An Overview to Microservice based Architecture

# Objectives:

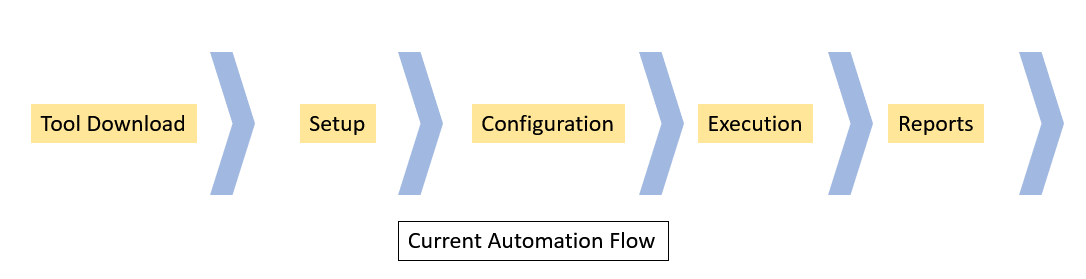
To achieve minimum of manual activities and setup process while using the current automation scripts.

# Introduction:

The proposal on Microservices based architecture with flask as an API server.

Current automation status and requirements,

* Monolithic
* Requires download and python setup to run
* Requires user’s system to execute operations



The above automation scripts can be improved using Microservice Architecture for more modular, maintainable and can be utilized without installation.

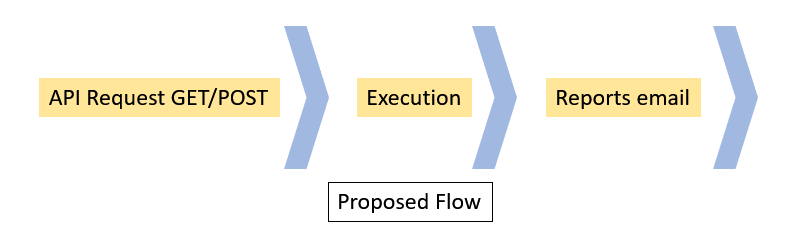
### What are the Microservices?

A group of small independent services that communicate over well-defined APIs.

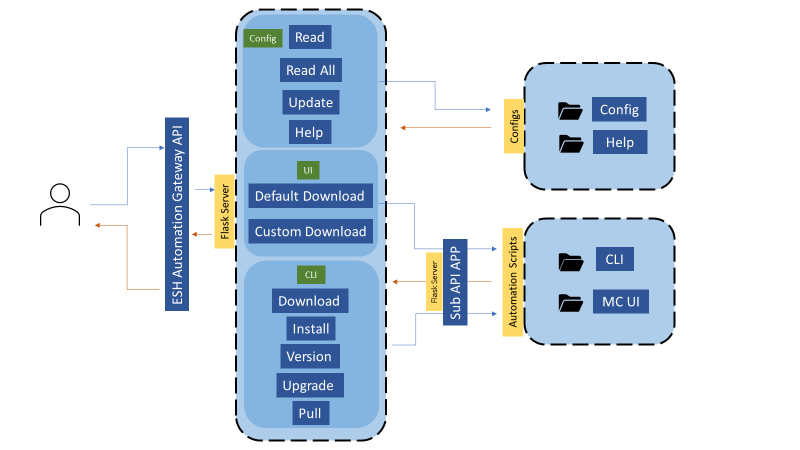
Improvements and feature can be described as following:

1. Modules will be independent and more maintainable
2. Helpful in exposing APIs for faster executions of TCs
3. No installation or system will be required.
4. In current development cycle, we have APIs, CLI and MC as Main modules. These modules with reasonable tweaks can be demonstrated using APIs.

