Set up the Development Environment

To prepare a development environment that supports your work in Azure and on your local PC.

* Operating System: Windows 10 will be used as the OS. Windows is used by most of your team, so it was a logical choice. You make a note to the team that Azure services support other operating systems (such as Mac OS and Linux), and that Microsoft provides supporting documentation for the members of your team who choose one of these alternatives.
* General Coding Tools: Visual Studio Code and Azure CLI will be used as the primary coding tools.

In support of these decisions, you will be setting up the following environment:

* Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later). Including
  + 4GB – 8GB system RAM (higher the better for Docker)
  + Hyper-V and Containers features of Windows must be enabled.
  + BIOS-level hardware virtualization support must be enabled in the BIOS settings.

**Note**: When setting up the development environment on a virtual machine, the VM environment must support nested virtualization - https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/user-guide/nested-virtualization

* Azure CLI (current/latest)
* .NET Core 3.1.200 (or later) SDK
* VS Code (latest)
* Azure Extensions for VS Code and Azure CLI

**Note**: A virtual machine has been created for this course that provides a majority of the tools specified above. The instructions below support using the prepared VM or setting up the development environment locally using your PC.

Steps

1: Install Developer Tools and Products

**Important**: The tools and products associated with this Exercise are pre-installed on the virtual machine created for this course. Before continuing, check with your course Instructor to understand if you will be completing labs using the hosted lab VM environment or setting up the dev environment locally on your PC.

**Task 1:***Install .NET Core*

.NET Core is a cross-platform version of .NET for building websites, services, and console apps.

1. To open the .NET Core download page, use the following link: https://dotnet.microsoft.com/download
2. On the .NET download page, under .NET Core, click **Download .NET Core SDK**.

The .NET Core SDK is used to build .NET Core apps. You will be using it to build/edit code files during the labs in this course.

1. On the popup menu, click **Run**, and then follow the on-screen instructions to complete the installation.

The installation should take less than a minute to complete. The following components will be installed:

* + .NET Core SDK 3.1.100 or later
  + .NET Core Runtime 3.1.100 or later
  + ASP.NET Core Runtime 3.1.100 or later
  + .NET Core Windows Desktop Runtime 3.1.0 or later

The following resources are available for further information:

* + https://aka.ms/dotnet-docs
  + https://docs.microsoft.com/en-us/dotnet/core/install/dependencies?tabs=netcore31&pivots=os-windows
  + https://aka.ms/dotnet-sdk-docs
  + https://aka.ms/netcore3releasenotes
  + https://aka.ms/dotnet-tutorials

**Task 2:***Install Visual Studio Code*

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and run times (such as .NET and Unity).

1. To open the Visual Studio Code download page, click the following link: https://code.visualstudio.com/Download

Instructions for installing Visual Studio Code on Mac OS X and Linux can be found on the Visual Studio Code set up guide https://code.visualstudio.com/docs/setup/setup-overview. This page also includes more detailed instructions and tips for the Windows installation.

1. On the Download Visual Studio Code page, click **Windows**.

When you start the download, two things will happen: a popup dialog opens and some getting started guidance will be displayed.

1. On the popup dialog, to begin the setup process, click **Run** and then follow the on-screen instructions.

If you choose to Save the installer to your Downloads folder, you you can complete the installation by opening the folder and then double-clicking the VSCodeSetup executable.

By default, Visual Studio Code is installed in the "C:\Program Files (x86)\Microsoft VS Code" folder location (for a 64-bit machine). The set up process should only take about a minute.

**Note**: .NET Framework 4.5 is required for Visual Studio Code when installing on Windows. If you are using Windows 7, please ensure https://www.microsoft.com/en-us/download/details.aspx?id=30653 is installed.

For detailed instructions on installing Visual Studio Code, see the Microsoft Visual Studio Code Installation Instruction guide here: https://code.visualstudio.com/Docs/editor/setup

**Task 3:***Install Azure CLI*

Azure CLI is a command-line tool that is designed to make scripting Azure-related tasks easier. It also enables you to flexibly query data, and it supports long-running operations as non-blocking processes.

1. Open your browser, and then navigate to the Azure CLI tools download page: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest "Azure CLI Install"

You should be installing the latest version of the Azure CLI tools (currently version 2.4). If version 2.4 is not the latest version listed on this "azure-cli-latest" download page, install the more recent version.

