# Getting Started with fabric\_devops

This guide walks you through the basic stapes of getting up and running with **fabric\_devops** project. This will allow you to write, test and debug Python code which automates CI/CD tasks in Fabric-based solutions using the Fabric REST APIs. You will begin by configuring your developer environment so you run Python code locally on your computer in the Visual Studio Code debugger. After that, you will move through the configuration steps to enable the same Python code to run in the cloud within the context of a GitHub workflow.

This guide will lead you through the following steps to get you started with local development on your developer workstation.

1. Fork the **fabric\_devops** repository into your own repository
2. Clone the forked repository to a local **fabric\_devops** folder on your developer workstation
3. Open the local copy of the **fabric\_devops** folder in Visual Studio Code
4. Update configuration data in the **.env** file for local development and debugging
5. Run a set of demo Python scripts to experiment with sample Python code in the **fabric\_devops** project

After you are able to run and test the sample code in the **fabric\_devops** project locally, this guide will lead you through the following steps so you can run and test the code within the context of GitHub workflow actions.

1. Enable workflows in your forked copy of the **fabric\_devops** repository
2. Update configuration data by adding GitHub action secrets
3. Experiment by running the sample GitHub actions that are part of the **fabric\_devops** project

## Fork the fabric\_devops repository

Begin by forking the **fabric\_devops** repository.

1. Navigate to the repository in the browser at <https://github.com/FabricDevCamp/fabric-devops>.
2. Drop down the **Fork** menu in the upper right of the page.
3. Select the **Create a new fork** command.

A screenshot of a computer

AI-generated content may be incorrect.

1. Next, select the **Create fork** command.

A screenshot of a computer

AI-generated content may be incorrect.

This forked repository is your starting point. This is the repository in which you will test and experiment with Python code

The next step is to clone this repository to a local folder for testing in Visual Studio Code

A screenshot of a computer

AI-generated content may be incorrect.

Use git clone command or equivalent to copy files from repo to local project folder.

A screenshot of a computer

AI-generated content may be incorrect.

Open the project in Visual Studio Code.

A screenshot of a computer

AI-generated content may be incorrect.

## Updating Configuration in the Local .env File

Steps to add configuration for local debugging

Rename the sample.env file to .env

Update environmental variables in the .env file

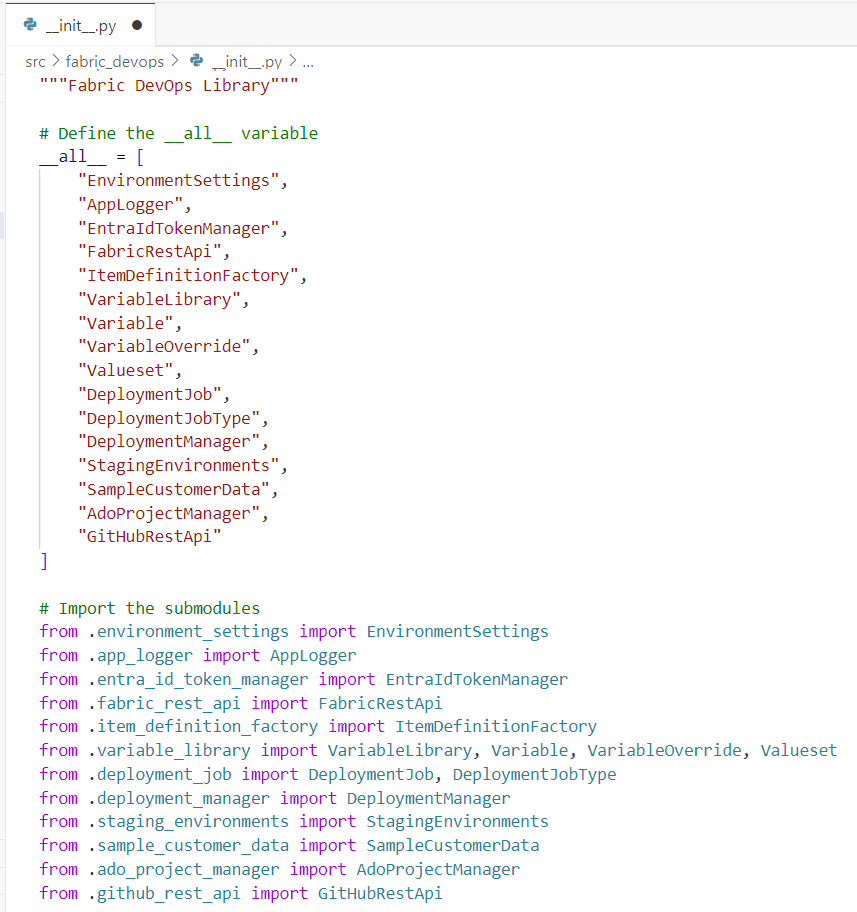
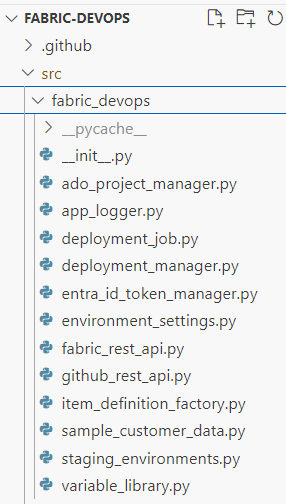
A screenshot of a computer

