before accepting a new version of the ISV’s software. In order to provide an ISV’s customer access to a test environment, the ISV will need to ensure it is secured, and isolated from both the ISV’s and the ISV’s customer’s network environments. This is a scenario that Azure makes relatively easy to setup.

This project provides an isolated and secure dev/test environment in Azure for running tests with Jenkins.

**Jenkins on Azure**

Jenkins is a CI/CD tool used by many developers and organizations.

A common pattern when using Jenkins for testing is to have a “master” Jenkins server that can deploy application code to one or more “agent” VMs, and then execute test suites on them. Each “agent” VM reports its results back to the Jenkins “master” where the results can be analyzed through the Jenkins web interface on the “master”.

To conserve on resources in a cloud environment, it would be useful to have Jenkins dynamically create “agent” VMs as needed for the test process, and then dispose of them when they are no longer needed. This is the perfect scenario for an on-demand computing environment like Microsoft Azure.

Microsoft has developed an Azure Agent VM plugin for Jenkins that enables Jenkins to do exactly this! Microsoft has also developed a plugin for Jenkins that enables it to store and retrieve artifacts in Azure Storage.

**Project Goals**

The goals of this project include:

* Create a secure test environment in Azure that is isolated from both an ISV’s corporate network, the Internet, and the ISV’s customer’s network.
* Provide access to the test environment via an RDP server in a DMZ.
* Use the Microsoft developed “Azure Agent VM” plugin for Jenkins to automate the spin up of ‘slave’ VMs for testing, and then shut them down when testing is complete.

**Project Details**

This project provides a Bash script using the Azure CLI 2.0 that deploys a private dev/test environment in Azure. It will feature the following:

* A virtual network
* Two subnets -> 1) Jumpbox DMZ & 2) DevTest
* A Jenkins Master server on a private DevTest subnet that is not accessible from the Internet.
* A Windows Server 2016 VM configured as a "jumpbox" with RDP access.
* Instructions for how to use & configure all of the pieces.
  + Azure plugin for Jenkins to enable Jenkins to create/delete Azure Agent VMs
  + Azure plugin for Jenkins to read/write artifacts from Azure Storage.

**Configuration Instructions**

After running the script azurejenkinsdevtest.azcli from the Bash shell (Mac, Linux, or WSL for Windows 10), follow the steps below to set up and configure the environment. Eventually, these steps can be scripted so that they are done on an automated basis.

**Login Credentials**

Below are the login accounts used by the various components:

**“win-jumpbox”**

* User: jumpboxadmin
* Password: DevTestPa55W0rd!

**“jenkinsmaster”**

* Linux OS credentials:
  + User: jenkinsadmin
  + Password: DevTestPa55W0rd!
* Jenkins Web Console credentials:
  + **Note:** Initial password is set by Bitnami VM template. You must retrieve it from the log in the Azure Portal under “Boot Diagnostics” for this VM. It is then recommended to change it to one you know.

Below are the steps required to configure the Windows Server “jumpbox” with Azure tools and Jenkins master with Azure plugins.

**Windows Server “Jumpbox”**

* + Login to the jumpbox via RDP from the Public IP as listed in the Azure Portal
  + Disable IE Enhanced protection mode (to enable you to download & install the tools)
  + Install Putty - [www.putty.org](http://www.putty.org)
  + Install PowerShell ISE - already installed on Server 2016
  + Install Azure Powershell Commandlets - <https://azure.microsoft.com/en-us/downloads/>
  + Install Azure CLI - Installed from WebPI - <http://aka.ms/webpi-azure-cli>

**Jenkins Master**

* + Get initial password for the Jenkins web console from the “jenkinsmaster” VM’s “Boot Diagnostics” in the Azure Portal
  + In the portal, obtain the “jenkinsmaster” private IP address by clicking on the “jenkins-NIC”
  + From the jumpbox’s browser, login to the Jenkins web console at the VM’s private IP obtained in the previous step (i.e. ‘http://10.0.2.x’)  
     **NOTE:** Bitnami Jenkins image does not use port 8080 for the web console. Port 80 on the root will reroute to /Jenkins”
    - User: user
    - Initial pwd: <obtain from boot diagnostics>
    - Change the password as soon as possible
  + Create an Azure Service Principal to authenticate Jenkins Agent plugin against your Azure subscription. Think of an Azure Service Principal as a “service account” that will provide the Jenkins master and its plugins with access to the Azure resources it needs within your subscription.  
      
    **Note:** This can be done from any computer, including your developer workstation. However, it is easiest to do this from a Mac or Linux machine that can run the bash script provided by the Azure VM Agent plugin for Jenkins. You can also run this script on a Windows 10 machine by using WSL for Windows 10.  
      
    Run the following bash script to generate the Azure Service principle:

bash <(curl -s <https://raw.githubusercontent.com/Azure/azure-devops-utils/master/bash/create-service-principal.sh>)

* + - The script will prompt you for credentials to the Azure subscription
    - You NEED TO BE THE Azure Subscription OWNER, not a co-admin!!!
    - COPY the entire results from the console and store in a text file. Treat as a “secret” like a database string  
      **Note:** If using curl to pull a SAS blob from storage, need to enclose the URL in quotes.
* Install Azure VM Agent & Azure Storage Plugins in the Jenkins web console
* Configure Azure Storage Plugin:
  + Enter the storageaccountname and storageaccountkey for the Azure storage account you wish to link.
  + This information can be obtained from the Azure portal.
* Configure Azure VM Agent plugin:
  + Enter the Azure service principal information (to provide Jenkins with credentials to Azure account)
  + Fill out the plug-in’s template form with all of the settings needed for the slave VMs. Some settings to be aware of:
    - **Labels:** The value you set here is what you will use in the Groovy script/Pipeline tool within Jenkins to refer to the slave VM
    - **Region:** This is the datacenter your slave VM will be created in. It should be the same region as your Jenkins master.
    - **Resource Group:** This is the resource group your VM will deployed to. It MUST be the same Resource Group as the VNet you will deploy to, AND also the same Resource Group as the Storage Account the custom (generalized) VM image is. If the Resource Group names don’t match, the error(s) you receive are not indicative of the problem.
    - **Virtual Network Name:** Must be the same VNet as the Jenkins master
    - **Subnet Name:** Should be the same as Jenkins master, but doesn’t have to be (as long as NSG security settings are not restricted between subnets.
    - **VM Image:** The Agent plugin works with only basic Windows and Linux VM images in the Azure marketplace. (ie “vanilla” Windows & Linux installations). However, you can use the Agent plugin to spin up/down instances of a custom VM that you provide. To do that, see the steps below.

**Jenkins Azure VM Custom Agent**

To use a custom VM image for the Agent VM, you must build it (or upload it) into Azure, then “generalize” the image so it can be spun up multiple times within Azure.

* Create and/or customize a VM from the Azure marketplace and deploy it into your Azure subscription..
* Ensure that the VM’s Resource Group is the same used by “jenkinsmaster”
* Ensure that the VM’s Vnet/Subnet: Jenkins-Subnet
* Ensure the VM has no Public IP
* <INSTALL CUSTOM SOFTWARE PREREQUISITES WITHIN THE VM>
* Generalize the Linux VM - <https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-linux-capture-image-nodejs?toc=%2fazure%2fvirtual-machines%2flinux%2ftoc.json>
* Grab URI for generalized image to use in the Azure VM Agent plugin settings.

For more details on how to configure the Azure VM Agent plugin in Jenkins, as well as create a sample Jenkins process to test it out, see <http://aka.ms/azjenkinsagents>