DevOps Kata

**Dev Test Labs**

Last updated: 12/05/2016



1. **TABLE OF CONTENT**

[Overview 3](#_Toc468690522)

[Prerequisites 3](#_Toc468690523)

[Exercises 3](#_Toc468690524)

[Exercise 1: Create a lab 4](#_Toc468690525)

[Task 1: Provision a lab 4](#_Toc468690526)

[Exercise 2: Add a VM with artifacts to a lab in Azure DevTest Labs 5](#_Toc468690527)

[Task 1: Add a VM with artifacts 5](#_Toc468690528)

[Task 2: Add an existing artifact to a VM 7](#_Toc468690529)

[Exercise 3: Use ARM 8](#_Toc468690530)

[Task 1: Save Azure Resource Manager template 9](#_Toc468690531)

## Overview

Whether your app is on-premises or in the cloud, you can automate build-deploy-test workflows and choose the technologies and frameworks, then test your changes continuously in a fast, scalable, and efficient manner. Continuous testing with Visual Studio Team Services or Team Foundation Server ensures your app still works after every check-in and build, enabling you to find problems earlier by running tests automatically with each build. Choose the test technologies and frameworks you prefer to use. When your build is done, review your test results to start resolving the problems you find. Rich and actionable build-on-build reports let you instantly see if your builds are getting healthier. But it's not just about speed - detailed and customizable test results measure the quality of your app.

To determine what proportion of your project’s code is actually being tested by coded tests such as unit tests, you can use the code coverage feature of Visual Studio. To guard effectively against bugs, your tests should exercise or ‘cover’ a large proportion of your code. Code coverage analysis can be applied to both managed (CLI) and unmanaged (native) code. Code coverage is an option when you run test methods using Test Explorer. The results table shows the percentage of the code that was run in each assembly, class, and method. In addition, the source editor shows you which code has been tested.

### Prerequisites

1. In order to complete the lab
2. 1. Log on to your Visual Studio / MSDN subscription and create or use your own VSTS instance. Alternatively, you can use a team sandbox VSTS instance if you have one.
3. 2. Have an azure subscription available that you can deploy resource to.
4. 3. Visual Studio Enterprise

### Exercises

* 1. This hands-on lab includes the following exercises:
  2. Create a dev test lab enviornment
  3. Create a VM and add artifacts
  4. Templatize the VM and place in a pipeline
  5. Estimated time to complete this exersize: **15 minutes**.

Exercise 1: Create a lab

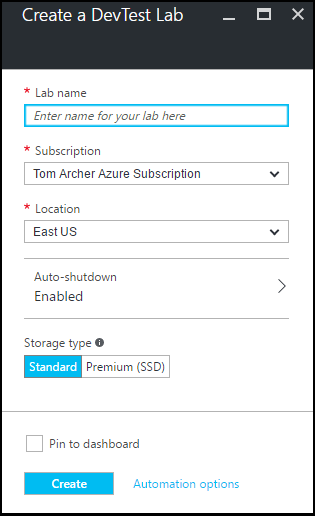
#### Task 1: Provision a lab

The following steps illustrate how to use the Azure portal to create a lab in Azure DevTest Labs. +

1. Sign in to the [Azure portal](http://go.microsoft.com/fwlink/p/?LinkID=525040).
2. Select **More services**, and then select **DevTest Labs** from the list.
3. On the **DevTest Labs** blade, select **Add**.

Add a lab

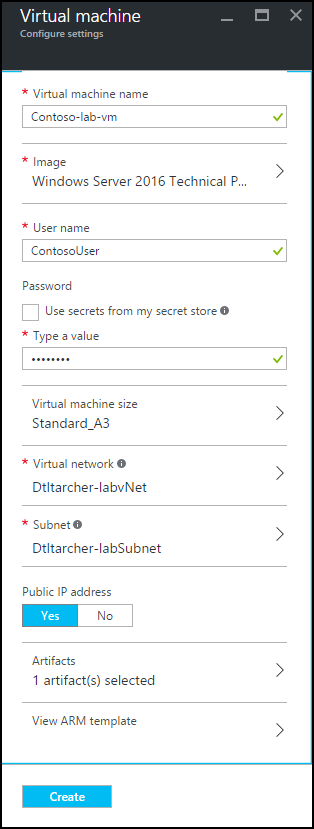
1. On the **Create a DevTest Lab** blade:
   1. Enter a **Lab Name** for the new lab.
   2. Select the **Subscription** to associate with the lab.
   3. Select a **Location** in which to store the lab.
   4. Select **Auto-shutdown** to specify if you want to enable - and define the parameters for - the automatic shutting down of all the lab's VMs.
   5. Select the **Storage type** to indicate the storage disk type for the lab's VMs.
   6. Select **Create**.



Exercise 2: Add a VM with artifacts to a lab in Azure DevTest Labs

#### Task 1: Add a VM with artifacts

1. Sign in to the [Azure portal](http://go.microsoft.com/fwlink/p/?LinkID=525040).
2. Select **More Services**, and then select **DevTest Labs** from the list.
3. From the list of labs, select the lab in which you want to create the VM.
4. On the lab's **Overview** blade, select **+ Virtual Machine**.  
   Add VM button
5. On the **Choose a base** blade, select a base for the VM.
6. On the **Virtual machine** blade, enter a name for the new virtual machine in the **Virtual machine name** text box.

