INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2017-2020

FOCUS GROUP ON AUTONOMOUS NETWORKS (FG-AN)

AN-I-298-R2

Original: English

Question(s):	N/A	February 2023 (TBC)

INPUT DOCUMENT

Source:	FUTMinna, Nigeria	
Title:	Report on activities for Build-a-thon 2022 from Tech Rangers team	
Contact:	Ebeledike Frank Chukwubuikem Computer Engineering Student FUTMinna	Email: frankcebeledike@gmail.com
Contact:	Emmanuel Othniel Eggah Computer Engineering Student FUTMinna	Email: emmanueleggah@gmail.com
Contact:	Abel Oche Moses Computer Engineering Student (FUTMinna, Nigeria)	Email: oche.m1902995@st.futminna.edu
Contact:	Dr James Agajo (Associate Professor) PhD (Telecommunication and Computer Engineering) Head of Department, Computer Engineering (FUTMinna, Nigeria)	Email: james.agajo@futminna.edu.ng

Keywords: 5G, Artificial Intelligence, build-a-thon, Challenge, closed loop, controller, hackathon, Machine Learning, Proof of concept

Abstract: This contribution provides a report on activities by Tech Rangers team towards the Build-a-thon 2022. We analyse AN-usecase-001 [FGAN-use cases], “Import and export of knowledge in an autonomous network”, produce a design as per the reference design in the Build-a-thon repository. We also provide the corresponding code based on the reference code in the Build-a-thon 2022 repository.

1. References

[FGAN-use cases] ITU-T Focus Group Autonomous Networks Technical Specification “Use cases for Autonomous Networks” https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Use-case-AN.pdf

[Build-a-thon 2022] https://github.com/vrra/FGAN-Build-a-thon-2022

[FG AN Arch framework] Architecture framework for Autonomous Networks, https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Architecture-AN.pdf

https://hkrtrainings.com/what-is-tosca
https://alien4cloud.github.io/#/documentation/3.0.0/devops_guide/normative_types/tosca_concepts_types_normative_nodes.html

[FGAN Arch] ITU-T Focus Group on Autonomous Networks Technical Specification

"Architecture framework for Autonomous Networks" https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Architecture-AN.pdf

[TOSCA v1.3] TOSCA Simple Profile in YAML Version 1.3, OASIS Standard, 26 February 2020,https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.pdf

[Pydroid] https://www.makeuseof.com/install-pydroid-android/

[WINEST Github] https://github.com/Winest-Nigeria/fgan-I-298-R1

[FGAN-Build-a-thon] ITU-T FGAN Build-a-thon report https://extranet.itu.int/sites/itu-t/focusgroups/an/input/FGAN-I-289-R1.docx

[video] https://youtu.be/eL1-1vCSiMM

2. Introduction

This report was written by students from Federal University of Technology Minna towards the upcoming Build a thon 2022 activity. The team has produced TOSCA YAML file corresponding to some of the use cases in FG AN and used xOpera to parse and verify the YAML file.

The TOSCA metamodel uses the concept of service templates that describe cloud workloads as a topology template, which is a graph of node templates modelling the components a workload is made up of and of relationship templates modelling the relations between those components. TOSCA service template are instantiated at runtime using a TOSCA orchestrator (xOpera) and the order of component instantiation is based on the relationship between components. TOSCA is one of best and most often used automated testing tools. It is widely employed in large-scale applications to achieve successful outcomes.
In the TOSCA Simple Profile, TOSCA service templates must always have, as the first line in its YAML file, the keyword "tosca_definitions_version" with an associated TOSCA Namespace Alias value.

Fig 3 of [FGAN-Build-a-thon], (see below), shows the representation of modules and controller descriptions.

