DevTest 9.x – Best Practice Architecture

#2 - Hybrid environment (on premise and public cloud)

Introduction

Customers often ask “Where do we Start” when setting up and configuring their DevTest architecture. Of course, this really depends on how they plan to use DevTest - whether using Virtual Services or running tests or maybe a combination of both. So based on potential usage of DevTest, we started building a Best Practice Architecture Guide based on real world deployments. As a part of this initiative, we will build out several architectures that are implemented by either customers or CA Services or are new/tested architectures that fit a specific need.

Our intention is to provide options for customers, partners and CA Services on how to implement DevTest within their environment to meet the specific use cases.

This document will contain guidance on implementing DevTest on premise, in private or public clouds or hybrid environments. It will describe the specific DevTest Capabilities.

This second use case walks through setting up DevTest 9.0 in a Microsoft ® Windows Azure (Azure) environment. The Source Control Management (SCM) is located on premise (on physical systems or in private cloud).

Document Changes

Version	Date	Primary Author	Description
1.0	12/04/2015	Ulrich Vogt Koustubh Warty	Initial creation #2 - Hybrid environment (on premise and public cloud)

Contents

Business Case

This architecture sample covers the business case of a customer who wants to deploy a DevTest configuration in a public cloud for internal or external partners to access.

In this sample, implementation of this DevTest architecture consists of installing and configuring all the DevTest components in Microsoft Azure environment on individual VMs as explained in the document. The SCM will be on the premise. Every time a DevTest user works on any DevTest assets namely Test Cases, Virtual Services, mar files, etc., they will checkout from the SCM server locally, and after modification of those assets, will check back the modified asset(s) to the SCM server.

Architectural Considerations

Following restrictions apply to a DevTest architecture and are recommended to follow:

1. To warrant performance Registry database must be located electronically close to DevTest server components. Please see Database Requirements for details.
2. Enterprise Dashboard database contains audit data. Resources related to audit data must not be released.
3. The SCM used here is the free standard edition of the Visual SVN server from https://www.visualsvn.com/server/
4. Every user machine will have a client that can talk to the SVN server. In this case, we have used the free version of Tortoise SVN Client from https://tortoisesvn.net/downloads.html

   Note: This document does not go into the details on installing and configuring the SVN server and the client.

This architecture is designed for Service Virtualization and Application Test.

Architecture Diagram

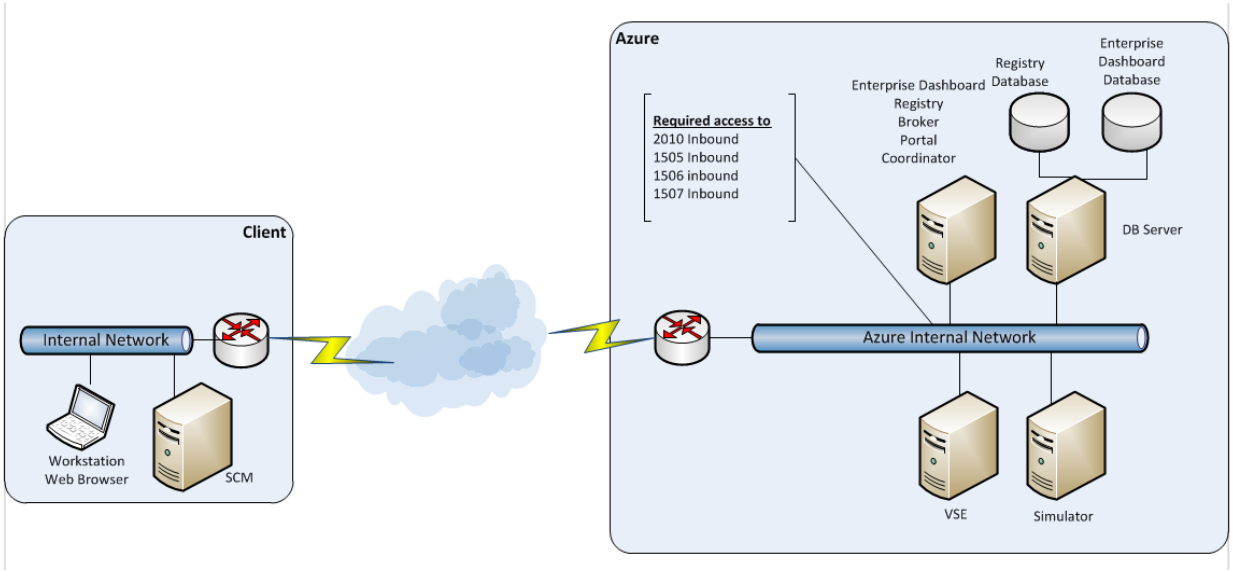


Figure 1 - Architecture Diagram

The Architecture diagram shows ports that need to be open for external DevTest service access. For the sake of simplicity, it does not show ports that needs to be open for other services, such as Remote Desktop Protocol or Windows PowerShell.

Implementation Details

Installation of some DevTest components depends on business case requirement. Most components are required regardless of use case to provide the DevTest infrastructure.

DevTest Server Components

In order to run Service Virtualization or Application Test in cloud environments DevTest VSE and DevTest Simulator service must be provided. These components require additional DevTest services installed and running:

DevTest service	**Requires**
Simulator	Coordinator, Registry
VSE	Registry
Shared among DevTest components	Portal

- Enterprise Dashboard

  The Enterprise Dashboard service and its database is installed on the Azure VM in the cloud. For the sake of simplicity, the database is installed locally on the same system as the Enterprise Dashboard service, but a dedicated database server can also be used.

  Installation of the Enterprise Dashboard service is mandatory.

- Registry

  Registry service and its database are installed in the cloud as well. The Registry database is installed on a dedicated database server to better visualize the need for a database resource. The Registry Service requires access to the Enterprise Dashboard service for license information and for providing usage data.

  Installation of a Registry service is required for every DevTest use case.

- Broker

  Broker Service is installed on the same system as the Registry Service. Broker Service is required for Continuous Application Insight.

