## General

- The host is currently going through a redesign for v2 which moves it onto .NET core and move languages out into their own separate repos. These guidelines are generally written for the work going on in `dev` which is v2. The `master` branch and a few others are still `v1` and have a different workflow.

- The host has some dependencies on having some Azure resources provisioned, so before you get started, make sure you have reliable access to an Azure subscription. E2E tests require a lot of various services to be provisioned. If you're making large contributions which would affect E2E tests, it'll be expected that you can provision those services yourself.

- Be nice :) Everyone is busy and sometimes it can take a bit to get responses on things. Just be patient and, if you do poke someone for help, please be courteous and respectful.

## Questions & Help

If you have questions about Azure Functions, we encourage you to reach out to the community and Azure Functions dev team for help.

- For all questions and technical help, [our Q&A forums](https://docs.microsoft.com/en-us/answers/questions/topics/single/25345.html) are an easy place to have a conversation with our engineering team.

- For questions which fit the Stack Overflow format ("\*how\* does this work?"), we monitor the [azure-functions](http://stackoverflow.com/questions/tagged/azure-functions) tag.

- You can also tweet/follow [@AzureFunctions](https://twitter.com/azurefunctions).

While we do our best to help out in a timely basis, we don't have any promise around the above resources. If you need an SLA on support from us, it's recommended you invest in an [Azure Support plan](https://azure.microsoft.com/en-us/support/options/).

## Issues & feature requests

We track functional issues in a variety of places for Azure Functions. If you have found an issue or have a feature request, please submit an issue to the below repositories.

|Item|Description|Link|

|----|-----|-----|

|Documentation|Docs for Azure Functions features + getting started|[File an Issue](https://github.com/azure/azure-functions/issues)|

|Runtime|Script Host, Triggers & Bindings, Language Support|[File an Issue](https://github.com/Azure/azure-webjobs-sdk-script/issues)|

|Core Tools|Command line interface for local development|[File an Issue](https://github.com/Azure/azure-functions-cli/issues)|

|Dev Tools|Visual Studio and VS Code|[File an Issue](https://github.com/Azure/azure-functions/issues)|

|Portal|User Interface or Experience Issue|[File an Issue](https://github.com/ProjectKudu/AzureFunctionsPortal/issues)|

|Templates|Code Issues with Creation Template|[File an Issue](https://github.com/Azure/azure-webjobs-sdk-templates/issues)|

Before filing an issue, please check that it doesn't already exist. If you're not sure if you should file an issue, you can open up a [Q&A forum question](https://docs.microsoft.com/en-us/answers/questions/topics/single/25345.html). We also have a [uservoice feedback site](https://feedback.azure.com/forums/355860-azure-functions) which we can track your feature requests through.

## Pre-reqs for developing

- OS

- Windows 10 (suggested)

- Mac OS X/Linux (not-recommended for now)

- While you can develop from a Mac/Linux machine, it can be a rough experience and not all unit tests pass today. We have improvements where we hope to make this easier.

- Language runtimes

- Note: today you have to have Node.js and Java installed, but in the long run we hope move those tests out into their own repos

- [Java 8 ](http://www.oracle.com/technetwork/java/javase/downloads/index.html) (JDK and JRE required)

- [Node 8.4+](https://nodejs.org/en/)

- [.NET Core 2.2 SDK](https://dotnet.microsoft.com/download/dotnet-core/2.2)

- Editor

- [Visual Studio 2019](https://visualstudio.microsoft.com/vs/) (recommended)

- [VS Code](https://code.visualstudio.com/) (works, but has some quirks running tests)

- Misc tools (suggested)

- [git](https://git-scm.com/downloads) - source control

- [nvm (nvm-windows for windows)](https://github.com/coreybutler/nvm-windows) - Node Version Manager (for managing multiple versions of Node.js)

- Commander/ConEmu(Windows)/iTerm(mac) - console managers; makes dealing with lots of consoles more manageable

- [functions core tools](https://www.npmjs.com/package/azure-functions-core-tools) - helps for making samples/etc. `npm i -g azure-functions-core-tools@core`

## Running locally (Visual Studio) (v2+)

First thing you'll want to try is to run the Function host locally.

1. Set the WebJobs.Script.WebHost to the startup project

2. Change the debug configuration to "WebJobs.Script.WebHost.Core"

3. Add `AzureWebJobsScriptRoot` setting pointing at your test project

- You can add this variable a few ways:

1. Add a Environment Variable via a global variable (have to restart VS afterwards)

2. Add a setting to appsettings.json (be careful not to check it in)

3. Add an environment variable via the Debug configuration for the project (be careful not to check it in)

- You can create a simple hello world function app via the function core tools CLI. In a sample directory just run:

```

func init

func new -l javascript -t httptrigger -n hello

echo $PWD

```

The output of $PWD is your current directory, use that full path

4. Click debug - this should launch a new terminal and browser window. If you created a new http triggered function named "hello", you should be able to add "api/hello" to your base URL and see your Function's response and the `context.log` statement in the terminal.

If you want to test anything other than HTTP, it will require the `AzureWebJobsStorage` and `AzureWebJobsDashboard` settings get set to an Azure Storage Account connection string. You'll also need to add settings for any non-storage services you might want to connect to. You can do this via the same 3 methods described above.

## Running tests (Visual Studio)

There are three test projects in the WebJobs.Script solution

- WebJobs.Script.Tests

- WebJobs.Script.Test.Integration

The only thing you need to set up for the tests is your Storage Account for the integration tests. You need to set the `AzureWebJobsStorage` and `AzureWebJobsDashboard` settings for WebJobs.Script.Tests.Integration project. The appsettings.json method is pretty clean, but you need to create it. Be sure "CopyToOutput" is true and rebuild afterwards for the appsettings.json file. To run end to end (E2E) tests, see the set up requirements for Samples under [Home](https://github.com/Azure/azure-webjobs-sdk-script/wiki).

Then open up for your test explorer (CTRL+E, T) and click Run All. If any fail, you might have set something up wrong locally.

## Change flow

The general flow for making a change to the script host is:

1. ? Fork the repo (add the fork via `git remote add me <clone url here>`

2. ? Create a branch for your change (generally use dev) (`git checkout -b my-change`)

3. ? Make your change

4. ?? Test your changes

5. ?? Push your changes to your fork (`git push me my-change`)

6. ? Open a PR to the dev branch

7. ? Address feedback and make sure tests pass (yes even if it's an "unrelated" test failure)

8. ? [Squash](https://git-scm.com/docs/git-rebase) your changes into a meaningful commits (usually 1 commit) (`git rebase -i HEAD~N` where `N` is commits you want to squash)

9. :shipit: Rebase and merge (This will be done for you if you don't have contributor access)

10. ?? Delete your branch (optional)

## Getting help

- Leave comments on your PR and @ people for attention

- [@AzureFunctions](https://twitter.com/AzureFunctions) on twitter

- (MSFT Internal only) Functions Dev teams channel & email