1. On the **Install Azure CLI** page, select the install option for your OS (such as **Install on Windows**), and then follow the on-screen instructions to install the Azure CLI tool.

You will be given detailed instructions for using the Azure CLI tools during the labs in this course, but if you want more information now, see https://docs.microsoft.com/en-us/cli/azure/get-started-with-azure-cli?view=azure-cli-latest

2: Install Dev Tool Extensions

The Visual Studio Code and Azure CLI tools both support an Azure IoT extension that helps developers to create their solutions more efficiently. These extensions leverage the Azure IoT SDKs and will often reduce development time while ensuring security provisions. You will also be adding a C# extension for Visual Studio Code.

**Task 1:***Install Visual Studio Code Extensions*

1. Open Visual Studio Code.
2. On the left side of the Visual Studio Code window, click **Extensions**.

You can hover the mouse pointer over the buttons to display the button titles. The Extensions button is sixth from the top.

1. In the Visual Studio Code Extension manager, search for and then install the following Extensions:
   * [Azure Tools for Visual Studio Code](https://marketplace.visualstudio.com/items?itemName=ms-vscode.vscode-node-azure-pack) (ms-vscode.vscode-node-azure-pack) by Microsoft
2. Close Visual Studio Code.

Your development environment should now be set up!

3: Set Up Course Lab Files and Alternative Tools

A number of the labs in this course rely on pre-built resources, such as a code project that can be used as a starting point for the lab activity. These lab resources are provided in a GitHub project that you should download to your dev environment.

In addition to the resources that directly support the course labs (the resources contained in the GitHub project), there are some optional tools that you may choose to install because they support learning opportunities outside of this course. One example is PowerShell, which you may see referenced within Microsoft tutorials and other resources.

The instructions below lead you through the configuration of both these resource types.

**Task 1:***Download Course Lab Files*

Microsoft has created a GitHub repo to provide access to lab resource files. Having these files local to the dev environment is required in some cases and convenient in many others. In this task you will be downloading and extracting the contents of the repo within your development environment.

1. In your Web browser, navigate to the following location: <https://github.com/MicrosoftLearning/AZ-204-DevelopingSolutionsforMicrosoftAzure>
2. On the right side of the page, click **Clone or download**, and then click **Download ZIP**.
3. To save the ZIP file to your dev environment, click **Save**.
4. Once the file has been saved, click **Open folder**.
5. Right-click the saved ZIP file, and then click **Extract all**
6. Click **Browse**, and then navigate to folder location that is convenient to access.

By default, Windows has a https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file#maximum-path-length-limitation. As the file paths within the ZIP are already long, avoid extracting the archive within nested folders with a large file path. For example, the default path prompted to extract the zip could be similar to **c:\users\[username]\downloads\** **AZ-204-DevelopingSolutionsforMicrosoftAzure-master** - it is recommended that you shorten this as much as possible, to perhaps **c:\az204**

1. To extract the files, click **Extract**.

Be sure to make note of where you located the files.

**Task 2:***Install Azure PowerShell Module*

**Note**: The lab activities in this course do NOT include using PowerShell, however, you may see sample code in reference documents that use PowerShell. If you want to run PowerShell code, you can use the following instructions to complete the installation steps.

Azure PowerShell is a set of cmdlets for managing Azure resources directly from the PowerShell command line. Azure PowerShell is designed to make it easy to learn and get started with, but provides powerful features for automation. Written in .NET Standard, Azure PowerShell works with PowerShell 5.1 on Windows, and PowerShell 6.x and higher on all platforms.

**Warning**: You can't have both the AzureRM and Az modules installed for PowerShell 5.1 for Windows at the same time. If you need to keep AzureRM available on your system, install the Az module for PowerShell Core 6.x or later. To do this, install PowerShell Core 6.x or later and then follow these instructions in a PowerShell Core terminal.

1. Decide if you wish to install the Azure PowerShell module for just the current user (recommended approach) or for all users.
2. Launch the PowerShell terminal of your choice - if you are installing for all users you must launch an elevated PowerShell session either by either selecting **run as administrator** or with the **sudo** command on macOS or Linux.
3. To only install for the current user, enter the following command:

powershell

Install-Module -Name Az -AllowClobber -Scope CurrentUser

or to install for all users on a system, enter the following command:

powershell

Install-Module -Name Az -AllowClobber -Scope AllUsers

1. By default, the PowerShell gallery isn't configured as a trusted repository for PowerShellGet. The first time you use the PSGallery you see the following prompt:

output

Untrusted repository

You are installing the modules from an untrusted repository. If you trust this repository, change

its InstallationPolicy value by running the Set-PSRepository cmdlet.