- Portal

  Portal Service is installed on the same system as the Registry Service. Portal Service is required for Web access to DevTest server components

- Coordinator

  Coordinator Service is installed on the same system as the Registry Service. Coordinator Service is required to stage test cases to Simulator Services. A Coordinator Service can manage multiple Simulator services.

  Coordinator services are required for Application Test use cases only.

- Simulator

  Simulator service is installed on a separate system. If multiple Simulator services are required for scalability reasons, they are assumed to be installed on different systems.

  Simulator services are required for Application Test use cases only. Therefore VMs for Simulator services can be provided on demand.

- Virtual Service Environment (VSE)

  VSE service is installed on a separate system. If multiple VSE services are required for scalability reasons, they are assumed to be installed on different systems.

  VSE services are required for Service Virtualization only. Therefore VMs for VSE services can be provided on demand.

Network

The Azure network blocks any inbound network traffic that is not required to access the provided systems. Any ports that are needed to access DevTest components must be published in Azure. Some well-known port numbers in DevTest might not be available for publishing, but must be mapped or configured to different port numbers.

Similarly, customer and client networks might also be restricted for inbound and outbound traffic. These firewalls also have to be configured to allow access to the DevTest ports in Azure.

System Setup

DevTest is set up in a distributed configuration. It is distributed in two separate environments, in the DMZ and in Microsoft Azure.

- Microsoft Azure

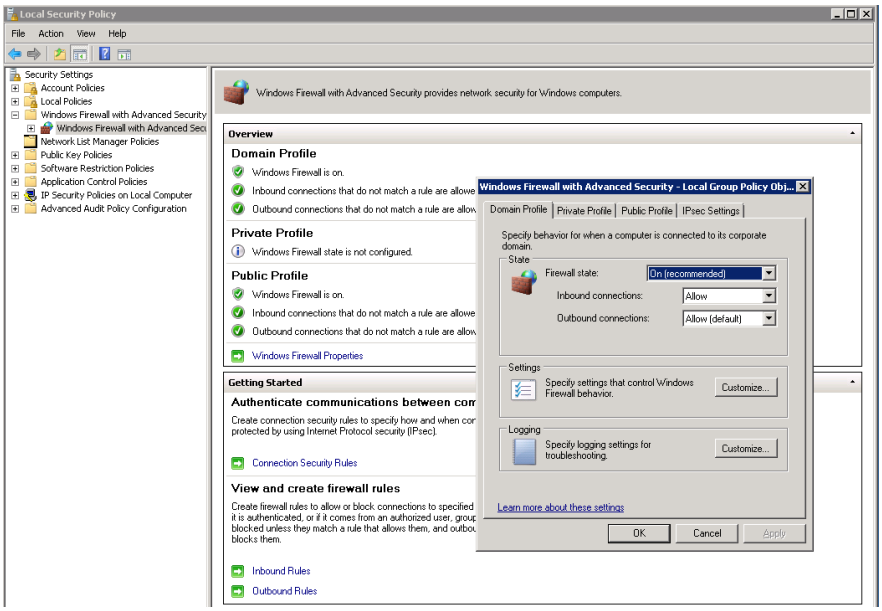
Service	Host name	Public IP
Enterprise Dashboard, Registry, Broker, Coordinator, Portal	devtest-uc2-reg	104.208.38.99:62306
Simulator	devtest-uc2-sim	104.208.38.99:54750
VSE	devtest-uc2-vse	104.208.38.99:57151
Visual SVN	devtest-uc2-sim	104.208.38.99:54750

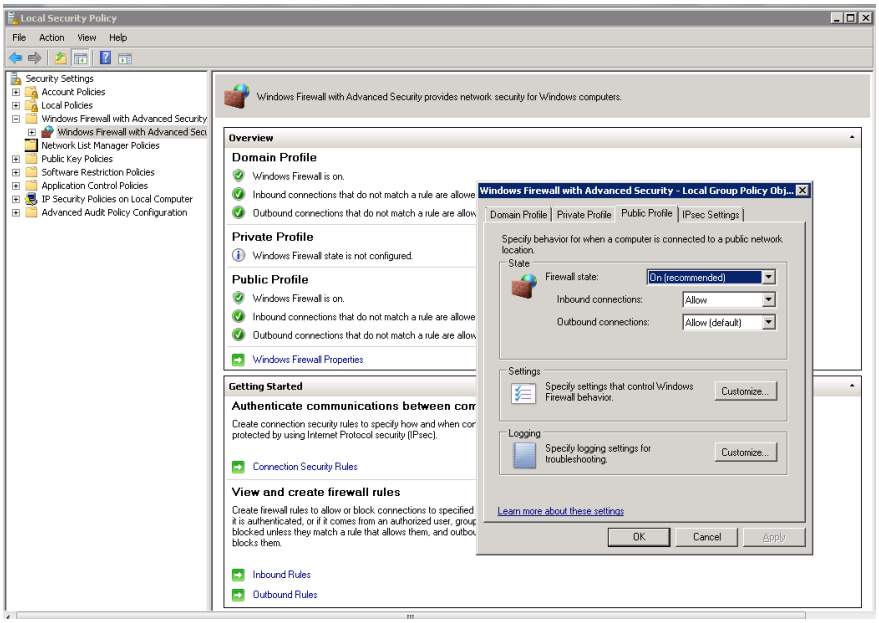
Note: The ports in the table above 62306, 54750 and 571515 are all ports that are used to connect to the VM using RDP.