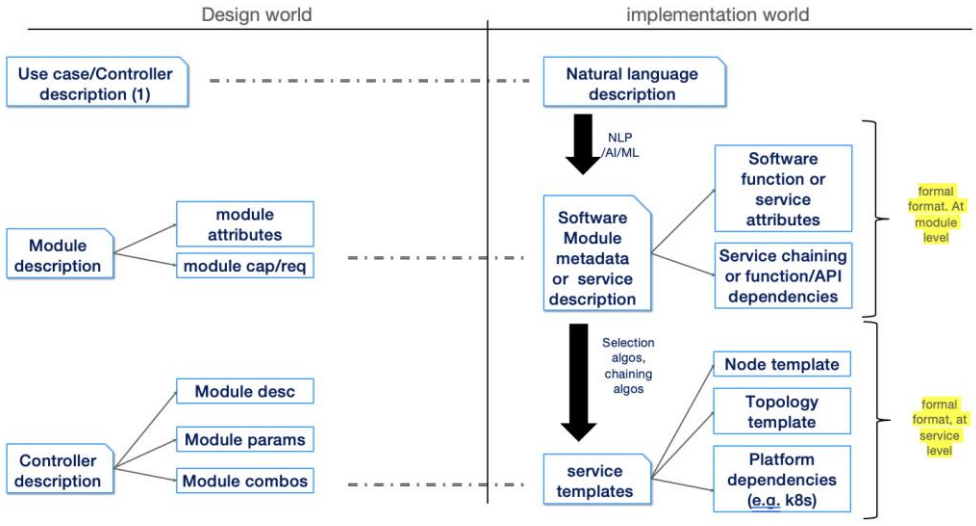


Figure 1: Problem statement steps [FGAN-I-289-R1].

The team selected Usecase-001 from the FG AN use case document. The use case was carefully studied, from which the team came up with a design (shown in Figure 2) which was implemented in TOSCA YAML. The design was validated by parsing it using the xOpera open-source orchestrator. This was tested on the Google Colab platform. Also, the team was able to show how can be replicated on android devices [see Appendix II].

Also, the team was able to show how a the YAML file can be generated from the use case stored in a graph database (an example is using neo4j). To achieve this, a basic template YAML template of the use case was written. Ruamel (a python YAML editing library) was used to populate the nodes of the use case by using information about the use case gotten from querying the graph database. Finally, the generated YAML was then validated by parsing it using xOpera.

These steps are summarized in Figure 2.

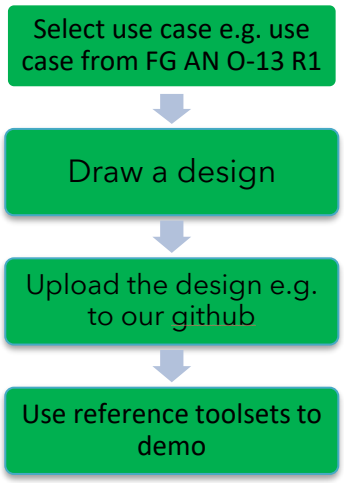


Figure 2: Problem statement steps [FGAN-I-289-R1].

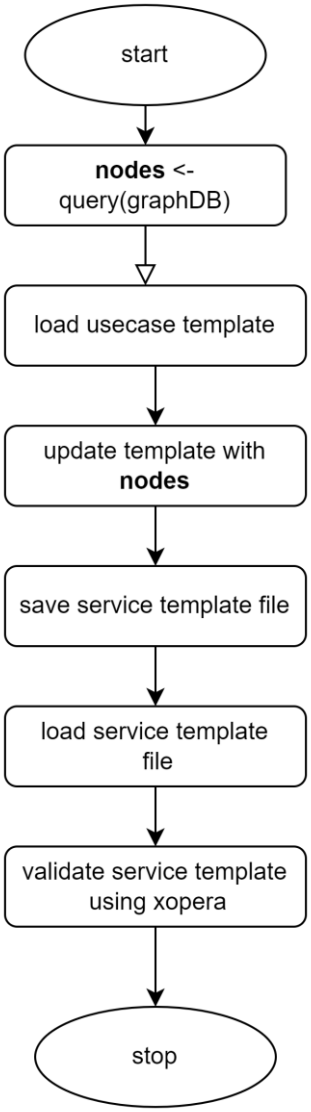


Figure 3: Flow chart of the service template generation from the graphDB.

In this contribution, we describe the use case, the approach taken by the team in designing the use case and the results from our experiments. For this, we take the following steps:
(a) Analyse the use case (manual step)
(b) Create neo4j sub graph (manual step)
(c) Create YAML (using Ruamel)
(d) Parse and deploy the YAML using xOpera

NOTE – steps (c) and (d) have been successfully done using [Pydroid] and Google colab environment.

The approach adopted by the team to develop the close loop are:

A. Populate Neo4j database with the actors and relationships from our use case in TOSCA.

Here we are using python to populate our Neo4j database with the actors and corresponding relationships. We also used the Ruamel.yaml python package which is a YAML parser/emitter to create a YAML file from Neo4j database.