Are you sure you want to install the modules from 'PSGallery'?

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"):

1. Answer **Yes** or **Yes to All** to continue with the installation.

The Az module is a rollup module for the Azure PowerShell cmdlets. Installing it downloads all of the available Azure Resource Manager modules, and makes their cmdlets available for use.

**Note**: If the **Az** module is already installed, you can update to the latest version using:

powershell

Update-Module -Name Az

3.Register resource providers

Many different types of resources will be created during this course, some of which may not have been register for use in the current subscription. While some resources are registered automatically during the first use, others must be registered before they can be used, otherwise errors will be reported.

Task 1 - Register resource providers using the Azure CLI

The Azure CLI provides a number of commands to help manage resource providers. In this task, you will ensure that the resource providers required for this course are registered.

1. Using a browser, open the https://shell.azure.com/ and login with the Azure subscription you are using for this course.
2. To view a list of the current state of the resource providers, enter the following command:

powershell

az provider list -o table

This will display a *long* list of resources, similar to:

powershell

Namespace RegistrationPolicy RegistrationState

--------------------------------------- -------------------- -------------------

Microsoft.OperationalInsights RegistrationRequired Registered

microsoft.insights RegistrationRequired NotRegistered

Microsoft.DataLakeStore RegistrationRequired Registered

Microsoft.DataLakeAnalytics RegistrationRequired Registered

Microsoft.Web RegistrationRequired Registered

Microsoft.ContainerRegistry RegistrationRequired Registered

Microsoft.ResourceHealth RegistrationRequired Registered

Microsoft.BotService RegistrationRequired Registered

Microsoft.Search RegistrationRequired Registered

Microsoft.EventGrid RegistrationRequired Registered

Microsoft.SignalRService RegistrationRequired Registered

Microsoft.VSOnline RegistrationRequired Registered

Microsoft.Sql RegistrationRequired Registered

Microsoft.ContainerService RegistrationRequired Registered

Microsoft.ManagedIdentity RegistrationRequired Registered

...

1. To see return a list of the namespaces that contains the string **Event**, run the following command:

powershell

az provider list -o table --query "[?contains(namespace, 'Event')]"

The results will be similar to:

powershell

Namespace RegistrationState RegistrationPolicy

------------------- ------------------- --------------------

Microsoft.EventGrid NotRegistered RegistrationRequired

Microsoft.EventHub Registered RegistrationRequired

1. To register the resources required for this course, execute the following commands:

powershell

az provider register --namespace "Microsoft.EventGrid" --accept-terms

az provider register --namespace "Microsoft.EventHub" --accept-terms

az provider register --namespace "Microsoft.Insights" --accept-terms

az provider register --namespace "Microsoft.TimeSeriesInsights" --accept-terms

**NOTE**: You may see a warning that **-accept-terms** is in preview - you can ignore this.

**NOTE**: The **microsoft.insights** is listed in lowercase - however the register/unregister commands are case-insensitive.

1. To view the updated status of the resources, execute the following commands:

powershell

az provider list -o table --query "[?(contains(namespace, 'insight') || contains(namespace, 'Event') || contains(namespace, 'TimeSeriesInsights'))]"

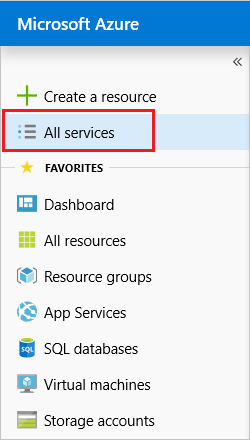
**NOTE**: Although the register/unregister commands are case-insensitive, the query language is not, so **insight** must be lowercase.

The resources should now be registered.

