

3. How to Setup your development environment - PS Engineering - Nebula

Overview

This document is for setting up working IDE for AWS Lambda development. IDE used here is Microsoft Visual Studio Code which have various nice features for Node.JS development along with compatibility with AWS Lambda development and deployment. This document has step by step guidelines to create a sample Hello world program in Node.js and deploy it in AWS Lambda.

Prerequisites








1. **User can connect with IT when they are doing this installation. They will manage it with proper policy settings in one go to avoid bit9 approvals.**
2. **Try to install VSC inside C:\egainbatch folder so that if we want to execute some batch file from IDE then it won't need bit9 request.**
3. Download the VSC (Visual Studio Code) installer from the site : <https://code.visualstudio.com/download>.
4. Download the Node 12.x version from <https://nodejs.org/en/download/>.
5. Install git
6. Try to get 64 bit versions of both installers above.
7. Install code formatter and other plugins from 8. Coding Guidelines - Client side
8. <https://medium.com/better-programming/how-to-deploy-a-local-serverless-application-with-aws-sam-b7b314c3048c> -
9. Create an AWS account and IAM user with proper roles as we will need to add this credential in VSC later on. If need help then refer this tutorial .
10. Install docker from the <https://hub.docker.com/editions/community/docker-ce-desktop-windows>. Please follow steps mentioned below to do all required changes in Docker to make it compatible in eGain machine. For more info visit - <https://docs.docker.com/docker-for-windows/>
11. Also note while installing all these software we may need lot of bit9 requests so it can be time taking process. Eventually we will get them whitelisted by IT.
12. Please go through Node JS project setup in VSC before going through Lambda deployment, use this [tutorial](#).

Steps

Docker Installation :

1. Install docker from the link provided and then launch Docker App. Launching will create some issues due to lot of bit9 approvals. Note that almost all bit9 requests will be approved except kubectl.exe if you installed Docker 19.0.3.x version. To overcome follow below steps
 - a. This issue of kubectl.exe is coming due to one of the virus scanner clyance marking this harmful. IT team checks for publisher (verified source for download) , virus scan among the few things required for approval. This exe file was failing in virus scan done by Total Virus website. This is an open source website for virus scans of almost 71 Antiviruses.
 - b. To test your file in this website you can directly provide your exe file or SHA-256 of your file.
 - c. kubectl.exe version 1.15.x which comes by default with Docker is picked up by the scanner. So we can use different version for which this Virus scanner doesn't show any errors.
 - d. One such version is 1.13.x please note latest version as of Today is 1.18.x.
 - e. **Get the 1.13.x version from - <https://gcsweb.k8s.io/gcs/kubernetes-release/release/v1.13.0/bin/windows/amd64/>**
 - f. Download the new kubectl.exe and replace it with existing "{(Docker Source path)}\Docker\resources\bin" folder. For example replace the existing kubectl.exe in "C:\Program Files\Docker\Docker\resources\bin" folder with downloaded one (version 1.13.x).

This PC > OS (C:) > Program Files > Docker > Docker > resources > bin

<input type="checkbox"/> Name	Date modified	Type	Size
 docker.exe	07-05-2020 13:58	Application	57,017 KB
 docker-compose.exe	07-05-2020 13:58	Application	9,569 KB
 docker-credential-desktop.exe	07-05-2020 13:58	Application	13,586 KB
 docker-credential-wincred.exe	07-05-2020 13:58	Application	2,765 KB
 kubectl.exe	07-05-2020 18:17	Application	38,715 KB
 kubectl-backup.exe	07-05-2020 13:58	Application	42,459 KB
 notary.exe	07-05-2020 13:58	Application	9,125 KB