Firewall Policies

All virtual machines of the DevTest cloud in Azure are configured with identical firewall settings:

- Same firewall settings for Domain Profile, Private Profile and Public Profile
    - Firewall is activated (on) for Domain and Public Profiles, but off for Private Profile
- If this firewall is not configured accordingly then there are no Inbound and Outbound connections available between the VMs and the outside world





Azure Setup

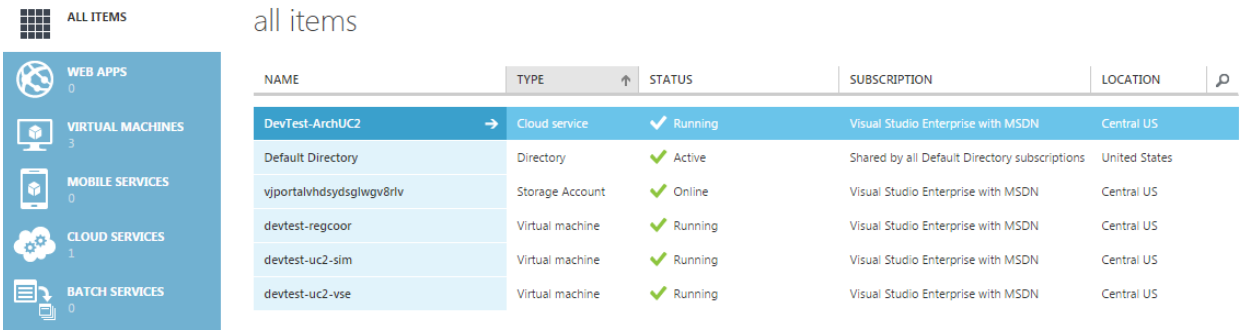
The Microsoft Azure setup of the DevTest components requires to create and to configure

- Cloud Service to configure. A cloud service is required before creating any VM on the Azure site. The cloud service hosts all the VMs, databases and webapps, etc. If a cloud service does not exist then the administrator is prompted to create a new one during the provisioning of the first VM.
- The set of Virtual Machines to run the assigned DevTest components

The virtual machines are all based on Microsoft Windows 2012 Server, which is a 64-Bit system.

Cloud Service

In Azure, the Cloud Service 'DevTest-ArchUC2' defines the environment for the virtual machines discussed in this document.

