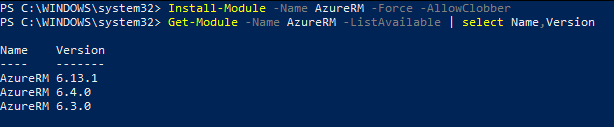
Install Powershell

**$PSVersionTable.PSVersion**

**install –Module –Name AzureRM –Force –AllowClobber**

**Get-Module -Name AzureRM -ListAvailable | select Name,Version**

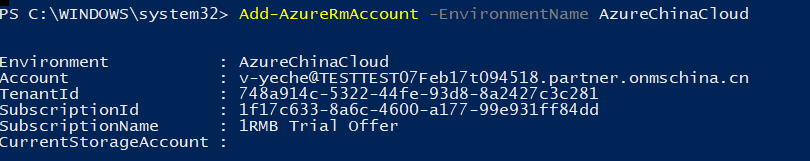
****

**Import-Module -Name AzureRM -RequiredVersion 6.13.1**

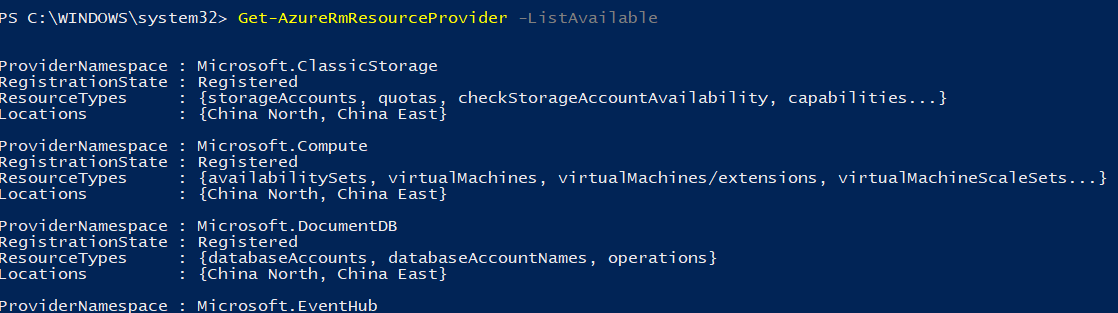
**Add-AzureRmAccount -EnvironmentName AzureChinaCloud**

|  |  |
| --- | --- |
| Account | [v-yeche@TESTTEST07Feb17t094518.partner.onmschina.cn](mailto:v-yeche@TESTTEST07Feb17t094518.partner.onmschina.cn) |
| Password | [Ryan=Lindsayc5](mailto:abc!@123D) |

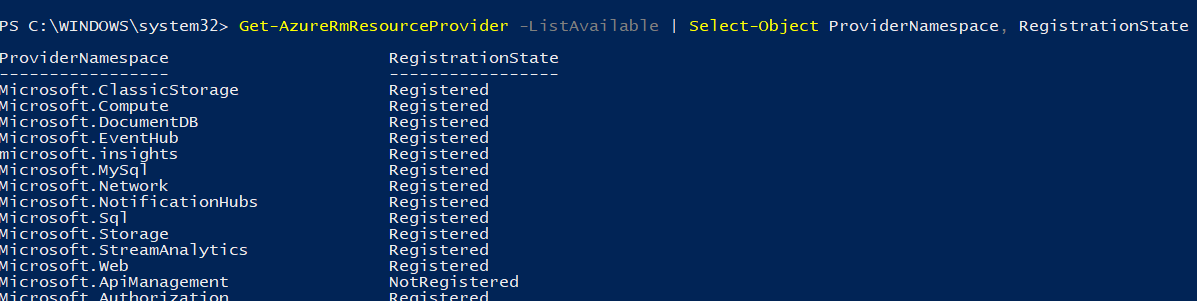
**az provider list --query "[].{Provider:namespace, Status:registrationState}" --out table**