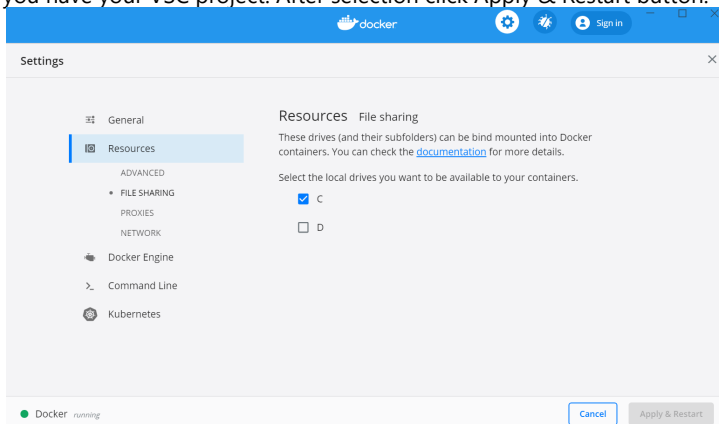
- g. After that go to "{(Docker Source path)}\Docker\resources" folder and open componentsVersion.json file.

h. Change the version of KubernteVersion attribute in json file with new downloaded version.

C:\Program Files\Docker\Docker\resources\components\Version.json - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

i. After this process raise the bit9 request for kubectl.exe again either by Launching Docker app or try opening settings from your windows App tray.

2. Once all request processes are done then we need to perform another important step to share our Drive with Docker. If this is not done then you will get Http 500 error while launching your Lambda function from VSC.
3. Please make sure Docker is running on Linux container not windows. You can do this by right click on docker icon from app tray.
4. From windows tray right click on Docker icon and then open Settings. Inside setting go to Resources→File Sharing. Select the Drive where you have your VSC project. After selection click Apply & Restart button.

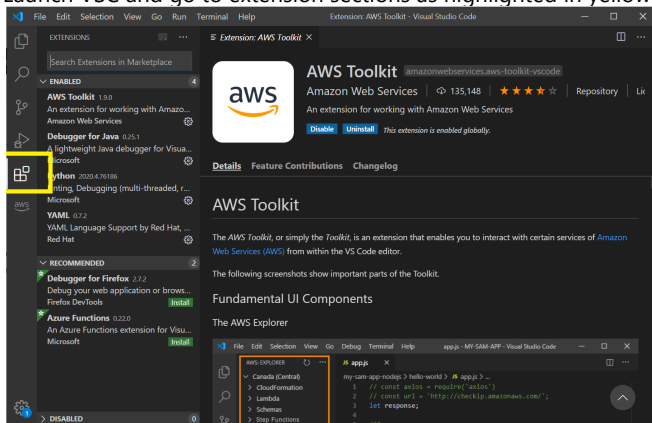


5. After these steps your Docker is ready for Lambda testing. Workaround used here for kubectl.exe is for Kubernetes which is not required for our testing. As in our case we just need Docker container to launch AWS Lambda function. Kubernetes is container orchestration system for docker and better for big clusters which is not our case. Even though Kubernetes will work here if someone needs to use.

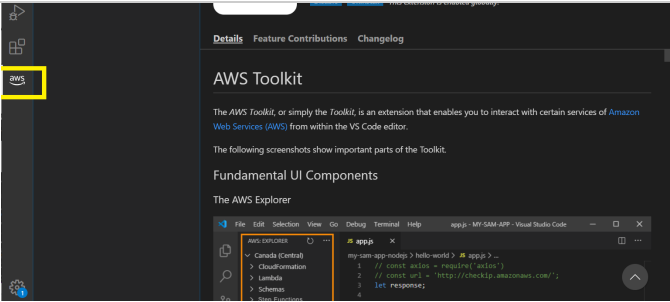
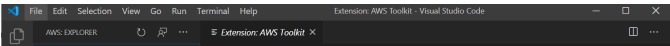
Visual Studio Code:

After VSC installation follow the below steps to configure AWS Lambda development features

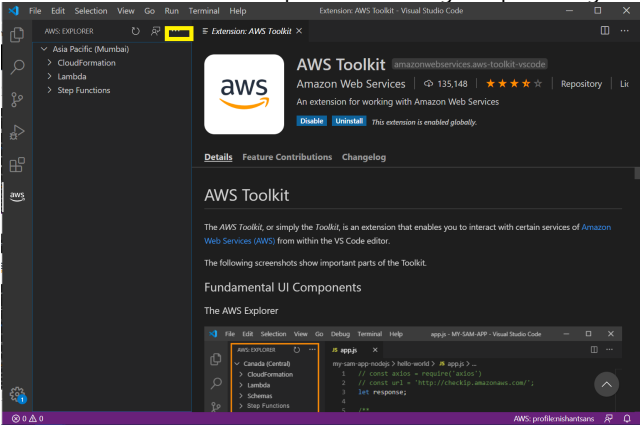
1. Launch VSC and go to extension sections as highlighted in yellow section in below pic.