B. Creating a YAML file which is made up of seven nodes (AN_Orchestrator_0, KB_manager_0, KB_0, Auto_controller_generator_0, OpenCN_0, ML_pipeline_0 and Human_Operator_0).

3. Problem statement

Due to the increasing numbers of data that needs to be processed by networks, manual techniques for management cannot handle the complexity of the network as a result there is a need for an autonomous network that runs with minimal to no human intervention—able to configure, monitor, and maintain itself independently is required to process those data at high speed with low latency with high accuracy in other to maintain a smooth running of the system.

AN-usecase-001 from [FGAN-use cases] was studied.

General use case scenarios comprise of the following steps:

1. Knowledge is imported from outside or peer entities of the AN components
2. Knowledge is referred internally in the AN components, e.g., for driving evolution, driving exploration, configuration of automation loops.
3. Generate report for human consumption
4. Knowledge is stored and updated within the AN components
5. Knowledge is exported from the AN components to outside or peer entities.

To this end, the team carried out a research on how to build a close-loop system that manages its resources and we used the TOSCA simple profile in YAML to design the close loop prototype

4. Approach taken by the team

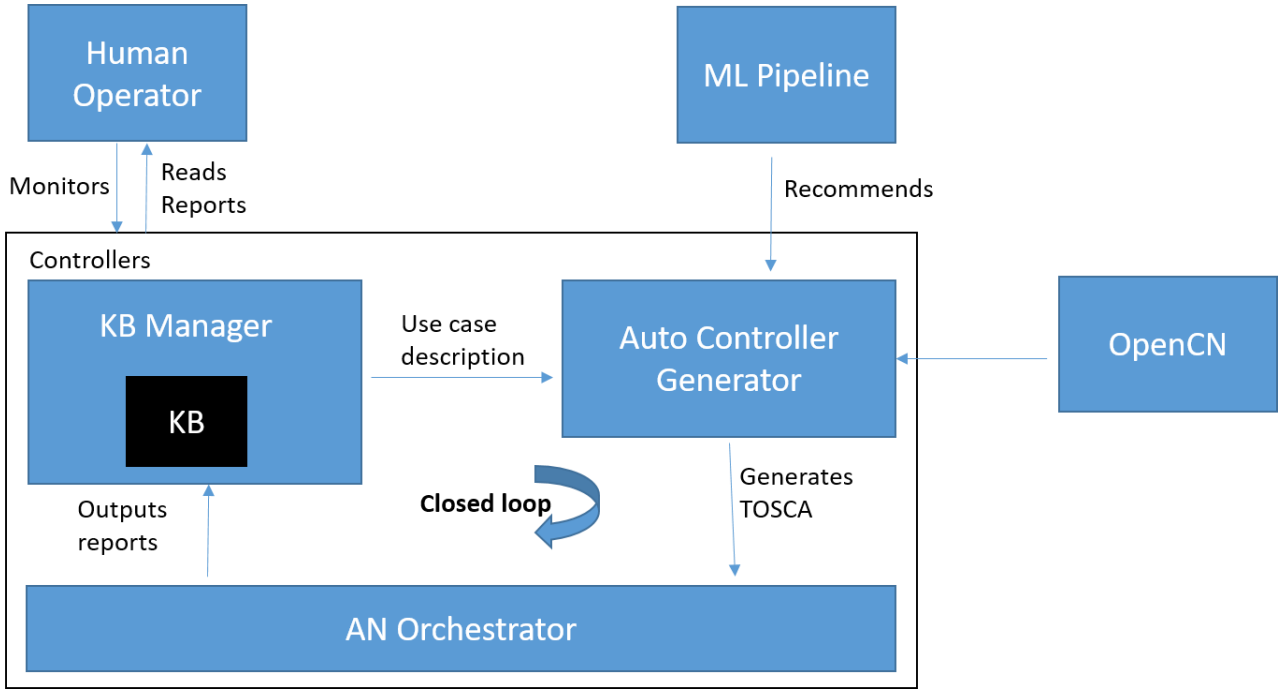
The approach adopted by our team to develop the close loop are:

C. Populate Neo4j database with the actors and relationships from our use case in TOSCA.

Here we are using python to populate our Neo4j database with the actors and corresponding relationships. We also used the Ruamel.yaml python package which is a YAML parser/emitter to create a YAML file from Neo4j database.

D. Creating a YAML file which is made up of seven nodes (AN_Orchestrator_0, KB_manager_0, KB_0, Auto_controller_generator_0, OpenCN_0, ML_pipeline_0 and Human_Operator_0).

