### How to Contribute

#### Developer Guidance

AW-SDX software system is based on a service-oriented architecture consisting of multiple dockerizable components. While being distributed and deployable independently, these components are developed with RestAPI endpoints and message queues with data models and messages following carefully designed specifications so that they communicate with each other to accomplish the end-to-end provisioning and other control workflows.

AW-SDX software team consists of a few developers and does not have testers. However, we have built a virtualized testing environment that is highly automated to allow easy duplication in other PC or Cloud platform. Therefore it is imperative for each developer to follow certain high-level rules and conventions to contribute.

* 1. **Writing Code**
     1. Go through and get a clear understanding on functions, data models, and interfaces of different system components.
     2. Propose the implementation design to be reviewed within the team.
     3. Create a meta issue in identify the implementation goals, modules, and functions agreed upon by the team review.
     4. Create a branch to develop the code.
     5. Follow Section 2 and 3 for the continuous code development.
  2. **Testing Code before submitting a PR request.**

We distinguish between two types of code that need to be tested in different ways before PR.

* + 1. Function test: The developer is responsible to provide the basic test cases for every major class and function, such as those in the PCE and DataModel repos, with an example input either through providing a main function or unittest.
    2. Component test: integration/interaction with other components, like SDX-LC to OXP and SDX-LC to SDX-Controller, or the RabbitMQ Message Queue subsystem with SDX-LC and SDX-controller.
       1. Unittest: Device the input and the expected output data models in JSON: (1) Mock topology in JSON; (2) mock request in JSON.
       2. RestAPI test: Swagger mock test for both end points and data model validation
       3. VM Testbed test: Deploy your own AW-SDX in the testbed and test the endpoints with the mock input data models
  1. **How to work with each other**

Each main component has its lead developer. It is important to follow a simple workflow between different modules in order to avoid duplicated work and inefficiency caused by confusion.

* + 1. When review and integrate with other modules, please first review and test with their specifications, APIs and main functions.
    2. If you think something is missing or not correct in another’s module, please first write a **feature request** to be reviewed by that module’s developer.
    3. After an agreement is reached, the module’s developer is responsible to design the new feature and get the approval with the requester.
    4. The module’s developer is responsible to implement and test the new feature.
    5. Then integration test will be conducted and PR procedure will be followed to accept the new implementation.

#### Code Style (PEP8)

#### Pull Request (PR) iteration with the established CI/CD workflow in Github.

#### 