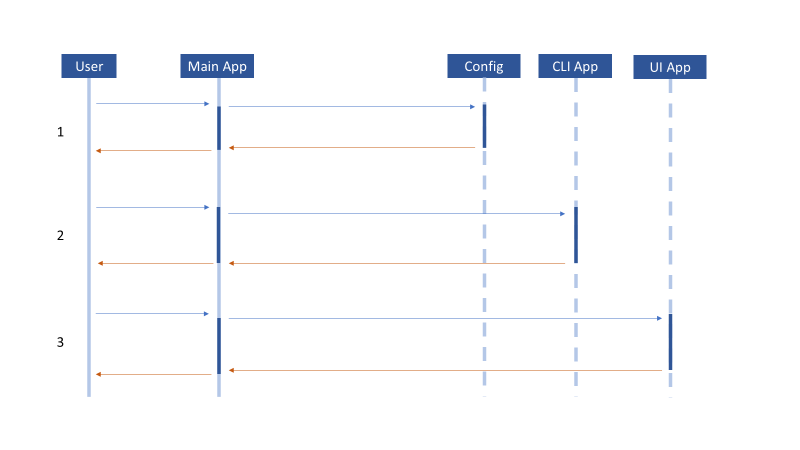
After API server setup at once, the operations will be similar to below flow.



# Architecture



# Activity Flow



# APIs for Configuration files.

Below are the API route details:

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Description | Type | Response |
| /configs | Get List of All config files | GET | [  "eshui\_testcases.cfg",  "mc\_params.cfg",  "mc\_testcases.cfg",  "params.cfg",  "servicelayer\_params.cfg",  "servicelayer\_testcases.cfg",  "testcases.cfg"  ] |
| /<cfg\_filename>/get\_section=<name> | Get section details from any config file by Config File Name | GET | {  "default\_dest\_folder": "E2ETests/downloads/OpenVINO/packages",  "default\_usecase": "Development and Runtime",  "latest\_version": "2021.3",  "os\_version": "Ubuntu 18.04 LTS",  "param\_key": "OpenVINO",  "recipe\_title": "IntelÂ® Distribution of OpenVINOâ„¢ Toolkit",  "usecases": "Development and Runtime,Runtime Only",  "webpage\_name": "OpenVINO"  } |
| /<cfgfile>/get\_section | Get All sections from any config file BY config file name | GET | { "env",  "url",  "page\_titles",  "cnda",  "OpenVINO",  ................} |
| /update/params | update the section in params.cfg | POST {'sections': 'test\_section1',  'param\_key': 'edge\_controls\_industrial',  'webpage\_name': 'edge\_controls\_industrial',  'default\_dest\_folder': 'E2ETests/downloads/edge\_controls\_industrial/packages',  'recipe\_title': 'Edge Controls for Industrial',  'latest\_version': '2.5-source',  'os\_version': 'Ubuntu 18.04 LTS'  } | |
| /update&run\_ui\_test | update the section in params.cfg and run test |

# APIs for UI

|  |  |  |
| --- | --- | --- |
| Description | Route | Payloads |
| Run default download By providing Payloads | /sub/run\_ui\_test | POST  {  'sections': 'test\_section1',  'param\_key': 'edge\_controls\_industrial',  'webpage\_name': 'edge\_controls\_industrial',  'default\_dest\_folder': 'E2ETests/downloads/edge\_controls\_industrial/packages', 'recipe\_title': 'Edge Controls for Industrial', 'latest\_version': '2.5-source', 'os\_version': 'Ubuntu 18.04 LTS' } |
| Run custom download By providing Payloads | /sub/custom/run\_ui\_test |
| Run default download By providing Package name | /sub/run\_ui\_test=<pkgname> | GET |
| Run custom download By providing Package name | /sub/custom/run\_ui\_test=<pkgname> | GET |

# APIs for CLI

|  |  |  |
| --- | --- | --- |
| Route | Payload | Response |
| checkDownload | {  'section': 'test\_section',  'env': '5',  'jenkins\_params':{  'PackageName':'connected\_logistics\_platform1',  'NewUI':'true',  'RiPackage':'true',  'UpdatePackageConfigIfExists':'true',  'EnvironmentURL':'https://software.intel.com',  'ServicelayerURL':'https://edgesoftwarehub.intel.com',  'TestType':'smoke',  'TestCasesList':'download=True install=True',  'ParamKey':'connected\_logistics\_platform1',  'WebpageName':'connected\_logistics\_platform',  'RecipeTitle':'Intel(r) Connected Logistics Platform',  'LatestVersion':'1.9',  'OsVersion':'Ubuntu 18.04 LTS',  'DefaultUsecase':'none',  'UseCases':'none',  'PypiDistribution':'false',  'PypiGuid':'none',  'EmailId':'muhilanx.saravanan@intel.com'}  } | Response: Jenkins Build Url  Mail: Final report through the email |
| checkInstall |
| checkUninstall |
| checkVersion |
| checkHelp |
| checkList |
| checkLog |
| checkUpdate |
| checkUpgrade |
| checkPull |
| checkExport |
| checkCommands |