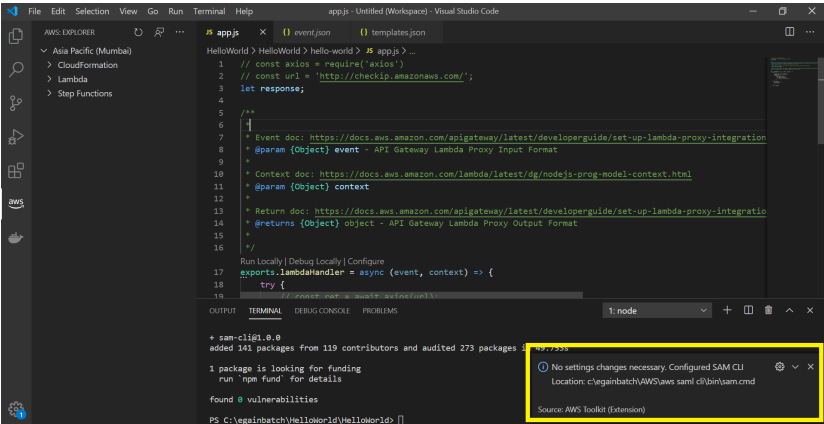
2. Type AWS toolkit and install it. After installation refresh it. Then new AWS icon will be visible below extension section, please refer highlighted section of below pic.



3. Go to AWS and select the ellipsis button at right top of navigation pane as shown in below screenshot.



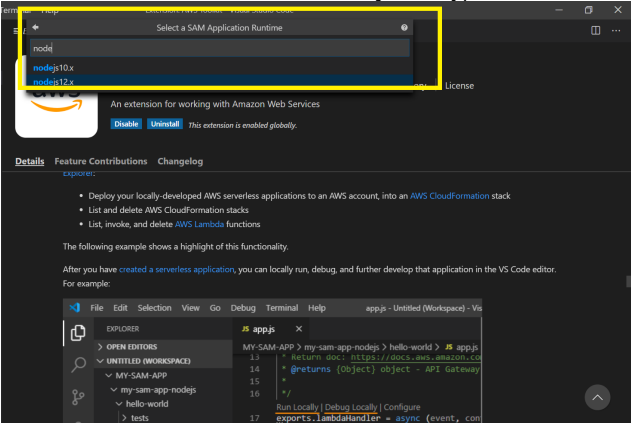
4. Press "Ctrl+Shift+P" (windows machine) and then type detect SAM CLI and select it. If SAM CLI is present in then it will be shown in right bottom notification of VSC.



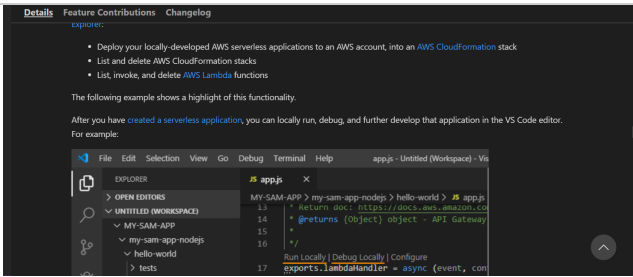
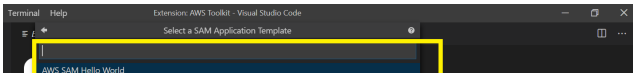
5. If not present then install sam-cli by going in Terminal and type "npm install sam-cli". You can choose appropriate folder for this if required.

6. After this lets create SAM application for which press "Ctrl+Shift+P" (windows machine) and then type AWS: Create new SAM application and press enter.