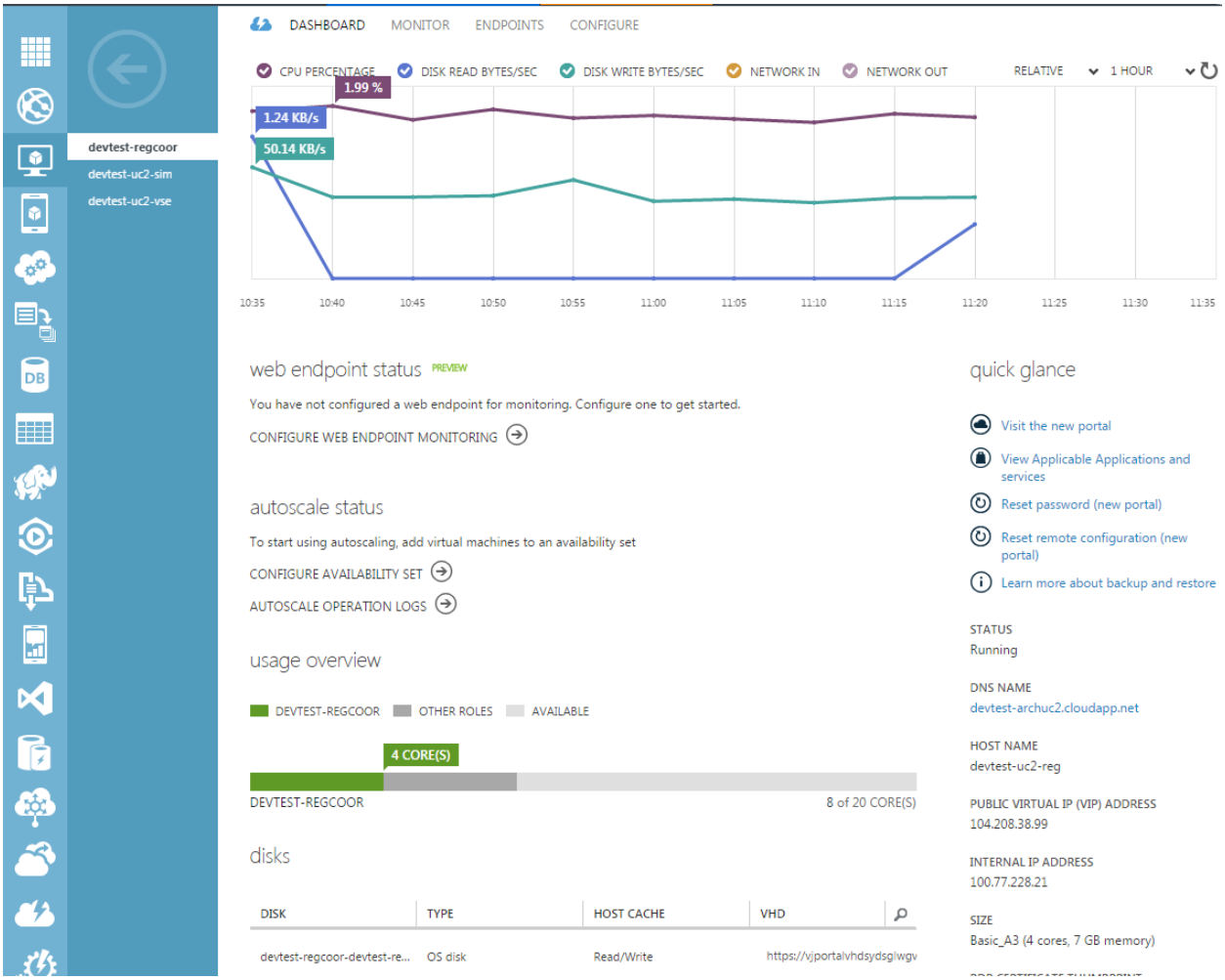


Virtual Machines

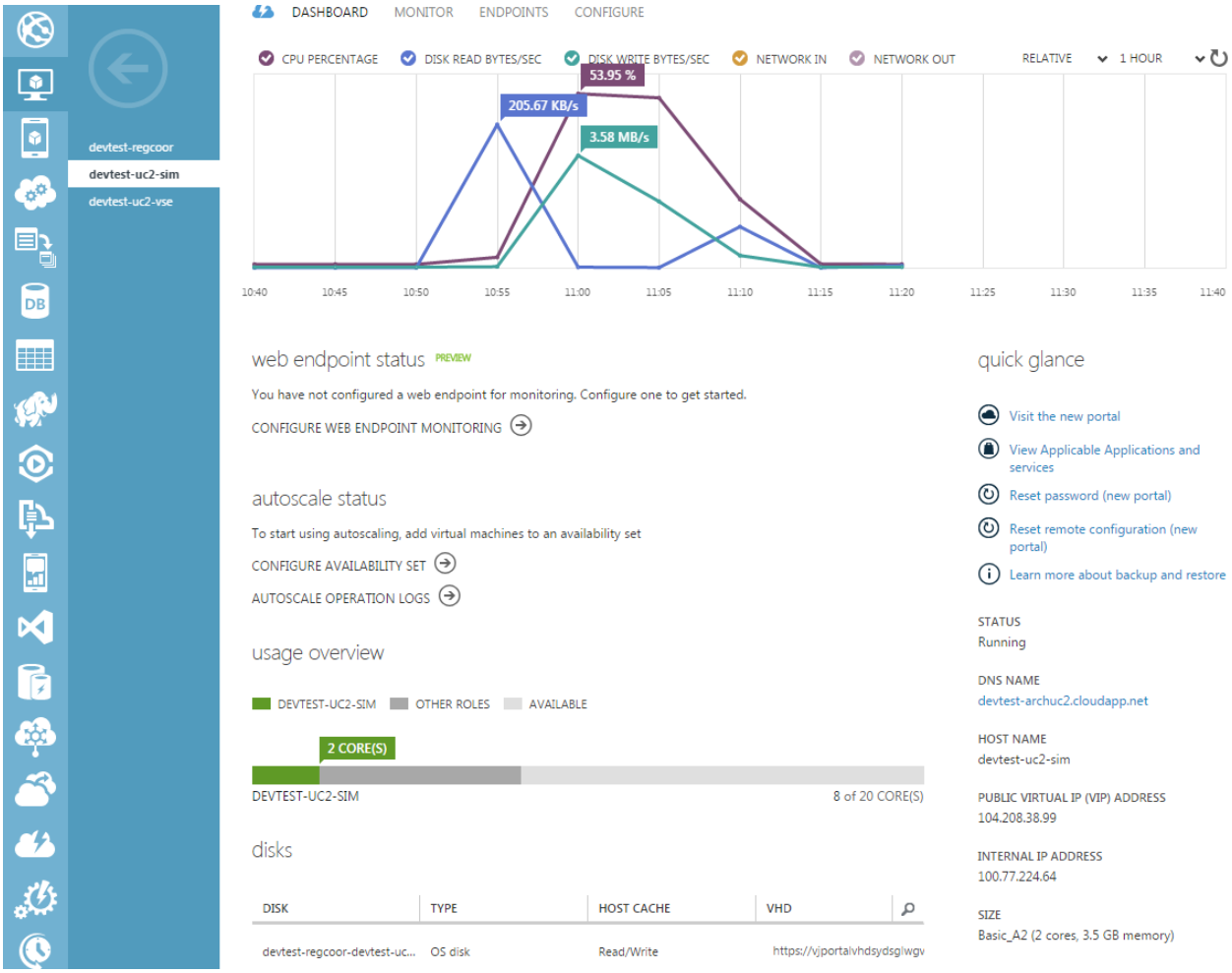
The DevTest Server components run in Microsoft Azure. They are configured with following resources:

Host name	Public IP	Private IP	CPU (Cores)	RAM	#Disks	#NICs
devtest-uc2-reg	104.208.38.99	100.77.228.21	2 Cores	7.0 GB	1 x 60GB	1 NIC
devtest-uc2-sim	104.208.38.99	100.77.224.64	2 Cores	3.5GB	1 x 60GB	1 NIC
devtest-uc2-vse	104.208.38.99	100.77.216.84	2 Cores	3.5GB	1 x 60GB	1 NIC

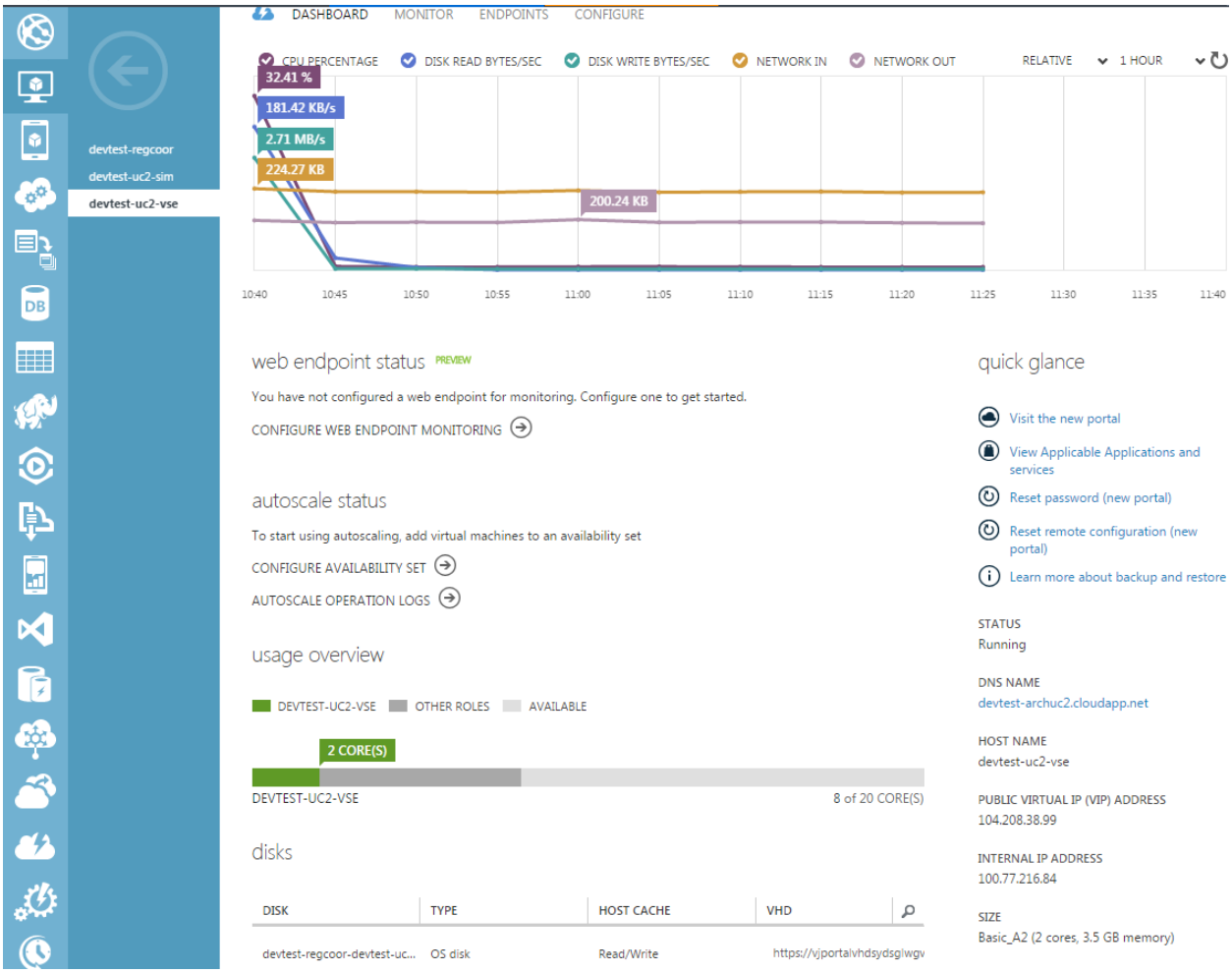
VM for Enterprise Dashboard/Registry/Coordinator/Portal