# Requirements

A server hosting APIs needs to be setup with below specifications and dependencies,

* A server with Linux or Window OS
* python3
* pip3
* flask
* oyaml
* selenium
* xvfb
* pyvirtualdisplay
* html-testRunner
* jsonschema
* jira
* upgrade requests

Once server is hosting the APIs user needs to access them to trigger required operations.

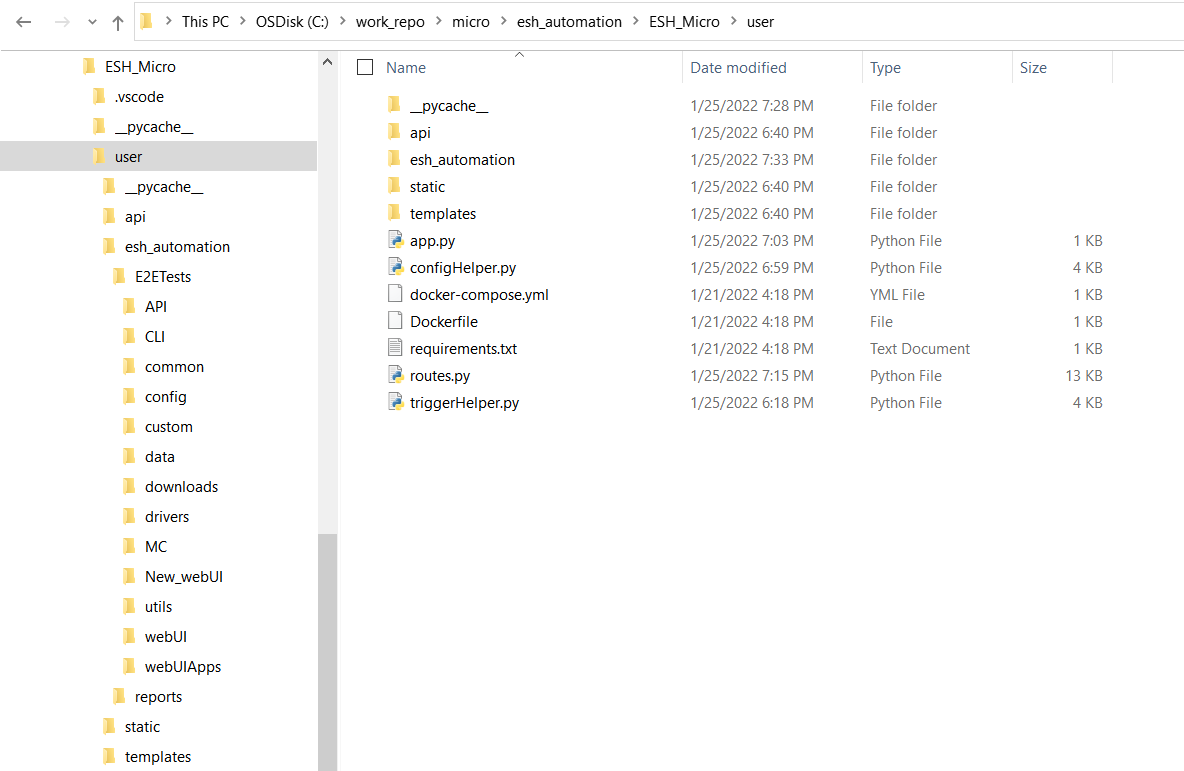
# Code structure

Code contains a main Flask app which hosts all APIs for Config files, Selector UI and CLI automation scripts.

