AWS/Serverless CLI steps

AWS (provider) Web Interface

- [ ] create a User in IAM, on the AWS Console
- [ ] download the access key, secret etc

Terminal Serverless Credentials Config

- [ ] update serverless ; npm i -g serverless
- [ ] run sls or serverless to see all the options. to configure, we want sls config credentials —help at first
- [ ] type in sls config credentials —provider [aws] —key [KEY] —secret [SECRET] —profile [default IS DEFAULT]. This creates a serverless profile doc on the local machine in a hidden path: /Users/[username]/.aws/credentials
- [ ] docs: https://serverless.com/framework/docs/providers/aws/guide/credentials/#creating-aws-access-keys

Terminal: set up serverless service / cloud function

- [ ] change into the directory in which you want to create the serverless functions folders
- [ ] use serverless create : sls create —help. The command could look like: sls create —template [RUNTIME eg: aws-nodejs] —path [NAME OF SERVICE / FOLDER NAME]

this should now create the basic folder structure for serverless in that FOLDER NAME.

Serverless Functions Configuration

- [ ] cd into the FOLDER that houses the serverless deployment.
- [ ] open the .yml configuration file and change settings on dev, region
- [ ] then change settings on EVENTS - the path, HTTP event (verb) etc
- [ ] take SPECIAL note of the functions > functionName > events etc.

Invoke / Testing functions locally (pre deploy) & (post deploy)

- [ ] sls invoke local --function hello [-f flag will work in place of —function]

- [ ] if function already deployed to service provider like AWS then use sls invoke -f hello [-l *for logs*]

Using Serverless Framework "Offline" (plugin) - SETUP

- [ ] this emulates the SLS environment locally on machine. useful for local test dev before deploying to a provider
- [ ] cd into the dir that has the services and do npm init (creates package.json)
- [ ] then run npm i serverless-offline —save-dev
- [ ] update the serverless YAML file to register the serverless-offline plugin. this lets the serverless CLI know that there is a new plugin available for commands
- [ ] to do this scroll to bottom of YAML file and (if there is none) create a plugins section on new line. then line break + indent two + hyphen + space + 'serverless-offline'
- [ ] test that it is working by checking all the sls commands available in terminal. it should show two new commands: offline, and offline start

```
metrics ........................ Show metrics for a specific function
offline ........................ Simulates API Gateway to call your lambda functions offline.
offline start .................. Simulates API Gateway to call your lambda functions offline using backward compatible ini
ion.
package ........................ Packages a Serverless service
```

Using Serverless Framework "Offline" (plugin) - USE

- [ ] in the terminal, run sls offline start
- [ ] this will create a localhost port and also give you the routes you currently have available, as well as the HTTO event (GET)

```
node_modules
.gitignore
handler.js
package-lock...  U
package.json     U
serverless.yml

Practice_Projects/serverlessTutePackt/sampleService  master ✗
▶ sls offline start
Serverless: Starting Offline: dev/us-east-1.

Serverless: Routes for hello:
Serverless: GET /hello_function

Serverless: Offline listening on http://localhost:3000
```

- [ ] go to the locahost in your browser and you should see a JSON response with the message returned by the function
- [ ] if you change the function you can refresh the web browser to see changes. **DO NOT FORGET** to ctrl+C the close the offline server.

Deploy to Endpoint

- [ ] cd *into* the serverless directory (not its parent).
  cd parentFolder/FOLDER
- [ ] test with sls deploy —help to see all the options. Deploys can be entire sls service, individual functions etc.
- [ ] to deploy, do sls deploy -v . : remember to be **within the serverless folder**
- [ ] take a note of the API Gateway end point - this is the URI that must be hit to invoke the function
- [ ] the API end point can also be retrieved using sls info

Changing functions &deploying individual function

- [ ] if you're only changing a single function within the serverless service, no need to do *sls deploy -v*. instead do → **sls deploy function —function <<functionName>> .**
- [ ] the functionName is best taken from the handler.js file so you get it right

---

Removing/ Un-deploy endpoints from a <u>SERVICE</u> (not the local machine)

- [ ] sls remove -v