Утилита группового конфигурирования Микротиков

Установка

1. Установить deno (https://deno.land/)
2. Клонировать репозиторий http://gitlab.wi-fi.click/support/mikrotik-manager.git
3. Запустить `deno run --allow-all --unstable manager.ts`

Параметры запуска

Поддерживаются следующие параметры:

`--project=<folder>` Указывает папку с проектом, в котором содержится список хостов, переменные и задачи. ***Единственный обязательный параметр.***

`--tasks=<tasks file>` Указывает файл откуда брать задачи для выполнения. Если указать только название файла, то будет искать в папке проекта. Если абсолютный или относительный путь, то будет использовать указанный. Удобно если есть общие задачи, которые можно применить к разным проектам. По умолчанию используется файл `tasks.yaml` из папки проекта.

`--only-hosts=<hosts>` Указывает для каких хостов выполнить задачи. Хосты указываются через запятую и должны существовать в файле с хостами (см. конфигурацию). Можно указывать название или IP.

`--only-names=<regex>` Указывает для каких хостов выполнить задачи. Указывать нужно регулярное выражение.

`--full-output` По умолчанию вывод является не полным. Пропускаются хосты и задачи, которые ранее были отработаны успешно. Данный флаг позволяет выводить сообщения о пропуске.

`--ignore-success` Позволяет повторно выполнить задачи, которые ранее уже успешно были выполнены.

Конфигурация

Хосты

Хосты указываются в формате YAML в файле `hosts.yaml` или CSV в файле `hosts.csv`. Сначала проверяется наличие YAML, а затем CSV.

Параметры хостов

- `name` - **Обязательный** - отображаемое имя хоста.
- `ip` - **Обязательный** - ip-адрес для подключения.
- `port` - порт API для подключения (*не ssl*). По умолчанию `8728`.
- `user` - имя пользователя для подключения. По умолчанию `admin`.
- `password` - пароль для подключения. По умолчанию пустой.

У каждого хоста могут быть дополнительные параметры, которые могут быть использованы в качестве переменных при выполнении задач.

hosts.yaml

Файл в формате YAML где корневой элемент `hosts` - массив с объектами, каждый из которых содержит описание хоста. Дополнительные параметры указываются как поля в объекте хоста.

```
hosts:
  - name: mk_sibintek_kaluga_azs_1
    ip: 10.10.42.58
    port: 8292
    user: admin
    password: wfc_admin_2020
  - name: mk_sibintek_kaluga_azs_2
    ip: 10.10.42.59
    port: 8292
    user: admin
    password: wfc_admin_2020
  - name: mk_sibintek_kaluga_azs_3
```

```
    ip: 10.10.42.60
    port: 8292
    user: admin
    password: wfc_admin_2020
  - name: mk_sibintek_kaluga_azs_5
    ip: 10.10.42.61
    port: 8292
    user: admin
    password: wfc_admin_2020
```

hosts.csv

Файл в формате CSV, где первая строка это заголовки параметров, а все последующие - записи с параметрами хостов. Дополнительные параметры указываются как столбцы. Разделитель в файле - запятая `,`.

```
name,ip,user,password
mk_sibintek_azs_10,10.10.41.181,admin,wfc_admin_2020
mk_sibintek_azs_4,10.10.41.182,admin,wfc_admin_2020
mk_sibintek_azs_5,10.10.41.183,admin,wfc_admin_2020
mk_sibintek_azs_37,10.10.41.184,admin,wfc_admin_2020
mk_sibintek_azs_80,10.10.41.185,admin,wfc_admin_2020
```

Задачи

.Задачи хранятся в формате YAML.

Задачи выполняются последовательно.

Корневые элементы

`id` - уникальный идентификатор файла задач. Используется для сохранения состояния выполнения задач. Позволяет изменять имя файла и переносить его из одной папки в другую. Если отсутствует, то для сохранения используется параметр, переданный в параметре `--tasks`.

> Online UUIDv4 generator

`ping-before` говорит о том, что нужно пропинговать хост перед выполнением задачи. Если хост не пингуется, то предпроверки и задачи не выполняются.

`dump-vars` - имя файла, куда после выполнения будут сохранены результаты и переменные. Сохраняется в формате csv и создается в текущей директории.

`vars` - переменные, используемые при выполнении задачи.

`tasks` - массив с задачами.