5. High-level flow chart



AN_ORCHESTRATOR:

As per [FGAN Arch], AN orchestrator is the component responsible for managing workflows and processes in the AN and steps in the lifecycle of controllers. To manage the workflows and processes in AN, AN orchestrator coordinates with various other functions in the AN as well as outside the AN.

Being part of the management plane, AN orchestrator provides interface to human operators in the form of reports regarding the functioning of the AN and human interfaces for configuring the AN, where applicable.

KB_MANAGER:

As per [FGAN Arch], KB manager is a subsystem which manages storage, querying, export, import and optimization and update knowledge, including that derived from different sources including structured or unstructured data from various components or other subsystems.

KB manager is a node which optimizes and manages data available on the close_loop but requires a node which can host its resources. Since AN_orchestrator has the capability of hosting node resources it is designed to be a host to the KB_ manager.

KB:

As per [FGAN Arch], Knowledge in AN is a collection of resources that helps in solving a specific type of problem. A knowledge base component manages knowledge derived from and used in autonomous networks. It is updated and accessed by various components in the autonomous network.

Knowledge includes metadata which is derived from the capabilities, status of AN components. This knowledge is stored and exchanged as part of interactions of AN components with knowledge base. Knowledge can be derived from different sources including structured or unstructured data

AUTO_CONTROLLER_GENERATOR:

This node represent generic software component that can be managed and run by a TOSCA Compute Node Type. It generate controller specifications using the existing repository in OpenCN, the knowledge base and an analytics function aided by AI/ML

OPEN CN:

This node stores the controllers for the AN.

ML_PIPELINE:

This node is able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyze and draw inference from patterns in data. It hosts analytics and recommends controllers in the AN

HUMAN_OPERATOR:

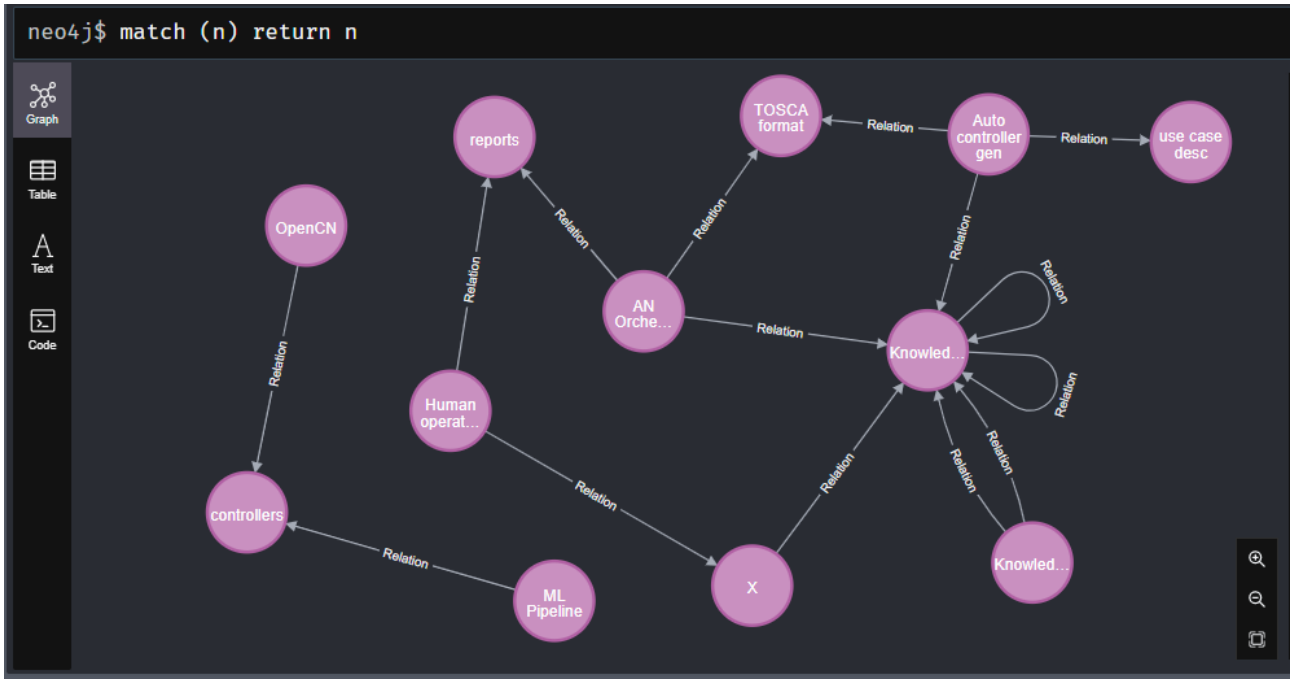
This node is just like a user friendly interface for human base instructions