VM for Simulator



VM for VSE



HTTP/TCP Ports exposed for each of the DevTest Components

Azure requires and offers a NAT configuration to expose access to services available on the virtual machines deployed in Azure.

External access to following virtual machines in Azure must be configured

Server	**VM name**
Registry/Coordinator/Portal	devtest-uc2-reg
Simulator	devtest-uc2-sim
VSE	devtest-uc2-vse

Following services on these virtual machines running Microsoft Windows Server need to be accessed from outside

- Windows Services

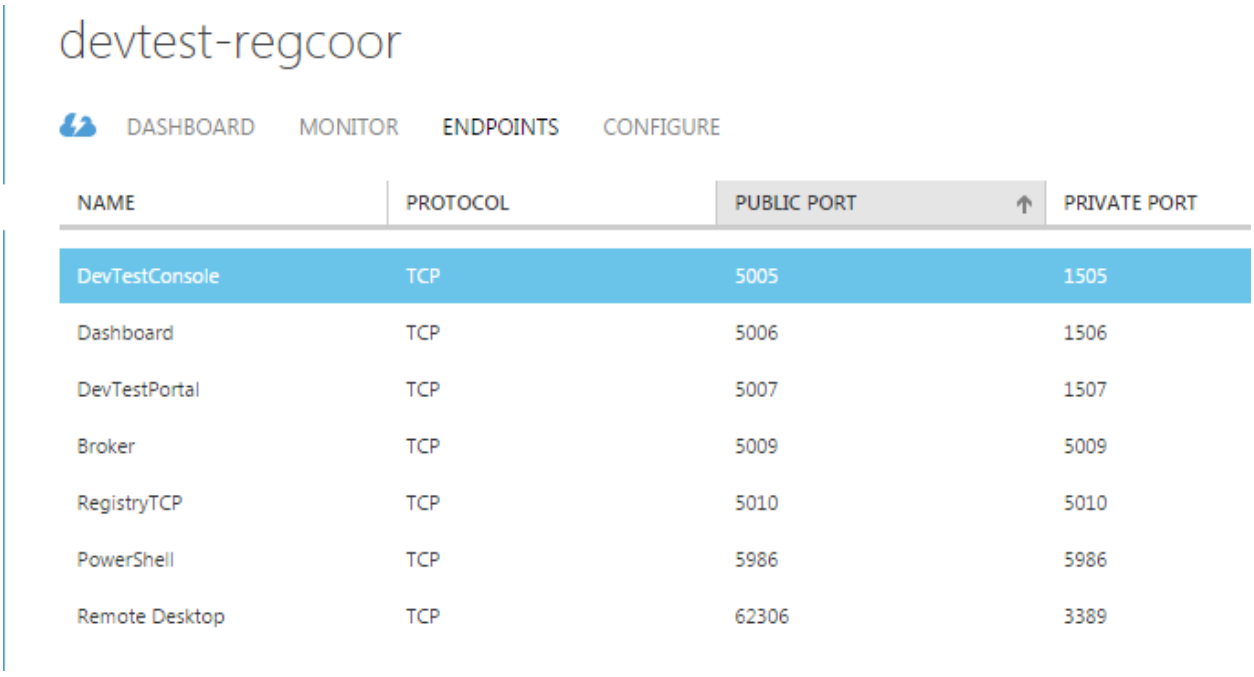
Service name	Default Port no.
Remote Desktop	3389
PowerShell	5986

- DevTest Services

Service name	Default Port no.	Modified Port no.
Registry	2010	5010
Portal	1507	None
Server Console	1505	None
Enterprise Dashboard	1506	None

When DevTest components are installed on VMs in the cloud they need to access the DevTest Enterprise Dashboard service. Therefore, outbound access to the Enterprise Dashboard service from MS Azure VMs to Enterprise Dashboard service on customer premise on default port 2003 must be available.

