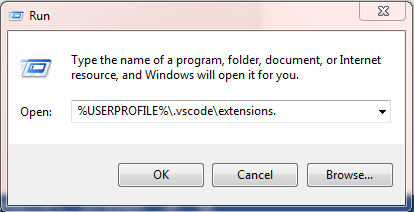
Openwisk extension

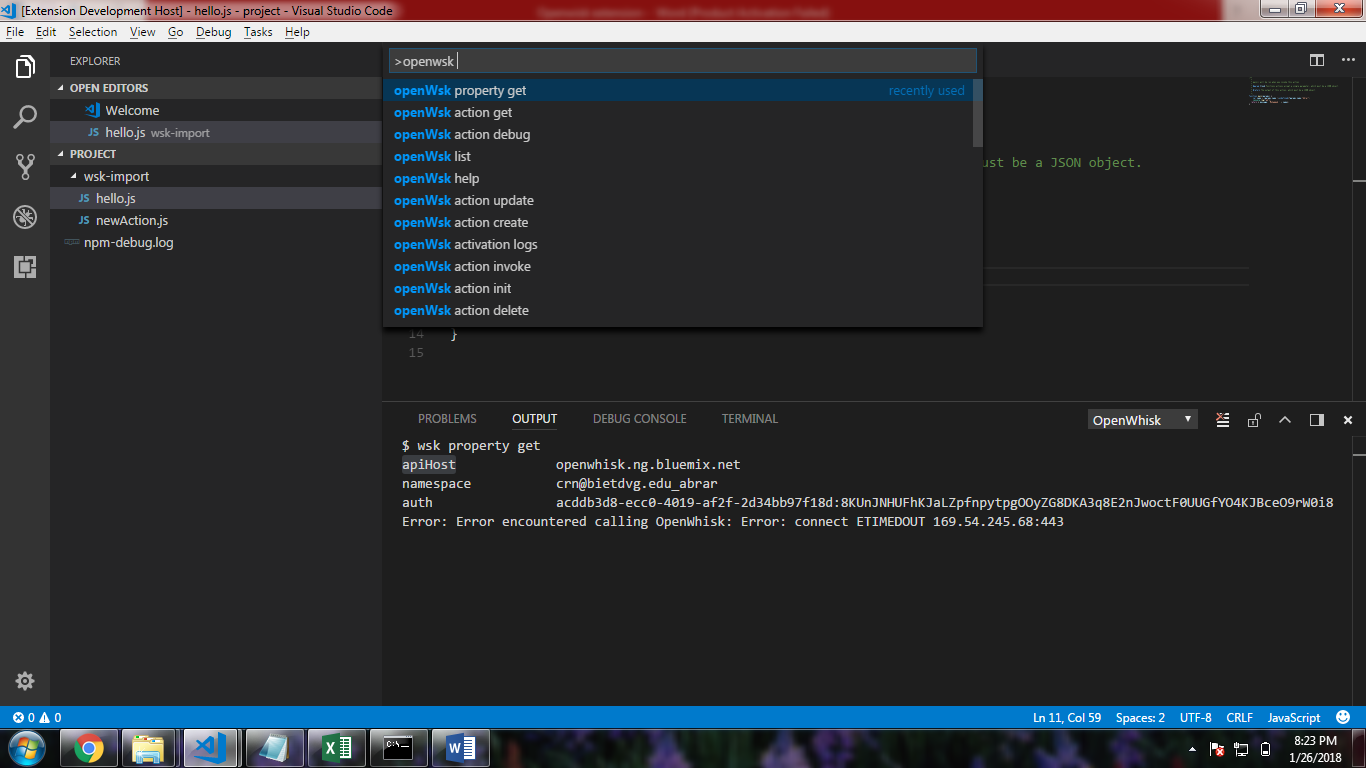
**Steps to add extension to vs code:**

1. Extract the folder.
2. Press ctrl+R on keyboard and enter %USERPROFILE%\.vscode\extensions. It will open vs code extension folder.



Mac ~/.vscode/extensions Linux ~/.vscode/extensions

1. Copy and paste the extracted folder in to the vs code extension folder.
2. Open Vs code it will detect extension changes click ok.
3. Copy the apihost, Namespace and auth api key from bluemix openwisk instance.
4. Press ctrl + Shift + P in vs code. Which will display commands of openwisk service in autosuggest.



1. Select openwsk property set for authorization and set apihost, namespace and auth.
2. Now you will be able to call other commands from the vs code.

# Local Debugger

**Steps to setup local debugger:**

There are basically two ways to setup the debugger in vscode.

1. Function call from global function/file.

2. Function call from the same file.

**1. Function call from global function/file:**

1. Create a project or a folder and name it as openwsk-debugger.

2. Create a MainFun.js file on the root directory of the project with the following code snippet. Which will run the action configured in launch.json

const actionFileName = './wks-import/hello.js'

const inputFileName = './inputParams/hello.json'

MainFun = () => {

let openWhiskFunction = require((process.argv[2] || actionFileName))

let payload = require((process.argv[3] || inputFileName))

const mainFunction = openWhiskFunction.main ? openWhiskFunction.main : fallback(actionFileName);

let promise = mainFunction(payload)

if (promise.then) {

promise.then(

function (response) {

console.log(JSON.stringify(response, null, 3))

})

.catch(

function (reason) {

console.error(JSON.stringify(reason, null, 3))

return {}

});

}

else {

console.log(JSON.stringify(promise, null, 3))

}

}

function fallback(action) {

eval(require("fs").readFileSync(action, "utf-8"));

if (main) {

return main;

} else {

console.error(action + " has no function main or no exports.main");

process.exit(1);

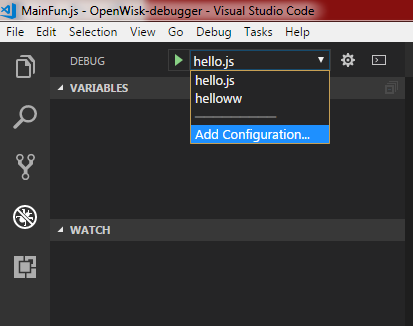
}

}

MainFun()

3. Import/Create an action using openwsk command from command palette. (Ctrl+Shift+P) which will create a folder wsk-import along with the action you created.