6. 5.0 PoC: results

1. refer to the appendix for the YAML

2. screenshot for the neo4j graph

```
app.create_actors_relationship_with_usecase("X", "refer", "Knowledge Base", "usecase_001")
app.create_actors_relationship_with_usecase("Knowledge Base", "import", "Knowledge Base", "usecase_001")
app.create_actors_relationship_with_usecase("Knowledge Base", "export", "Knowledge Base", "usecase_001")

app.create_actors_relationship_with_usecase("Knowledge Base Manager", "optimizes", "Knowledge Base", "usecase_001")
app.create_actors_relationship_with_usecase("Knowledge Base Manager", "export", "Knowledge Base", "usecase_001")

app.create_actors_relationship_with_usecase("AN Orchestrator", "refer", "Knowledge Base", "usecase_001")
app.create_actors_relationship_with_usecase("AN Orchestrator", "input", "TOSCA format", "usecase_001")
app.create_actors_relationship_with_usecase("AN Orchestrator", "output", "reports", "usecase_001")

app.create_actors_relationship_with_usecase("Auto controller generator", "refer", "Knowledge Base", "usecase_001")
app.create_actors_relationship_with_usecase("Auto controller generator", "output", "TOSCA format", "usecase_001")
app.create_actors_relationship_with_usecase("Auto controller generator", "input", "use case desc", "usecase_001")

app.create_actors_relationship_with_usecase("OpenCN", "stores", "controllers", "usecase_001")
app.create_actors_relationship_with_usecase("ML Pipeline", "recommends", "controllers", "usecase_001")

app.create_actors_relationship_with_usecase("Human operator", "input", "reports", "usecase_001")
app.create_actors_relationship_with_usecase("Human operator", "monitors", "X", "usecase_001")
```




```
actors = app.find_all_usecase_actors_label("usecase_001")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:112: DeprecationWarning: read_transaction has been renamed to execute_read
Found actor-1: X
X
Found actor-2: Knowledge Base
Knowledge Base
Found actor-3: Knowledge Base Manager
Knowledge Base Manager
Found actor-4: AN Orchestrator
AN Orchestrator
Found actor-5: TOSCA format
TOSCA format
Found actor-6: reports
reports
Found actor-7: Auto controller generator
Auto controller generator
Found actor-8: use case desc
use case desc
Found actor-9: OpenCN
OpenCN
Found actor-10: controllers
controllers
Found actor-11: ML Pipeline
ML Pipeline
Found actor-12: Human operator
Human operator
```

2. screenshot for the compile with xOpera

```
[22] !pip install ruamel.yaml
```

```
import sys
from ruamel.yaml import YAML
```

```
[24] def create_a_template(output=sys.stdout):
        yaml = YAML()
        service = {
            "tosca_definitions_version": "tosca_simple_yaml_1_3",
            "relationship_types": {
                "basicrelationship": {
                    "derived_from": "tosca.relationships.Root"
                }
            },
            "topology_template": {
                "node_templates": {
                    actors[3]: {
                        "type": "tosca.nodes.SoftwareComponent",
                        "description": "Stores knowledge related to the AN.",
                        "requirements": [{
                            "export": {
                                "node": "knowledge_base",
                                "relationship": "exports"
                            }}, {
                            "import": {
```

```
[21] output = open("service.yaml", 'w')
     create_a_template(output)
```

```
!pip install opera==0.6.5
```

```
[ ] !opera deploy service.yaml

    [Worker_0]   Deploying knowledge_base_0
    [Worker_0]   Deployment of knowledge_base_0 complete
    [Worker_0]   Deploying knowledge_base_manager_0
    [Worker_0]   Deployment of knowledge_base_manager_0 complete
    [Worker_0]   Deploying AN_Orchestrator_0
    [Worker_0]   Deployment of AN_Orchestrator_0 complete
    [Worker_0]   Deploying OpenCN_0
    [Worker_0]   Deployment of OpenCN_0 complete
    [Worker_0]   Deploying ML_pipeline_0
    [Worker_0]   Deployment of ML_pipeline_0 complete
    [Worker_0]   Deploying Auto_Controller_Generator_0
    [Worker_0]   Deployment of Auto_Controller_Generator_0 complete
    [Worker_0]   Deploying Human_operator_0
    [Worker_0]   Deployment of Human_operator_0 complete
```