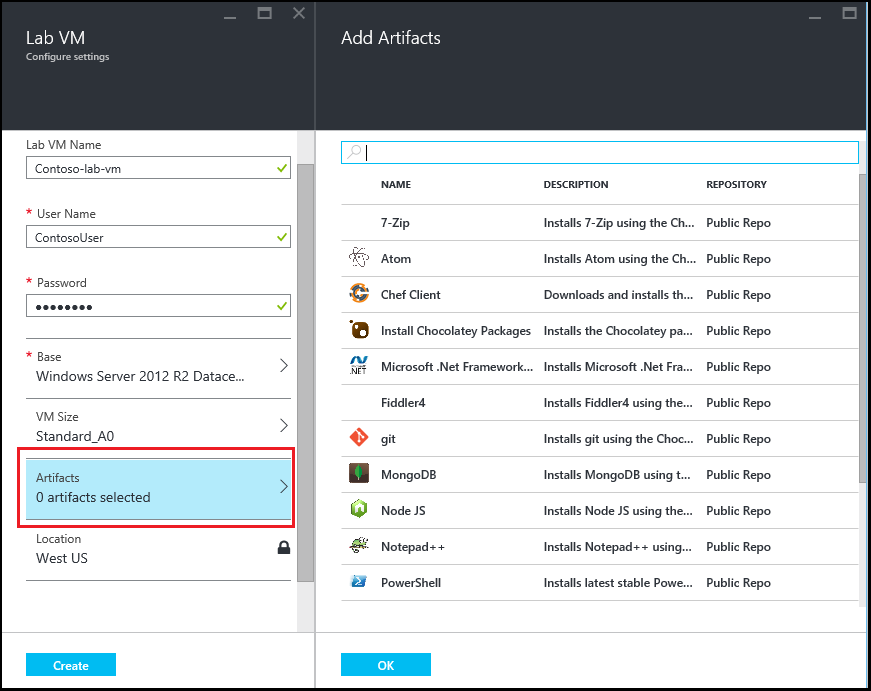


1. Enter a **User Name** that will be granted administrator privileges on the virtual machine.
2. If you want to use a password stored in your secret store, select **Use secrets from my secret store**, and specify a key value that corresponds to your secret (password). Otherwise, simply enter a password in the text field labeled **Type a value**.
3. Select **Virtual machine size** and select one of the predefined items that specify the processor cores, RAM size, and the hard drive size of the VM to create.
4. Select **Virtual network** and select the desired virtual network.
5. Select **Subnet** and select subnet.
6. If the lab policy is set to allow public IP addresses for the selected subnet, specify whether you want the IP address to be public by selecting either **Yes** or **No**. Otherwise, this option is disabled and selected as **No**.
7. Select **Artifacts** and - from the list of artifacts - select and configure the artifacts that you want to add to the base image. **Note:** If you're new to DevTest Labs or configuring artifacts, skip to the [Add an existing artifact to a VM](https://docs.microsoft.com/en-us/azure/devtest-lab/devtest-lab-add-vm-with-artifacts#add-an-existing-artifact-to-a-vm) section, and then return here when finished.
8. Select **Create** to add the specified VM to the lab.
9. The lab blade displays the status of the VM's creation; first as **Creating**, then as **Running** after the VM has been started.
10. Go to the [Next Steps](https://docs.microsoft.com/en-us/azure/devtest-lab/devtest-lab-add-vm-with-artifacts#next-steps) section.

Carry out the following steps to create a simple ASP.NET MVC app to deploy.

#### Task 2: Add an existing artifact to a VM

1. On the **Virtual machine** blade, select **Artifacts**.
2. On the **Add artifacts** blade, select the desired artifact.



1. Enter the required parameter values and any optional parameters that you need.
2. Select **Add** to add the artifact and return to the **Add Artifacts** blade.
3. Continue adding artifacts as needed for your VM.
4. Once you've added your artifacts, you can [change the order in which the artifacts are run](https://docs.microsoft.com/en-us/azure/devtest-lab/devtest-lab-add-vm-with-artifacts#change-the-order-in-which-artifacts-are-run). You can also go back to [view or modify an artifact](https://docs.microsoft.com/en-us/azure/devtest-lab/devtest-lab-add-vm-with-artifacts#view-or-modify-an-artifact).

Exercise 3: Use ARM

#### Task 1: Save Azure Resource Manager template

1. On the **Virtual machine** blade, select **View ARM Template**.
2. On the **View Azure Resource Manager Template blade**, select the template text.
3. Copy the selected text to the clipboard.
4. Select **OK** to close the **View Azure Resource Manager Template blade**.
5. Open a text editor.
6. Paste in the template text from the clipboard.
7. Save the file for later use.

Notes

* 1. Use <https://github.com/Azure/azure-quickstart-templates>
     1. This is baked into Visual Studio
     2. Most walkthrough are updated via this channel, so use the templates first!
  2. More advanced walkthrough
     1. <https://www.visualstudio.com/en-us/docs/release/examples/azure/provision-devtest-lab>
     2. <https://www.visualstudio.com/en-us/docs/release/examples/azure/net-to-azure-vm>