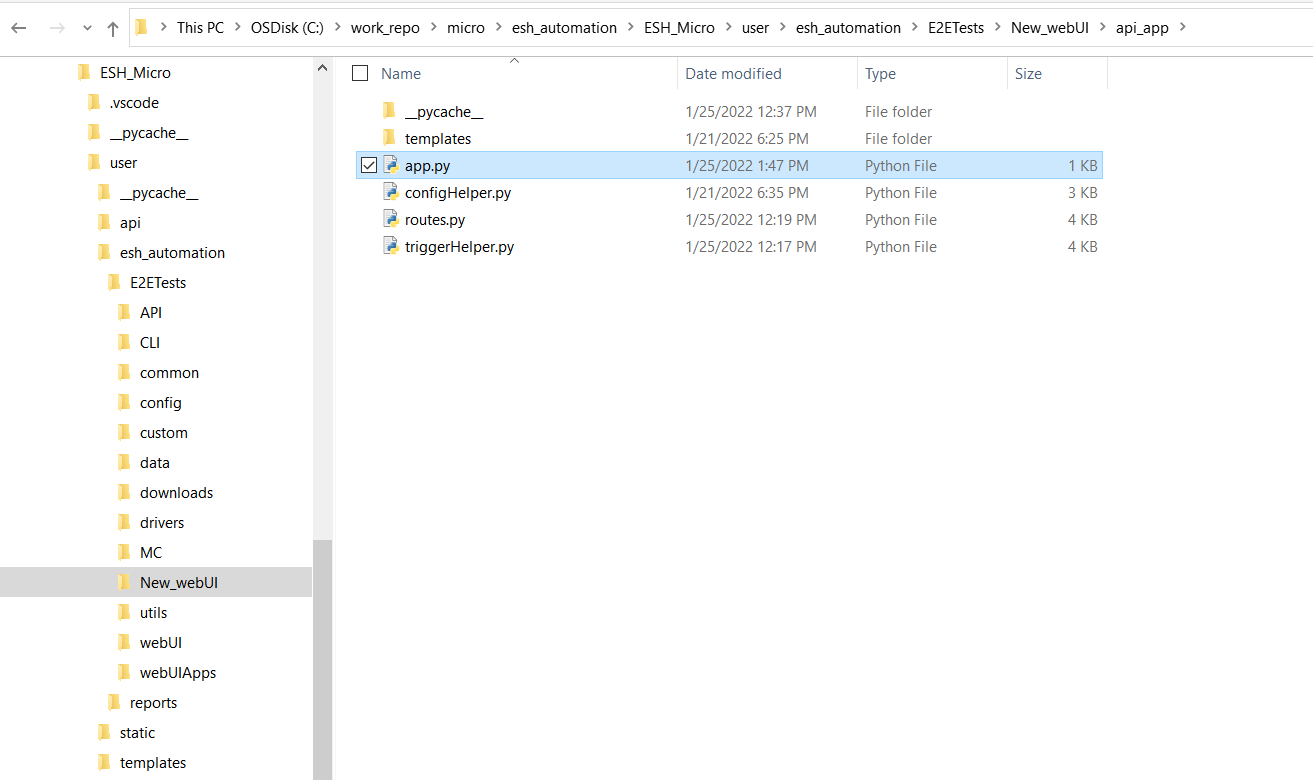
Each sub-API app is located inside automation suite where the script is located. A sub-API app can be in the ‘api\_app’ folder near the automation script.

Below is folder structure where code is located.

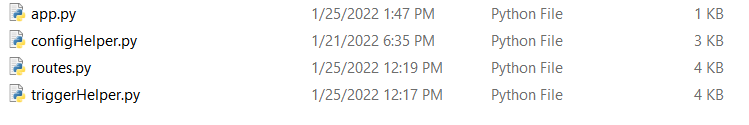
Below is location of Main Flask App.



Below is location of Sub API apps, here Selector UI.



Each of the flask app contains 4 files mainly



* App.py – for flask server hosting details
* Routes.py – for API routes and method call
* ConfigeHelper.py – for configuration(cfg) file read and write operations
* TriggerHelper.py – invokes the automation script as per API call