7. Problems encountered.

The major problem encountered during this activity are:

- Indentation of the YAML syntax

- ❖ Spelling errors due to syntax case sensitivity
- ❖ The version of YAML available on the device and the required version need to parse the files

8. Future activities:

This contribution provides a report on activities by Tech Rangers team towards the Build-a-thon 2022. We analyse AN-usecase-001 [Y.suppl 71], "Import and export of knowledge in an autonomous network", produce a design as per the reference design in the Build-a-thon repository. We also provide the corresponding code based on the reference code in the Build-a-thon 2022 repository.

At the time of this report the focus was on translating AN-usecase-001 to a YAML file from Neo4j Graphs using a Python script. in the future generating a YAML file for any usecase from Neo4j will be developed.

Appendix: YAML file

```
tosca_definitions_version: tosca_simple_yaml_1_3
relationship_types:
  basicrelationship:
    derived_from: tosca.relationships.Root
topology_template:
  node_templates:
    Knowledge Base:
      type: tosca.nodes.SoftwareComponent
      description: Stores knowledge related to the AN.
      requirements:
      - export:
          node: Knowledge Base
          relationship: exports
      - import:
          node: Knowledge Base
          relationship: imports
    Knowledge Base Manager:
      type: tosca.nodes.SoftwareComponent
      requirements:
      - export:
          node: Knowledge Base
          relationship: exports
      - optimize:
          node: Knowledge Base
          relationship: optimizes
```

```
  AN Orchestrator:
    type: tosca.nodes.Compute
    requirements:
    - refer:
        node: Knowledge Base
        relationship: refersTo
    - generates_Tosca:
        node: Auto controller generator
        relationship: generates
  Auto controller generator:
    type: tosca.nodes.SoftwareComponent
    description: Generate controller specifications
    requirements:
    - dependency: ML Pipeline
  OpenCN:
    type: tosca.nodes.SoftwareComponent
    description: Stores controllers
  ML Pipeline:
    type: tosca.nodes.SoftwareComponent
    description: Hosts analytics
  Human operator:
    type: tosca.nodes.SoftwareComponent
    description: Reads reports and monitors
relationship_templates:
  refersTo:
    type: basicrelationship
  optimizes:
    type: basicrelationship
  exports:
    type: basicrelationship
  stores:
    type: basicrelationship
  recommends:
    type: basicrelationship
  imports:
    type: basicrelationship
  reads:
    type: basicrelationship
  monitors:
```

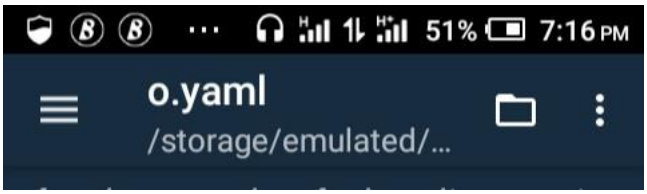
```
    type: basicrelationship
inputs:
    type: basicrelationship
generates:
    type: basicrelationship
```

Appendix: **How to install opera on Pydroid**

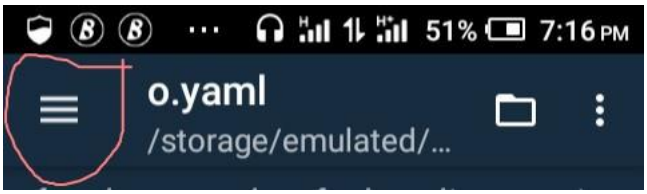
Since Pydriod is an IDE for python, to be able to parse YAML file on this platform there is a need to install a library called opera. This installation can be done on either terminal or on pip.