AI-generated content may be incorrect.

Here is what you need to edit.

* Configuration for Entra Id application used to authenticate as service principal
  + **FABRIC\_CLIENT\_ID**
  + **FABRIC\_CLIENT\_SECRET**
  + **FABRIC\_TENANT\_ID**
  + **SERVICE\_PRINCIPAL\_OBJECT\_ID**
* The Azure object Id for your Entra Id user account
  + **ADMIN\_USER\_ID**
* Capacity Id
  + **FABRIC\_CAPACITY\_ID**
* Personal access token used to create connections between Fabric workspace and GitHub
  + **PERSONAL\_ACCESS\_TOKEN\_GITHUB**

Sss



**fabric\_devops** Library exposes set of classes

* + **EnvironmentSettings**
  + **AppLogger**
  + **EntraIdTokenManager**
  + **FabricRestApi**
  + **ItemDefinitionFactory**
  + **VariableLibrary**
  + **DeploymentManager**
  + **StagingEnvironments**
  + **SampleCustomerData**
  + **AdoProjectManager**
  + **GitHubRestApi**

## EnvironmentSettings Class

EnvironmentSettings class provides access to environmental variables

Environmental variables loaded using os.getenv function

When running locally in Visual Studio Code, environmental variables loaded from .env file

When running locally in GitHub workflows, environmental variables loaded action secrets

A screenshot of a computer screen

AI-generated content may be incorrect.

## Authenticating with MSAL

Authentication and token acquisition in the fabric\_devops project is implemented using Microsoft Authentication Library (MSAL) for Python

* MSAL used to authenticate with Entra Id and acquire access tokens
  + MSAL provides **ConfidentialClientApplication** used acquire access tokens for service principals
  + MSAL provides **PublicClientApplication** used acquire access tokens for service principals
* Installing MSAL
  + Library installed from Python Package Index (PyPI) at **https://pypi.org/project/msal/**
  + Install command from terminal in Visual Studio Code: **pip install msal**
* Permission scopes for requesting access tokens for Fabric REST APIs
  + For service principal tokens: **https://api.fabric.microsoft.com/.default**
  + For user tokens: **https://api.fabric.microsoft.com/user\_impersonation**

## Running Demo Scripts

Once configuration Is set, you can run demo scripts

A screenshot of a computer

AI-generated content may be incorrect.

### Demo 01 - Acquire Access Token for Service Principal

Sssss

A screenshot of a computer program

AI-generated content may be incorrect.

### Demo 02 - Acquire Access Token for Interactive User

Xxxxx

A screenshot of a computer screen

AI-generated content may be incorrect.