Описание задачи

`id` - уникальный идентификатор файла задачи. Используется для сохранения состояния выполнения задач. Позволяет изменять имя задачи и переносить ее в другой файл. Если отсутствует, то для сохранения используется параметр `name`.

`name` - Имя задачи

`ignore-error` - Если указан true, то ошибка при выполнении задачи не прерывает выполнение всего списка задач.

`preflights` - Предварительные проверки и заполнение переменных.

`commands` - Команды для выполнения.

Предпроверки

Является массивом и выполняется последовательно. Если проверка не пройдена, то задача не выполняется. С помощью параметра `save-success-on-fail` можно считать это успехом и сохранить в базу. Может быть удобно, если есть набор устройств с разными версиями прошивки, в которых отличаются параметры команд.

Параметры

`command` - строка с командой API. Возможны замены параметров.

`args` - массив с параметрами команды API. Возможны замены параметров.

`checks` - проверки. Результатом выполнения любой команды является массив с объектами. Если логически предполагается, что это не массив, то возвращается массив с одним объектом (например у команды `/system/watchdog/print`).

> `length` - *Первая проверка.* Число. Проверяет длину массива с результатом. Например, можно проверить количество активных интерфейсов или записей в файрволе.
>
> `has` - *Вторая проверка.* Объект. Проверяет наличие в результате хотя бы одного объекта с указанными параметрами. Если значение - строка, то она проверяется на соответствие. Так же может быть объектом `regex: <regex>`, в этом случае проверка будет проходить на соответствие регулярному выражению.

> Проверить свои регулярки можно на Regex101

`vars` - переменные, которые необходимо сохранить. Объект. Значение - это *путь* до параметра в результате, например `[0]['.id']` - идентификатор первого объекта в результате выполнения задачи. Нумерация начинается с 0.

`save-success-on-fail` - позволяет считать непрохождение проверки успехом.

Команды

Является массивом с командами.

Параметры

`command` - строка с командой API. Возможны замены параметров.

`args` - массив с параметрами команды API. Возможны замены параметров.

`ignore-error` - Если указан true, то ошибка при выполнении команды не прерывает выполнение всего списка команд в задаче.

Переменные

В командах и их аргументах можно использовать переменные. Замена происходит с помощью библиотеки mustache.

В стандартном случае можно написать `{{var}}` и это заменится на значение переменной `var`.

> Если переменная имеет много трок и символы, которые декодируются в HTML, то надо использовать три скобки, например `{{{var}}}`.

Переменные берутся и перезаписываются для каждого хоста независимо в следующем порядке:

1. Переменные хоста
2. Переменные файла задач
3. Переменные из предпроверок (всех, последовательно).

Примеры

```
```

```
id: 64b019f1-2cb1-4c55-9388-b343eebd6e27
ping-before: true
vars:
  script: >
    {
      :local target "1.1.1.1"
      :local pingCount 10
      :local pingTreshold 8

      :local result [ping count=$pingCount $target]
      :log info "Ping has $result successes of $pingCount"

      :if ($result < $pingTreshold) do={
        :log info "Reset usb power"
        /system routerboard usb power-reset duration=5s
      } else={
        :log info "Skip usb power reset"
      }
    }
tasks:
  - id: 97b692e2-e1d6-475d-9b4b-e4d80aab9961
    name: Add script and scheduler
    ignore-error: false
    preflights:
      - command: /system/schedule/print
        args:
          - '?name=check_lte'
        checks:
          length: 0
      - command: /system/script/print
        args:
          - '?name=check_lte'
        checks:
          length: 0
    commands:
      - command: /system/script/add
        args:
          - '=name=check_lte'
          - '=source={{{script}}}'
        ignore-error: true
```

```
    - command: /system/schedule/add
      args:
        - '=name=check_lte'
        - '=on-event=/system script run check_lte'
        - '=interval=30s'
        - '=disabled=no'
      ignore-error: true
```

```
tasks:
  - name: Disable fastforward
    ignore-error: true
    preflights:
      - command: /ip/firewall/filter/print
        args:
          - '?action=fasttrack-connection'
        checks:
          length: 1
          has:
            disabled: 'false'
        vars:
          id: "[0]['.id']"
    commands:
      - command: /ip/firewall/filter/set
        args:
          - '=.id={{id}}'
          - '=disabled=true'
```

Быстрый старт