STEPS FOR PIP INSTALLATION

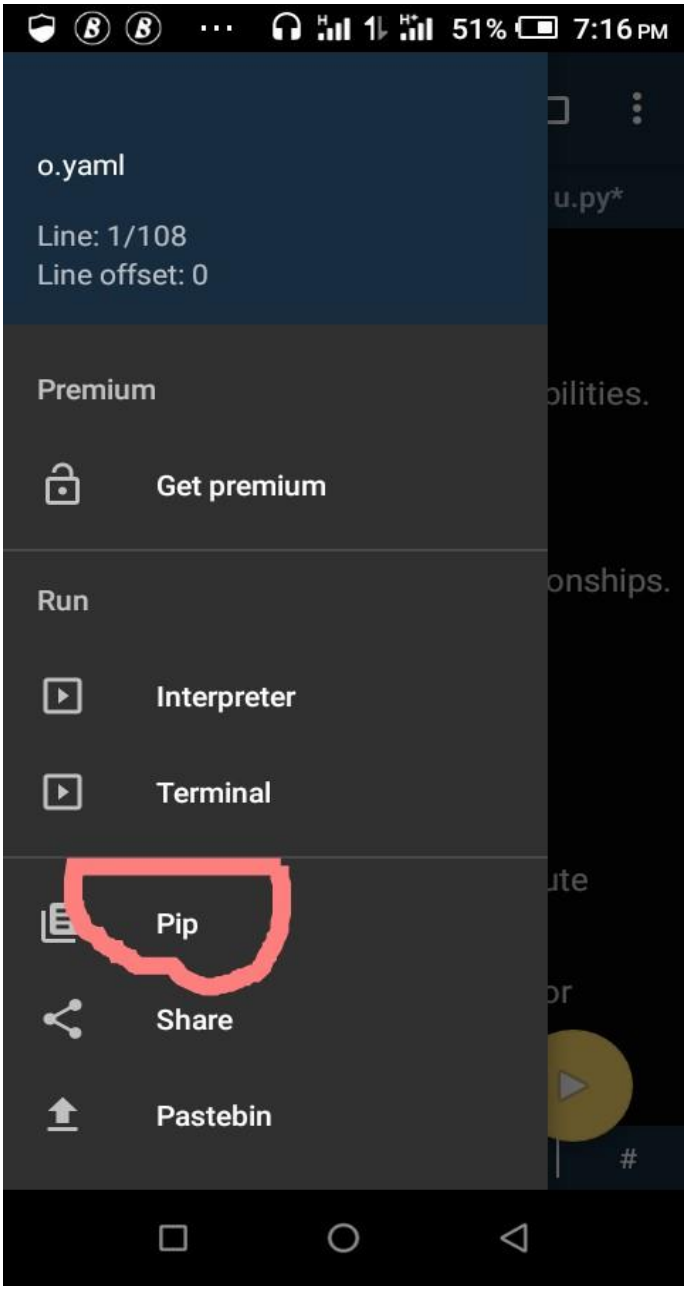
1. Open Pydriod [https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=en&gl=US] on your mobile.



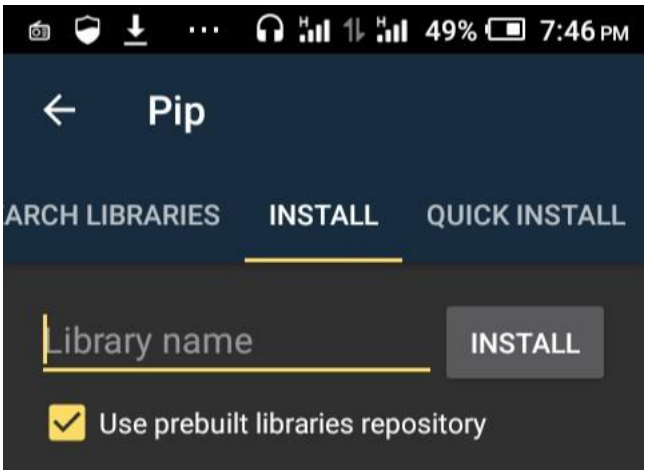
2. Click on the three horizontal lines at the top left hand side.



3. Click on pip

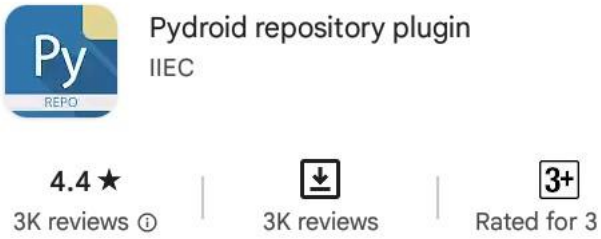


4. Type opera and click install



You may be redirected if you don't have the repository plug-in install on your phone.

To avoid this install pydroid repository plug-in from play store.



Demo video : https://youtu.be/eL1-1vCSiMM