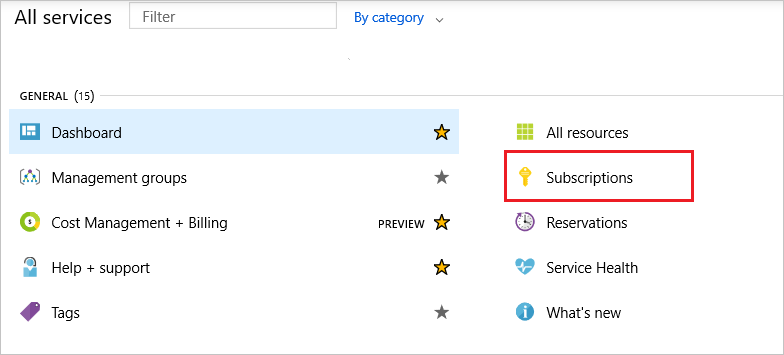
Task 2 - Register resource providers using the Azure Portal

You can see the registration status and register a resource provider namespace through the portal. In this task, you will familiarize yourself with the UI.

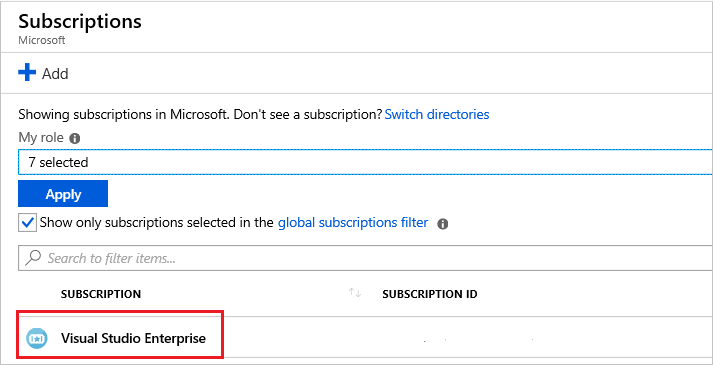
1. If necessary, log in to https://portal.azure.com using your Azure account credentials.
2. From the portal, select **All services**.



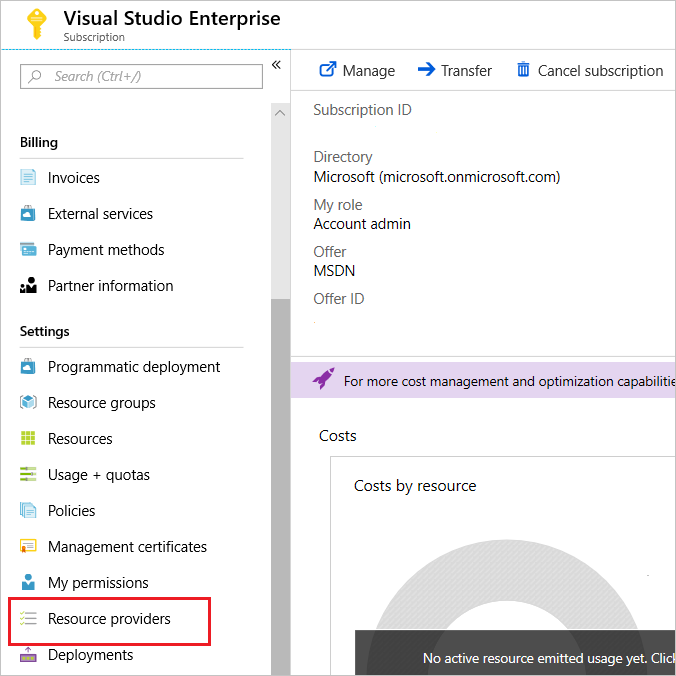
1. Select Subscriptions.



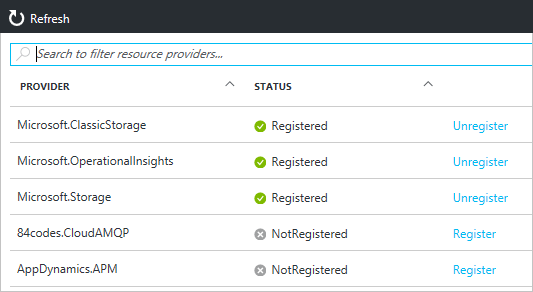
1. From the list of subscriptions, select the subscription you want to use for registering the resource provider.



1. For your subscription, select **Resource providers**.



1. Look at the list of resource providers, resources can be be registered or unregistered by clicking the appropriate action.



1. To filter the listed resources, in the search textbox, enter **insights**.

Notice that the list is filtered as search criteria is entered. The search is also case-insensitive.

Congratulations!

You have successfully completed this lab. Click **Next** to advance to the next lab.