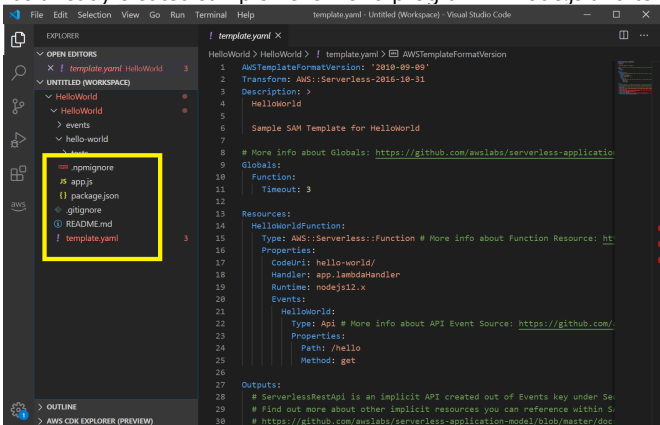
7. In next command window select Nodejs12.x, type Node so that options will be filtered out and then Nodejs12.x can be selected.



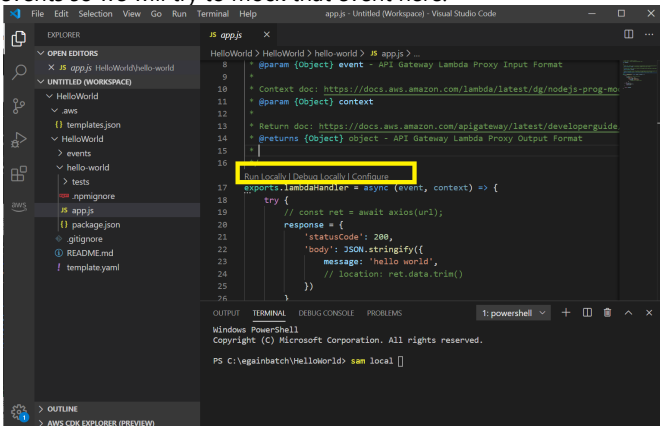
8. In the next screen select AWS SAM Hello World option.



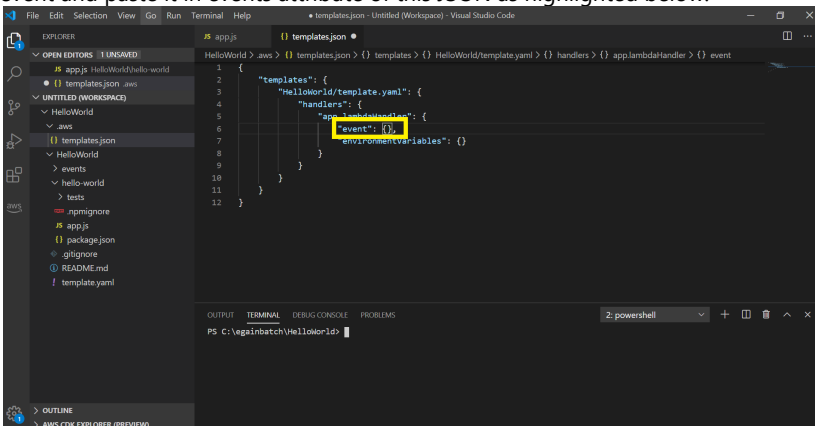
9. Provide the project folder in next screen, try to create inside D:\againbatch\{foldername}. This will help in future purpose to avoid unnecessary bit9 requests. Press enter in subsequent screen.
10. Sometimes it takes around a minute to create a SAM app so be patient. Go to explorer in VSC and check the hello world project. Notice it has already created sample hello world program in Node.js and template.yaml file.



11. Open app.js file and observe in code there is three options present Run locally | Debug Locally | Configure. With help of these options we can test our AWS Lambda function locally but before that we need to configure events using Configure tab. Lambda in AWS works on events so we will try to mock that event here.



12. Click on Configure will open new templates.json file open which will look something like below ref screenshot. Now we need to generate event and paste it in events attribute of this JSON as highlighted below.



13. Open new Terminal from Terminal options above or press "Ctrl+Shift+`". Type this command in terminal "sam local generate-event apigateway aws-proxy". This will generate mock event for API gateway call and we can use it for testing of our Lambda. For more details about all generate events using sam local refer this page - <https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-local-generate-event.html> or use "--help" options.