The following screenshots show the public port settings and mappings to private ports on the different VMs



devtest-uc2-sim

DASHBOARD MONITOR **ENDPOINTS** CONFIGURE

NAME	PROTOCOL	PUBLIC PORT ↑	PRIVATE PORT
Remote Desktop	TCP	54750	3389
PowerShell	TCP	55639	5986

devtest-uc2-vse

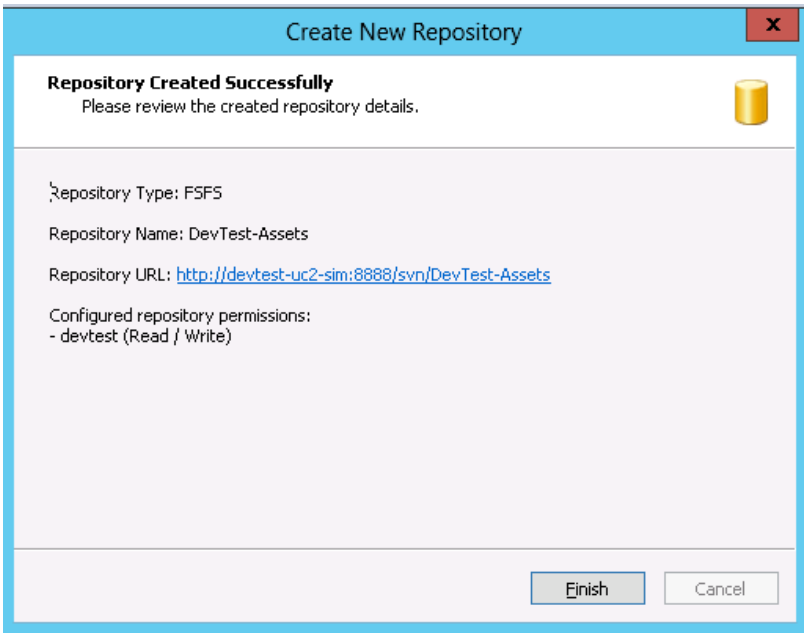
DASHBOARD MONITOR **ENDPOINTS** CONFIGURE

NAME	PROTOCOL	PUBLIC PORT ↑	PRIVATE PORT
Remote Desktop	TCP	57151	3389
PowerShell	TCP	62003	5986