4. Click on add configuration in order to generate a launch.json file inside .vscode folder.



5. Set the configuration inside the launch.json available in .vs code folder.

1. Set the name of the configuration. In my example I named it as hello.js

2. Set the first program/function to be executed when you debug the action.

In the below example it’s "program": "${workspaceFolder}/MainFun.js"

3. Include the arguments such as filename of the action which you would like to debug and json payload to give input parameter to the action.

In the below example it’s

"args": [

"./wsk-import/hello.js",

"./inputParams/hello.json"

]

{

"version": "0.2.0",

"configurations": [

{

"args": [

"./wsk-import/hello.js",

"./inputParams/hello.json"

],

"type": "node",

"request": "launch",

"name": "hello.js",

"program": "${workspaceFolder}/MainFun.js",

"outFiles": [

"${workspaceFolder}/\*\*/\*.js"

]

}

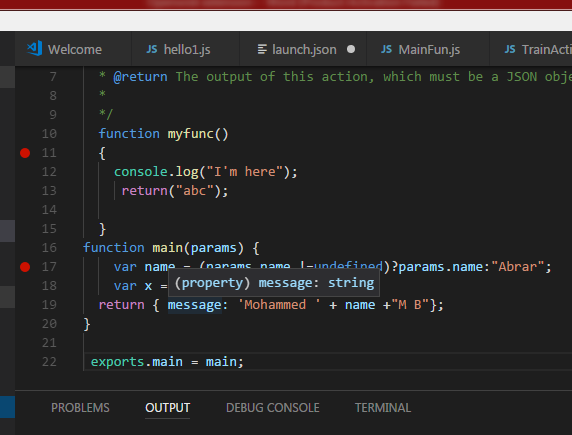
]

}

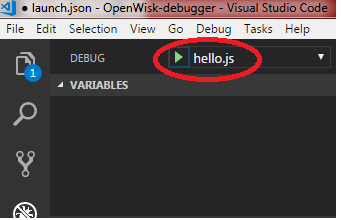
6. Set the breakpoint in the action file and before you run the debugger append

**export.main = main** at the end of the file.

**Note:** remove **export.main = main** while you create/update action over openwsk

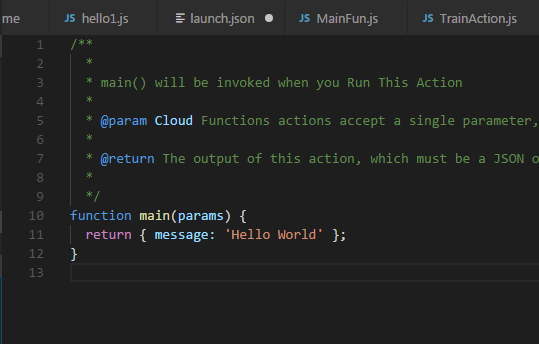


7. Go to menu and start the debugger.

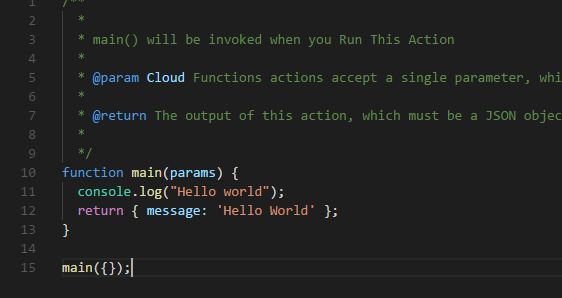


**2. Function call from the same file:**

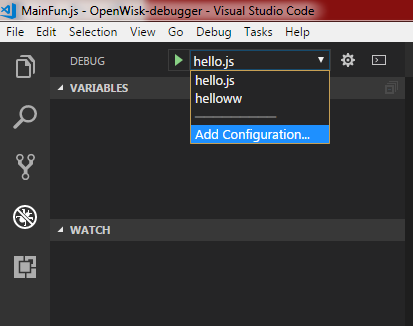
1. Import/create action using openwsk command.



2. Add function call at the end of the file. Example main({});



3. Click on add configuration in order to generate a launch.json file inside .vscode folder.

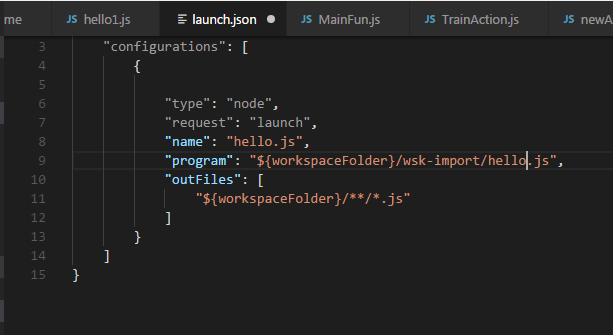


4. Set the configuration inside the launch.json available in .vs code folder.

1. Set the name of the configuration. In my example I named it as hello.js

2. Set the first function to be executed when you debug the action.

In the below example it’s "program": "${workspaceFolder}/wsk-import/hello.js"



5. Set the breakpoint where you want to debug and run the debugger.