14. Once you execute the above command then event generated needs to be copied manually from Terminal to templates.json.

TIP: Use clip command like "sam local generate-event apigateway aws-proxy | clip" this will copy generated event in clipboard and you can directly paste inside template.json as shown in below pic

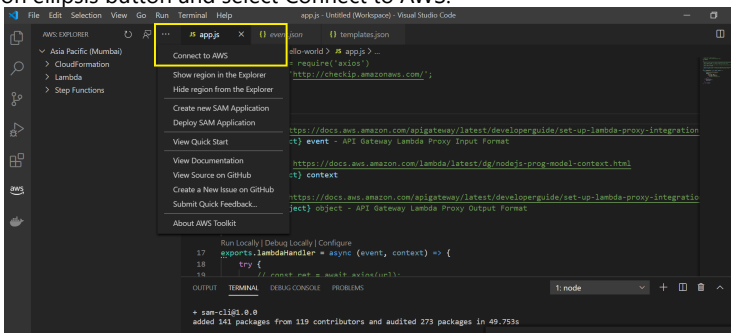


15. Now go back to app.js and click Run locally to test your Lambda function.

16. We can directly launch lambda function locally using following commands.

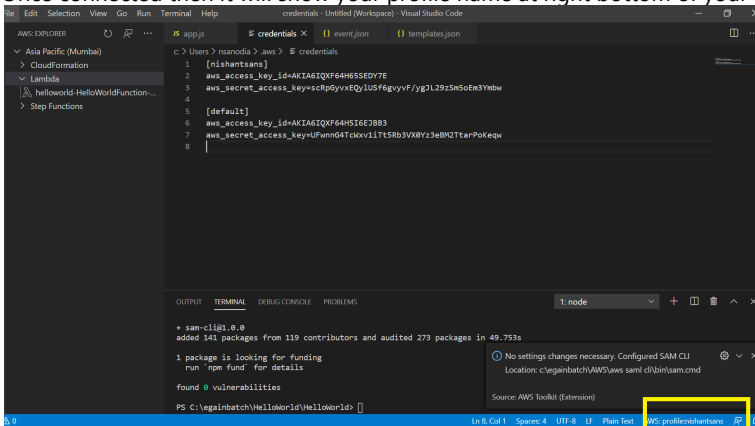
a. sam local invoke HelloWorldFunction --event events/event.json

17. To deploy AWS Lambda directly to AWS server we need to connect to AWS from VSC via toolkit, for which go to AWS extension and click on ellipsis button and select Connect to AWS.

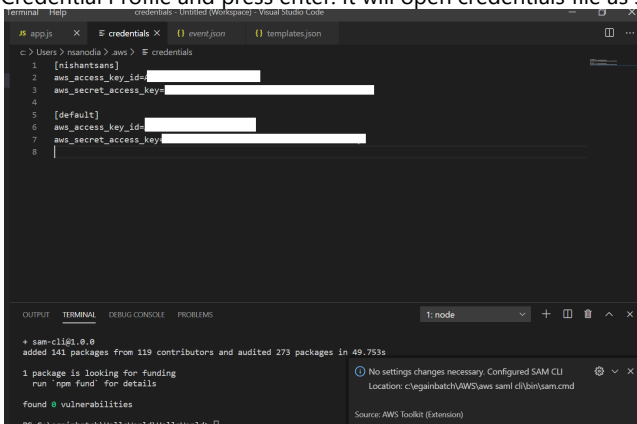


18. It will ask to create credential profile if not present where you have to provide AWS user access key and secret as mentioned in Prerequisite section.

19. Once connected then it will show your profile name at right bottom of your screen as shown in below screenshot.



20. You can have multiple profiles if required, you can do so by → Press "Ctrl+Shift+P" (windows machine) and then type AWS: Create Credential Profile and press enter. It will open credentials file as shown below where you can enter further credentials in same format.



21. Once connection is established then execute following command in Terminal to create s3 bucket where your code will be deployed.

```
aws s3 mb s3://{bucket_name} --region {region_name}
```

where bucket name should be unique if anyone has already used that name then try with other name

No labels

1 Comment



Happy Minocha

@Naveen Manoharan : Please modify this page to include VSCode settings file so that team is able to set the development environment as intended.

FYI @Nishant Sanodia / @Snehal Datar