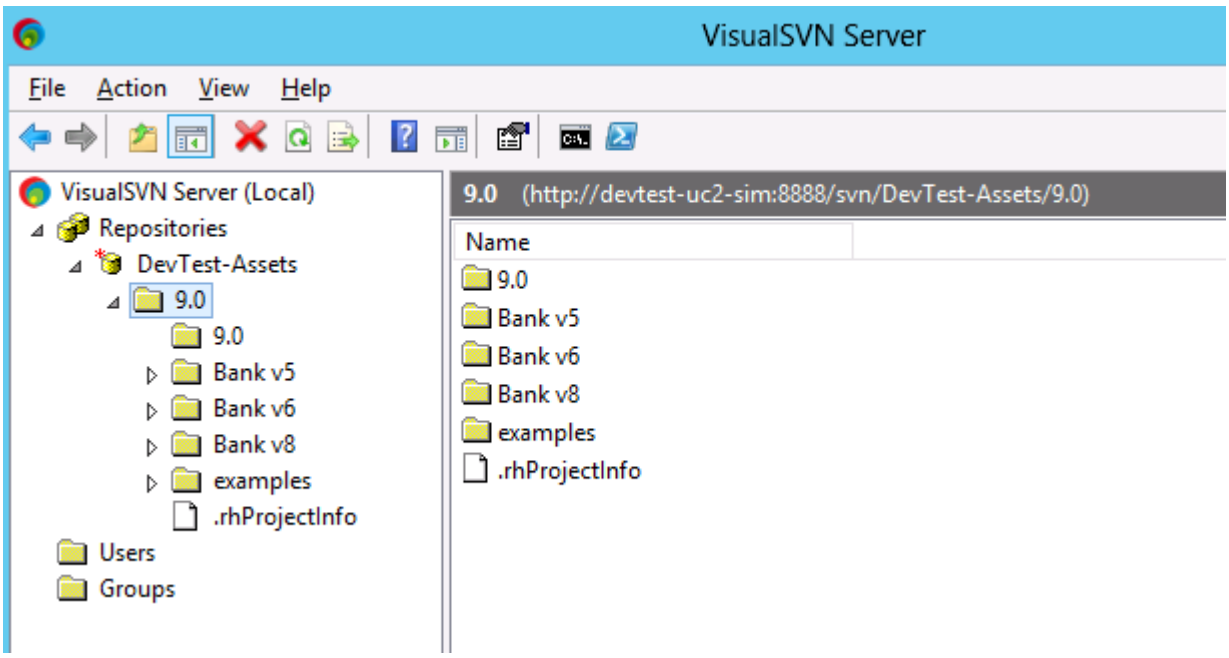
SCM Details

This section shows how to SCM is used –

- A new repository was created in Visual SVN Server called DevTest-Assets

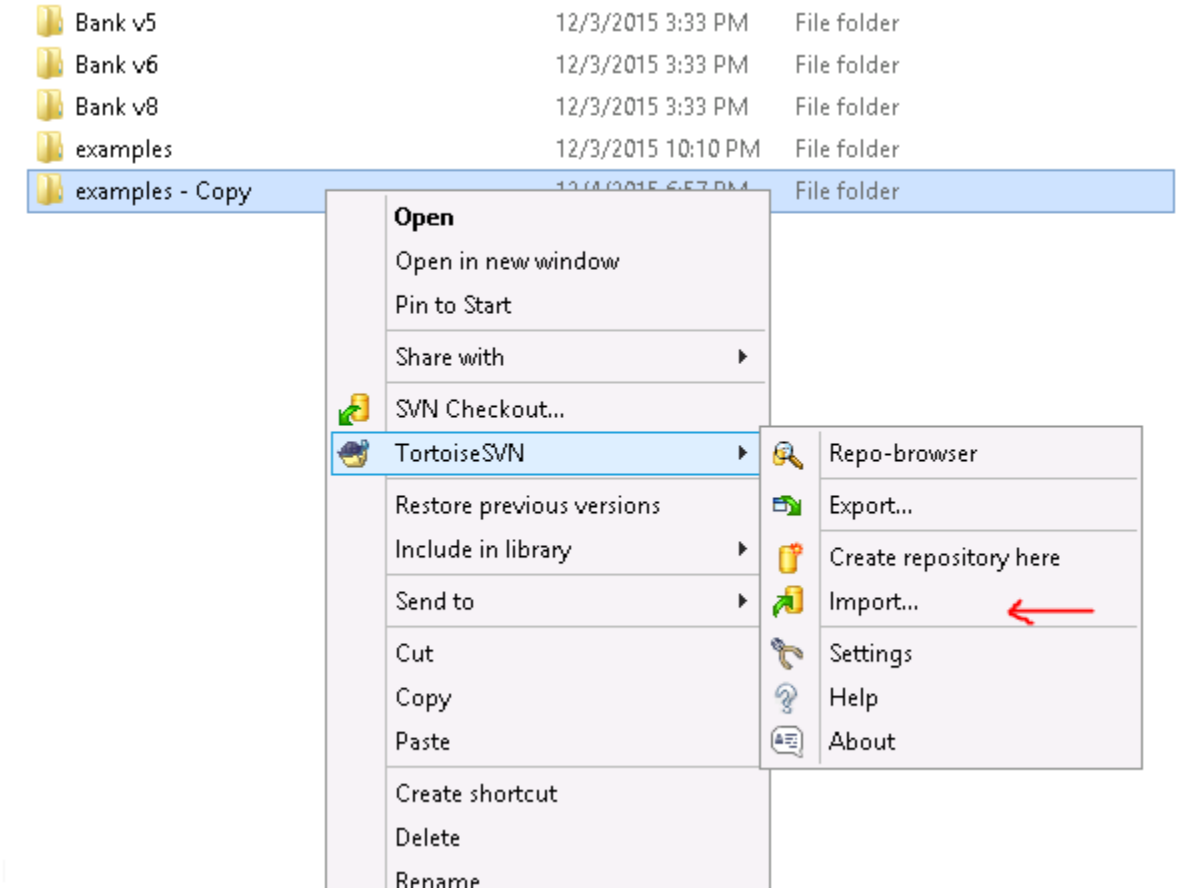


- From the location of the DevTest "Projects" folder, the Tortoise SVN client was used to import all the Project assets into SVN's DevTest-Assets repository

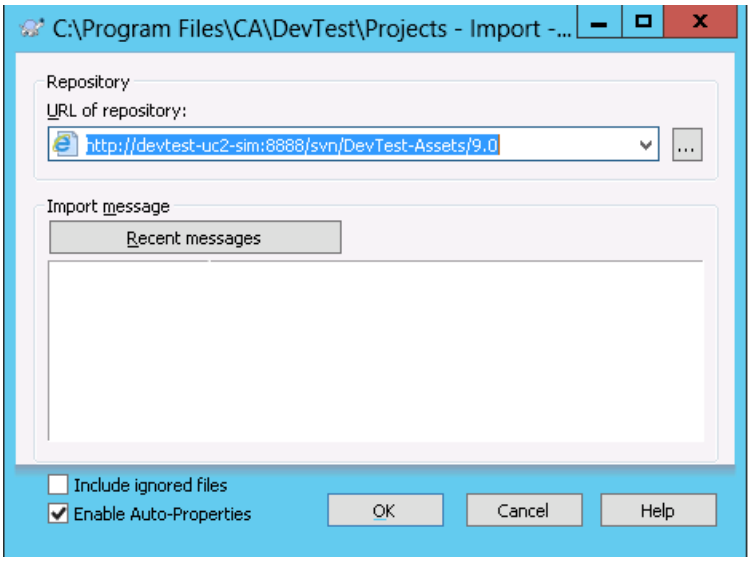


- At this point, the assumption is that the DevTest users have a mapped drive to the DevTest Projects folder. Once they create/update a new DevTest asset, they can simply commit those changes to the SVN using the Tortoise client.

- In this example, this end user would like to back up the ‘examples’ folder as ‘examples - copy’. After copy/paste, a new folder called ‘examples – copy’ is created.
- Then the user right clicks and proceeds to check in the new folder to SVN as shown below



Authenticate as required –



Completed check in to SVN server shown

A new folder is created in the SVN server for 'examples – copy'.

In this way, the same technique can be applied to any DevTest assets that are created/modified.

If you were using the DevTest Workstation and the assets were stored locally then you would check in and check out assets from the SVN server without the need to map to the centrally located DevTest Projects folder.

Setup Verification

This section covers steps to verify that the Azure setup is working correctly as expected. This includes steps to

- Determine that components are connected to the DevTest Registry service
- Verify that test cases can be launched from the DevTest Portal
- Verify that virtual services can be deployed from the DevTest Portal
- Check if tests can be monitored in DevTest Portal

- Check if virtual services show up in DevTest Portal's VSE

These verification steps are executed from the Enterprise Dashboard and the DevTest Portal to test the various access methods.

DevTest Portal (from within the company network)

DevTest Portal is used to access DevTest Server components:

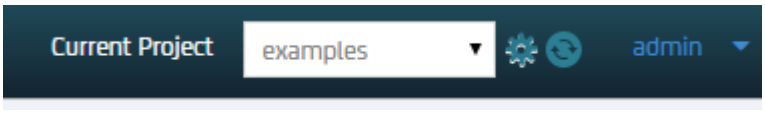
- Connectivity to required DevTest Server components such as Simulator, Coordinator and VSEs
- Staging a test to validate access to Coordinator and Simulator
- Deployment of a virtual service to verify access to VSE

