# Lab

This is the *PowerShell* lab.

# Lab objectives

Get familiar on how to use PowerShell through to manage ARM resources.

# Install pre-requisites

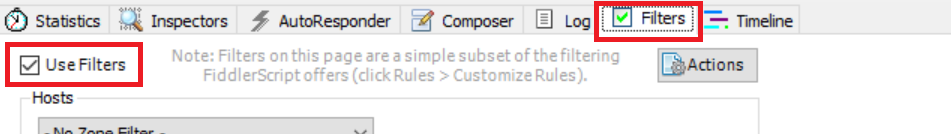
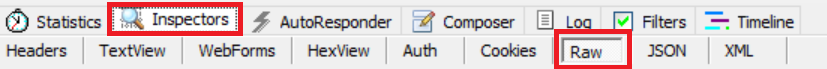
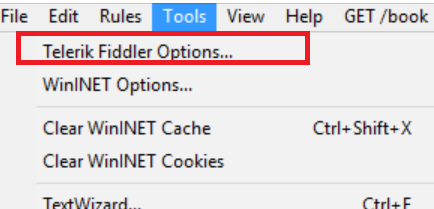
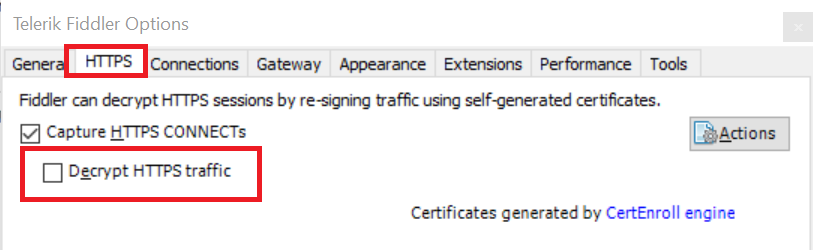
1. Install Microsoft Azure PowerShell <https://azure.microsoft.com/en-us/downloads/>

# Login

1. Open up PowerShell ISE
2. Type Add-AzureRmAccount
3. Enter your credentials ; those credentials should be the same you are using to log into the Azure Portal
4. If you have more than one subscriptions
   1. Type Get-AzureRmSubscription
   2. This should list all subscriptions you have access (even partial) to
   3. Select the *SubscriptionId* (a GUID) of the subscription you want to use
   4. Type Select-AzureRmSubscription -SubscriptionId <SubscriptionId>  
      *<SubscriptionId>* is the value you just selected
   5. This will select the specified subscription as the “current one”, i.e. future queries will be done against that subscription

# Fiddler (optional)

This section is optional. It consists into starting fiddler (can be downloaded at <http://www.telerik.com/fiddler>) to trace web queries and visualize the REST APIs called.

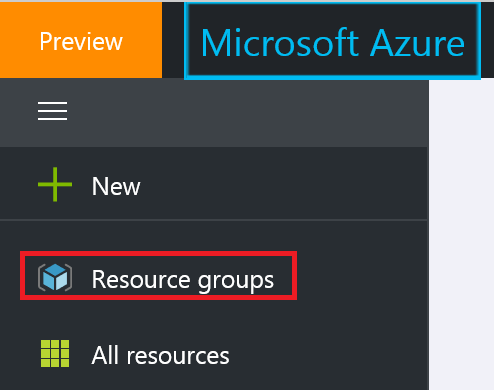
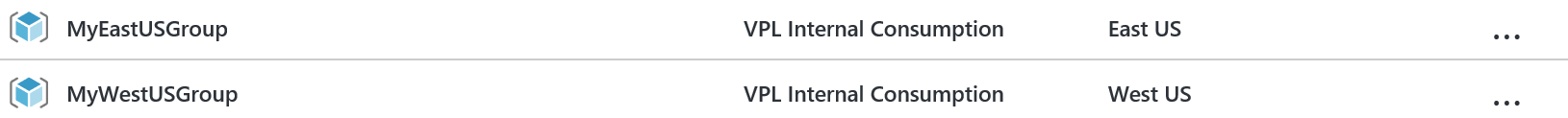
1. Start Fiddler
2. Go to the *Filters* tab  
   
3. Select *Use Filters* check box
4. In the *Request Headers* of the same tab
   1. Select *Show only if URL contains* check box  
      
   2. Type *management.azure.com/subscriptions* in the adjacent text box
   3. This will limit the number of queries captured by Fiddler and make it easier for you to spot the requests that interest you
5. Select the *Inspectors* tab  
   
6. Select the *Raw* tab underneath, this is the request pane ; you’ll be able to do this only when the first request will be in
7. Select the *Raw* tab in the pane underneath, this is the response pane
8. You can also select JSON as Azure ARM REST APIs are done in JSON
9. Ensure HTTPS traffic is decrypted
   1. Select the *Tools* menu in the menu bar
   2. Select *Telerik Fiddler Options…* in the sub menu  
      
   3. In the dialog box, select the HTTPS tab
   4. Check the *Decrypt HTTPS traffic* checkbox  
      
   5. Click ok

For each request, notice the HTTP header “Authorization” passing a *bearer* token in the request.

# List ARM Resource Providers

1. In PowerShell ISE, type Get-AzureRmResourceProvider
2. This will list all the ARM providers that can be accessed through the REST API
3. Resource Provider are organized within namespaces
4. Type Get-AzureRmResourceProvider -ProviderNamespace Microsoft.Compute
5. This will list all the resource providers under the *Microsoft.Compute* namespace

# Create Resource Groups

1. In ISE, type New-AzureRmResourceGroup -Name MyEastUSGroup -Location eastus
2. Then type New-AzureRmResourceGroup -Name MyWestUSGroup -Location westus
3. Open the Azure portal: <https://portal.azure.com/>
4. Navigate to your resource groups  
   
5. In the list, you should find the two resource group we just created  
   
6. Notice the two resource groups have been created in the specified regions

# PowerShell objects

PowerShell on ARM REST API behaves the same way as any other PowerShell scripts, i.e. it manipulates PowerShell (i.e. .NET) objects. You can therefore do what you would do in PowerShell.

1. In ISE, type Get-AzureRmResourceGroup
2. This will list all the resource groups in your subscription
3. Type Get-AzureRmResourceGroup | Where-Object {$\_.ResourceId.Contains("My")}
4. This should list only the resource groups having “My” in their name
5. Type Get-AzureRmResourceGroup | Where-Object {$\_.ResourceId.Contains("My")} | Select-Object ResourceGroupName, Location
6. This should list the same groups but outputting only their name & location
7. Type (Get-AzureRmResourceGroup).GetType().Name
8. This should give you an array, i.e. *object[]*
9. Type (Get-AzureRmResourceGroup)[0].GetType().FullName
10. This should give you *Microsoft.Azure.Commands.Resources.Models.PSResourceGroup*, the type of the first element in the list of resource groups

# Notes

Have a look at <https://azure.microsoft.com/en-us/documentation/articles/resource-manager-supported-services/> for the schema of different resources.

# Exercise

Try to list the resource groups by location. Hint: think about using the Group-Object cmdlet.

# Clean up

We won’t be using the resource groups we have created so we can delete them

1. In ISE, type Remove-AzureRmResourceGroup -Name MyEastUSGroup
2. You will be prompted by a dialog box to confirm, click *Yes*
3. Type Remove-AzureRmResourceGroup -Name MyWestUSGroup -Force
4. This will not prompt you
5. If you refresh the resource group list in the portal, you should see both groups have disappeared