### Demo 03 - Acquire Access Token for User with Device Code

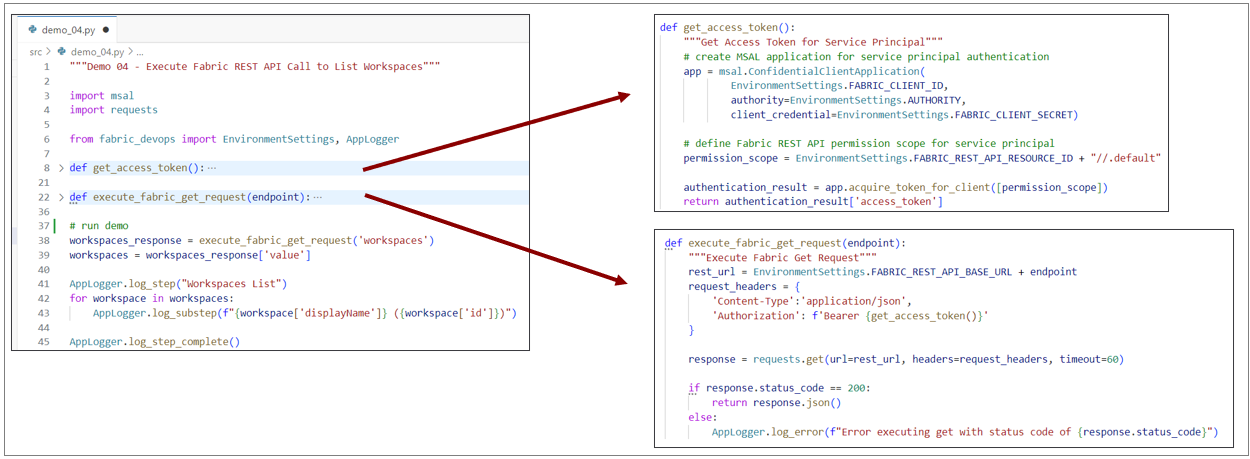
Xxxxxx

A screenshot of a computer program

AI-generated content may be incorrect.

### Demo 04 - Execute Fabric REST API Call to List Workspaces

Xxxx



Ssss

A screenshot of a computer program

AI-generated content may be incorrect.

### Demo 05 - Execute Fabric REST API Call to List Capacities You Need a Fabric Capacity Id for Testing.

### A screenshot of a computer program AI-generated content may be incorrect.

Copy the capacity ID of a capacity based on F SKU, FT SKU or P SKU

A screenshot of a computer

AI-generated content may be incorrect.

### Demo 06 - Execute Fabric REST API to Create Workspace

Xxxxx

A screenshot of a computer program

AI-generated content may be incorrect.

Handling Longing Running Operations in POST Requests

A screenshot of a computer program

AI-generated content may be incorrect.

### FabricRestApi class

A screenshot of a computer

AI-generated content may be incorrect.

Xxx

### Demo 07 - Create Workspace using FabricRestApi Class xxx

A screenshot of a computer program

AI-generated content may be incorrect.

### Demo 08 - Deploy Complete Fabric Solution

ssss

### Coming Soon: The Fabric REST API Python SDK

Python SDK for Fabric REST API adds valuable productivity boost

* + Abstracts away executing HTTP requests, adding **Authorization** header and handling responses
  + Parses together target REST URLs containing workspace Id, item Ids and specific endpoints
  + Handles serializing/deserializing JSON payloads sent back and forth over the wire
  + Provides automatic support for API calls which implement **long-running operations (LRO)** pattern
  + Provides automatic support for dealing with **continuation tokens** and merging **paginated results**

Inline\_code python

def deploy\_fabcon\_solution(

        target\_workspace,

        deploy\_job = StagingEnvironments.get\_dev\_environment()):

    """Deploy FabCon Solution"""

    bronze\_lakehouse\_name = "sales\_bronze"

    silver\_lakehouse\_name = "sales\_silver"

    gold\_warehouse\_name = "sales"

Inline\_code