Component Connectivity Check

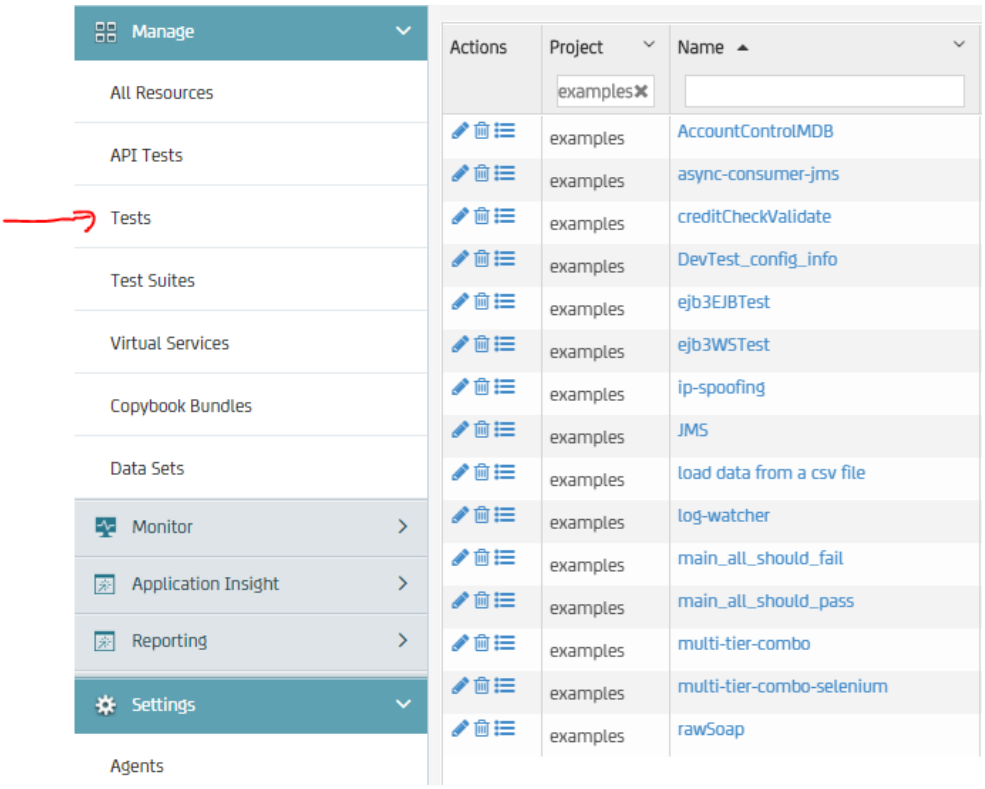
	Shows the Enterprise Dashboard with the following components running in the Azure environment • Registry • Coordinator • Simulator • VSE • Workstations

Running Tests from DevTest Portal

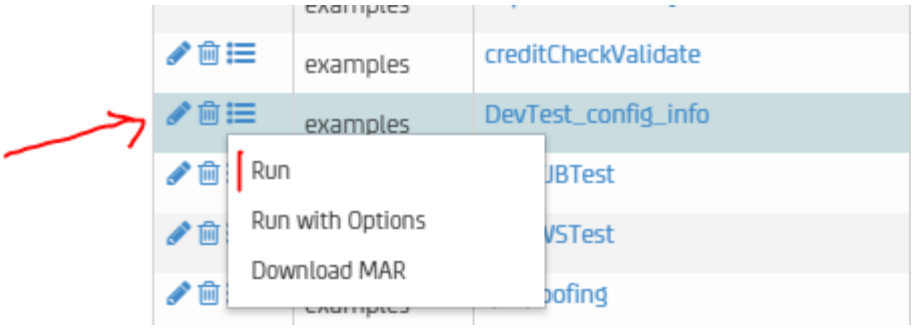
In the DevTest Portal, select the 'examples' folder. *Note – to view the 'examples' folder in the DevTest Portal, you would need to copy that folder to the Projects folder*.



In the left menu – select Manage → Tests



You will now locate and run the DevTest_config_info test case.

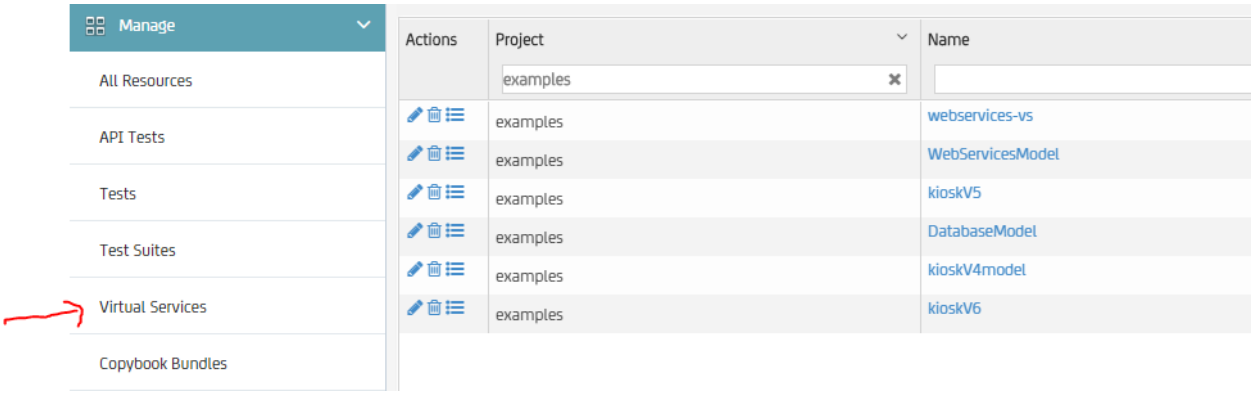


As soon as you click Run, a new tab called 'Monitoring Tests' is opened. The test should run to completion.

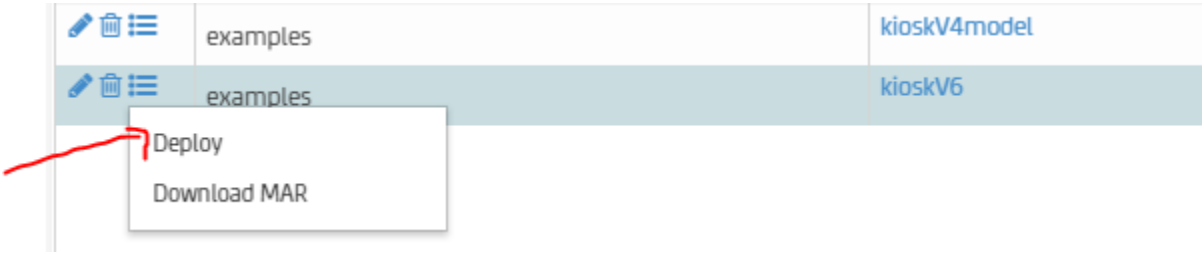
You can now click on the completed test case name to drill down into the details of the execution steps.

Deploy Virtual Service to VSE from the DevTest Portal

In the left menu, click on Manage → Virtual Services to open a list of available virtual services in the examples folder.



Deploy the virtual service called as kioskV6 by clicking on the Options action and selecting Deploy



Take the default for the VSE server and provide an optional Group tag. You should see a message about successful deployment.

Now click on Monitor →Virtual Service Environment → VSE. This will open a new tab called VSE and you should see the kioskV6 VS deployed and running successfully.