**Azure Dev Environment Setup**

The purpose of this document is to set up an Azure development environment. This includes setting up a Python environment with initial installs, installing VS Code and its extensions, working with GitHub, creating a Python virtual environment using conda, and working with Azure Functions and bindings, such as adding an Event Hub Trigger and a CosmosDB output binding to an Azure Function.

1. **Python Environment Setup: Initial Installs**
2. Install VS Code. Download can be found here: <https://code.visualstudio.com/download>
3. Once VS Code is installed we will install the following extensions: **Python, Azure Functions, Azure CLI Tools, GIT Hub Pull Requests and Issues, Azure Tools**. Select the extensions icon from the activity board.

A screenshot of a computer

Description automatically generated with medium confidence

1. Graphical user interface, text

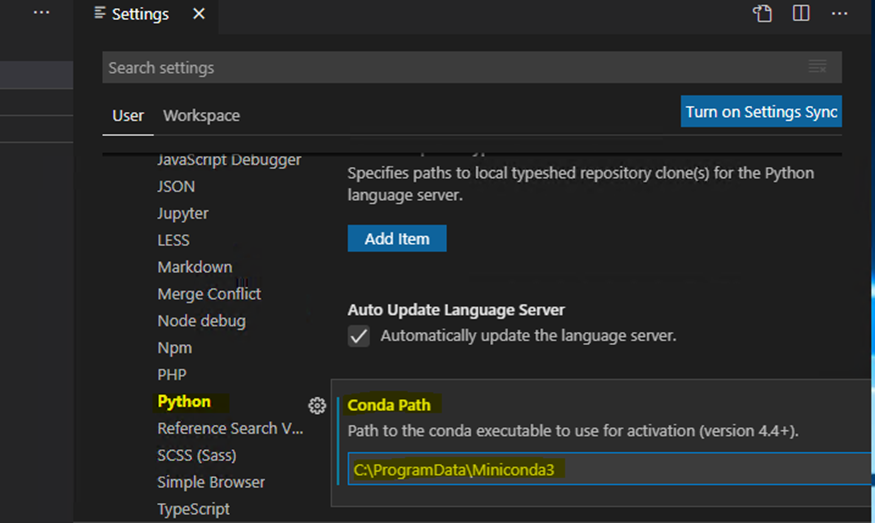
   Description automatically generatedSearch for Python. Select “Install” if not already there.
2. Repeat the steps for all the other extensions needed. Please note a reload of VSCode may be needed after the ext installs.
3. Install Azure CLI. This will be used to create and manage Azure resources. Download can be found here: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>
4. **Working with GitHub: GitHub allows you to share your source code and collaborate with others.**
5. Once the GitHub extension has been installed. You will need to sign in. Follow the prompts to authenticate with GitHub.
6. Select the Accounts Icon from the bottom left inside the activity board. Sign into GitHub account.

A screenshot of a computer

Description automatically generated

1. Once signed in you can manage your repositories as described here: <https://code.visualstudio.com/docs/editor/github>
2. Note that you will need GIT installed on your desktop. You can find install guide here: <https://github.com/git-guides/install-git>
3. **Create Python Virtual Environment using conda:**
4. Start by downloading latest Python. Download can be found here: <https://www.python.org/downloads/>
5. Also download MiniConda with Python 3.9. Download can be found here: <https://docs.conda.io/en/latest/miniconda.html#windows-installers>
6. Once installed you may manually specify the to the conda executable path in VSCode.
   * Open the command palette in VSCode (ctrl +shift+p)
   * A screenshot of a computer

     Description automatically generatedEnter **Preferences: Open User Settings** and hit enter.
   * Under the extensions->python find the “Conda Path” settings. Set with the appropriate path.



1. Open a New Terminal Window. Select New Command prompt form the drop down list within terminal.

Text

Description automatically generated

1. From the terminal window run the following command: **conda create --name myenv**
2. This will display the environment location path. Type Y to proceed.

Text

Description automatically generated

1. From here the environment is created and you may activate the environment by running the following command: **conda activate myenv**

Text

Description automatically generated

1. **Function Apps:**

You may either create a new project or sign into your azure account to use existing Azure Function. You may follow the steps from the link below to create a function.

<https://docs.microsoft.com/en-us/azure/azure-functions/create-first-function-vs-code-python>

1. **Adding Event Hub Trigger to Azure function:**

A binding is added to the bindings array in your function.json file.

1. From VSCode navigate to the Azure Icon within the activity bar on the left.
2. Once there you should see the Functions tab. Click on tab to expand.
3. From the function menu select the “Create Function” lightning icon. Search for the Azure Event Hub trigger using the search bar.

A screenshot of a computer

Description automatically generated with medium confidence

1. Provide a name of the trigger function. Type name within search bar

Graphical user interface, text, application, email

Description automatically generated

1. Select local app setting location or create a new location. (supplies the “Values” array of application settings and connection strings used when a project is running locally)
2. Provide a name of the event hub. Type name within search bar

Graphical user interface, text, application, email

Description automatically generated

1. Provide consumer group name. (A consumer group is a view (state, position, or offset) of an entire event hub. Consumer groups enable multiple consuming applications to each have a separate view of the event stream, and to read the stream independently at their own pace and with their own offsets.) You can also select “$default”. You can create up to 20 consumer groups for a standard tier.

Graphical user interface, text, application, email

Description automatically generated

1. The function should now be created and appear under the functions menu.

More info here: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-develop-vs-code?tabs=python>

1. **Adding CosmosDB output binding**

To add a CosmosDB output binding follow the steps under the “Add an output binding” section of this document link below.

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-integrate-store-unstructured-data-cosmosdb?tabs=javascript>