Network Classifier

Goal

In this exercise, you will develop a classification service that identifies network devices.
This service takes classification rules and communication patterns and produces an output that contains the classification of the network devices.
In the process, the service will continuously read the communications patterns and try to match them against existing rules.
The output file contains devices and their classification according to the rules they matched.

General Guidelines

1. The exercise has several steps. Be sure to read them all before you begin.
2. Plan your work & deliver value incrementally according to the steps.
3. Pay attention to the quality of your code (design, test, clean code)
4. If something is not clear - do ask.
5. You can make any assumption to overcome questions / requirement gaps.
   Make sure to log your assumptions and write the reasoning behind them.
6. You can use the internet freely
7. You can assume the input is valid

Requirements

Service Input

Classification rules: classification_rules.csv

id (int), **type** (text), **argument** (nullable text), **classification** (text)

- **id** - the rule id
- **type** - the rule's pre-defined type (this will be explained later on)
- **argument** - argument relevant to the current rule.
- **classification** - the type of network device that will match this rule

Communication patterns: communications.csv

id (int), **timestamp** (int), **device_id** (text), **protocol_name** (text), **host** (text)

- **id** - the event id
- **timestamp** - seconds since epoch
- **device_id** - the device that the event talks about
- **protocol_name** - the protocol that triggered this event

- **host** - an IP address related to the event. Could be empty for event types for which IP is not relevant

Service Output

Classification of the network devices: classifications.csv

Classifications (csv)

id (int), **device_id** (text), **classification** (text)

- **id** - the sequential ID of the line, beginning with 1
- **device_id** - the device that the classification is about
- **classification** - the chosen classification

Tasks

1. Add support for rule type ‘communicating_protocol’ that receives the protocol_name to match as an argument in the ‘argument’ field
   Example:
   i. Rule: ‘1,communicating_protocol,http,user endpoint’
   ii. On Communication: ‘1,1578455846,aaaa,http,10.0.0.1’
   iii. Will Output: ‘1,aaaa,user endpoint’

2. Add support for rule type ‘communicating_with’ that receives in the ‘argument’ field the IP address to match as an argument
   Example:
   i. Rule: ‘1,communicating_with,10.1.1.1,ct’
   ii. On Communication: ‘1,1578455846,aaaa,ssl,10.1.1.1’
   iii. Will Output: ‘1,aaaa,ct’

3. Add support for rule type ‘communicating_with_subnet’ that receives the subnet to match as an argument in the ‘argument’ field
   Example:
   i. Rule: ‘1,communicating_with_subnet,10.1.1.0/24,ct’
   ii. On Communication: ‘1,1578455846,aaaa,ssl,10.1.1.8’
   iii. Will Output: ‘1,aaaa,ct’

4. Add support for rule type ‘communicating_with_domain’ that receives the domain to match as an argument in the ‘argument’ field
   Example:
   i. Rule: ‘1,communicating_with_domain,www.apple.com,iphone’
   ii. On Communication: ‘1,1578455846,aaaa,ssl,10.1.1.8’ (www.apple.com lookup returned 10.1.1.8)
   iii. Will Output: ‘1,aaaa,iphone’

Bonus

5. Process communications concurrently (!)
6. Add support for rule type ‘multi_rules’ that receives a list of rule ids (separated with ‘-’) to match as an argument in the ‘argument’ field.
   You may assume multi_rules will have greater id than the rules it contains
   Example:
   i. Rule:
      ‘1,communicating_with,10.1.1.1,ct
      ‘2,communicating_protocol,ssl,user endpoint”’
      3,multi_rules,1-2,mri’
   ii. On Communications:
      ‘1,1578455846,aaaa,http,10.1.1.1
      2,1578455846,aaaa,ssl,20.1.1.1’
   iii. Will Output: ‘1,aaaa,mri’

Notes:

1. If a single device_id received more than one classification, the last one (by communication id and than by rule id) will be the one selected
2. If a device will have no classification we should output ‘unknown’
3. The system should be always on and process the communications line by line