****

**Get-AzureRmResourceProvider -ListAvailable**

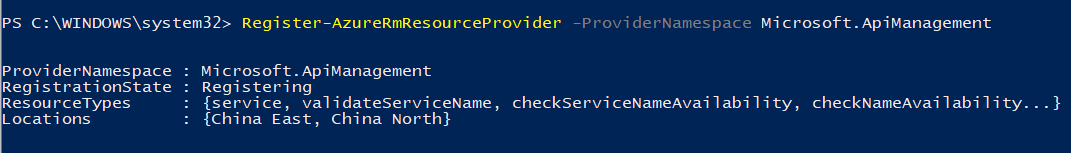
****

**Get-AzureRmResourceProvider -ListAvailable | Select-Object ProviderNamespace, RegistrationState**

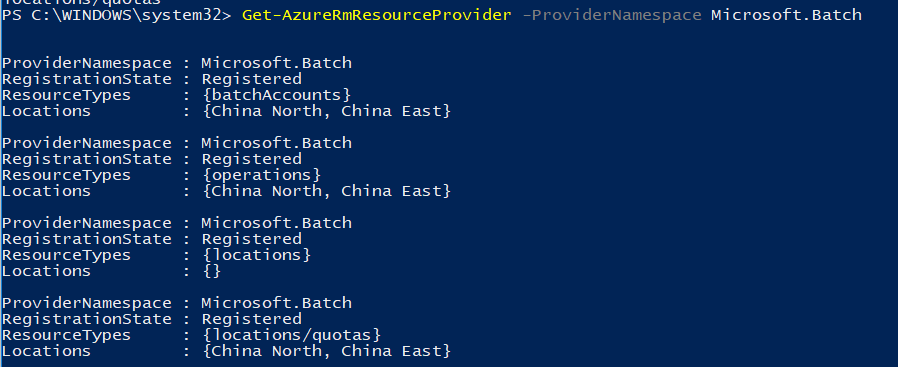


**Register-AzureRmResourceProvider -ProviderNamespace Microsoft.ApiManagement**

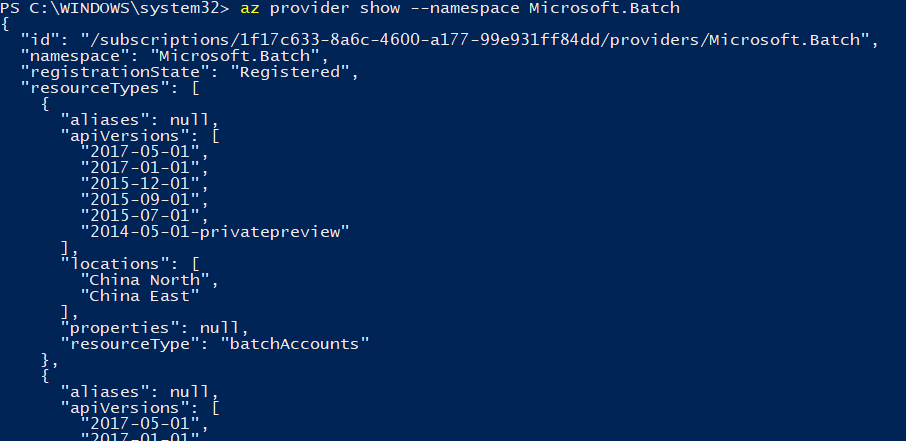
**az provider register --namespace Microsoft.Batch**

****

**Get-AzureRmResourceProvider -ProviderNamespace Microsoft.Batch**

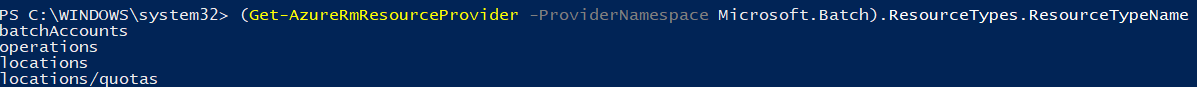
****

**az provider show --namespace Microsoft.Batch**

****

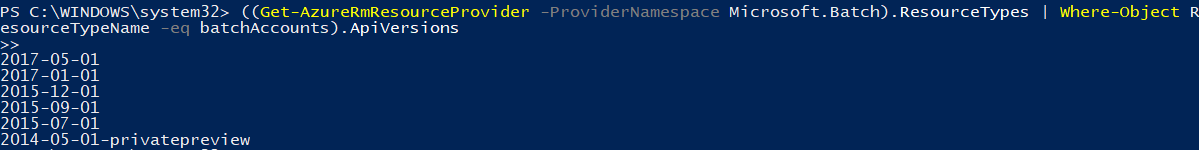
**(Get-AzureRmResourceProvider -ProviderNamespace Microsoft.Batch).ResourceTypes.ResourceTypeName**

**az provider show --namespace Microsoft.Batch --query "resourceTypes[\*].resourceType" --out table**

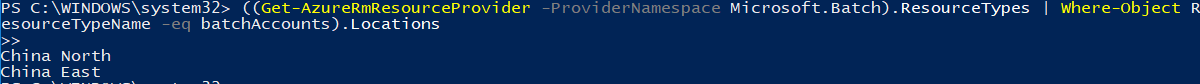
****

**((Get-AzureRmResourceProvider -ProviderNamespace Microsoft.Batch).ResourceTypes | Where-Object ResourceTypeName -eq batchAccounts).ApiVersions**

**az provider show --namespace Microsoft.Batch --query "resourceTypes[?resourceType=='batchAccounts'].apiVersions | [0]" --out table**

****

**((Get-AzureRmResourceProvider -ProviderNamespace Microsoft.Batch).ResourceTypes | Where-Object ResourceTypeName -eq batchAccounts).Locations**

****

Install-Module -Name PowerShellGet -MinimumVersion 2.0.1

Install-Module -Name PowerShellGet -MinimumVersion 2.0.1 –Force -AllowClobber

**Install-Module Microsoft.ServiceFabric.Powershell.Http –AllowPrerelease –Force -AllowClobber –Scope CurrentUser**

The first time I ran into this error I was stumped for while finding a solution. Ultimately it comes down to using Self-Signed Certificates in vCenter, as most of us do.  In general using Invoke-WebRequest or Invoke-RestMethod against a server using a Self-Signed Certificate will cause this error, it’s not just related to vCenter.

The solution is quite simple.  I found a snippet of code some time back that I keep on hand in this situation.  It basically ignores certificate validate in PowerShell allowing you to make a connection with Invoke-WebRequest.  All you have to do it paste this code into your PowerShell session before you run Invoke-WebRequest against a server with a Self-Signed Certificate.

**if** (-not ([System.Management.Automation.PSTypeName]'ServerCertificateValidationCallback').Type)

{

$certCallback = **@"**

**using System;**

**using System.Net;**

**using System.Net.Security;**

**using System.Security.Cryptography.X509Certificates;**

**public class ServerCertificateValidationCallback**

**{**

**public static void Ignore()**

**{**

**if(ServicePointManager.ServerCertificateValidationCallback ==null)**

**{**

**ServicePointManager.ServerCertificateValidationCallback +=**

**delegate**

**(**

**Object obj,**

**X509Certificate certificate,**

**X509Chain chain,**

**SslPolicyErrors errors**

**)**

**{**

**return true;**

**};**

**}**

**}**

**}**

**"@**

    Add-Type $certCallback

 }

[ServerCertificateValidationCallback]::Ignore()