Arista CLoudVision Container Manager Script

Release 0.2

EMEA

Apr 30, 2019

Contents:

CHAPTER 1

Container manager for CoudVision

Generic script to manage containers on Arista Cloudvision server. It is based on cvprac library to interact using APIs calls between your client and CVP interface.

Supported Features

- **Check** if container exists on CVP.
- **Create** a container on CVP topology
- **Delete** a container from CVP topology.
- Get **information** from Cloudvision based on container's name.
- Collect list of attached devices.
- Move a devices to an existing container.

Complete documentation available on read the doc

1.1 Known Issue

Due to a change in CVP API, change-control needs to get snapshot referenced per task. Current version of `cvprack` does not support it in version 1.0.1

Fix is available in develop version. To install development version, use pip:

```
$ pip install git+https://github.com/aristanetworks/cvprac.git@develop
```

1.2 Getting Started

```
$ pip install git+https://github.com/titom73/arista-cvp-container-management.git

# Update your credential information
$ cat <<EOT > env.variables.sh
export CVP_HOST='xxx.xxx.xxx.xxx'
export CVP_PORT=443
export CVP_PROTO='https'
export CVP_USER='username'
export CVP_PASS='password'
export CVP_TZ='France'
export CVP_COUNTRY='France'
EOT

# run script (assuming VLANs configlet is present on CVP)
$ arista-cvp-container-management -j actions.json
```

1.3 License

Project is published under BSD License.

1.4 Ask question or report issue

Please open an issue on Github this is the fastest way to get an answer.

1.5 Contribute

Contributing pull requests are gladly welcomed for this repository. If you are planning a big change, please start a discussion first to make sure we'll be able to merge it.

CHAPTER 2

Installation

Script can be used with 2 different installation method:

- git clone for testing. In this case it is recommended to use a virtual-environment
- Python PIP module to install binary directly to your syste. A virtual-environment is also recommended for testing purpose.

2.1 Installation with PIP

```
$ pip install git+https://github.com/titom73/arista-cvp-container-management.git

# Update your credential information
$ cat <<EOT > env.variables.sh
export CVP_HOST='xxx.xxx.xxx.xxx'
export CVP_PORT=443
export CVP_PROTO='https'
export CVP_USER='username'
export CVP_PASS='password'
EOT

$ source env.variables.sh

# run script (assuming VLANs configlet is present on CVP)
$ arista-cvp-container-management -j actions.json
```

2.2 Git Clone

It is highly recommended to use Python virtual environment for testing

```
$ git clone https://github.com/titom73/configlet-cvp-uploader.git

$ pip install -r requirements.txt

# Update your credential information
$ cat <<EOT > env.variables.sh
export CVP_HOST='xxx.xxx.xxx.xxx'
export CVP_PORT=443
export CVP_PROTO='https'
export CVP_USER='username'
export CVP_PASS='password'
EOT

$ source env.variables.sh

# run script (assuming VLANs configlet is present on CVP)
$ bin/aristaCvpContainerManager -j actions.json
```

CHAPTER 3

Script options

Script provides a set of different options and all can be set by using *SHELL* environment variables or *CLI* parameters.

3.1 Options within shell environment

By default, script will lookup for a set of variables in your environment:

- `CVP_HOST`: Hostname or IP address of CVP server
- `CVP_PORT`: CVP port to use to communicate with API engine. Default is 443
- `CVP_PROTO`: Transport protocol to discuss with CVP. Default is HTTPS
- `CVP_USER`: Username to use for CVP connection
- `CVP_PASS`: Password to use for CVP connection
- `LOG_LEVEL`: Script verbosity. Default is info
- `CVP_TZ`: Timezone used to configure change-control
- `TZ_COUNTRY`: Country to use in change-control configuration.
- `CERT_VALIDATION`: Whether or not activate SSL Cert validation. Default is False to manage self signed certificates.

In your shell, execute following commands:

```
export CVP_HOST='IP_ADDRESS_OF_CVP_SERVER'
export CVP_PORT=443
export CVP_PROTO='https'
export CVP_USER='YOUR_CVP_USERNAME'
export CVP_PASS='YOUR_CVP_PASSWORD'
export CVP_TZ=France
export CVP_COUNTRY='France'
```

A script example is available in the repository for informational purpose

It can be configured in your `~/.bashrc` or in VARIABLES of a CI/CD pipeline as well.

3.2 Options from the CLI

This approach overrides options defined in your shell environment

```
$ arista-cvp-container-management -h

usage: arista-cvp-container-management.py [-h] [-v] [-c CONFIGLET] [-u USERNAME]
                            [-p PASSWORD] [-s CVP] [-d DEBUG_LEVEL]
                            [-j JSON]

Configlet Uploader to CVP

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -c CONFIGLET, --configlet CONFIGLET
                        Configlet path to use on CVP
  -u USERNAME, --username U SERNAME
                        Username for CVP
  -p PASSWORD, --password PASSWORD
                        Password for CVP
  -s CVP, --cvp CVP     Address of CVP server
  -d DEBUG_LEVEL, --debug_level DEBUG_LEVEL
                        Verbose level (debug / info / war ning / error /
                        critical)
  -j JSON, --json JSON  File with list of actions to execute)
```

CHAPTER 4

How to use configilet uploader

Script uses a JSON file to describe list of actions to run on CloudVision server. This json file is provided to the script by using `-json` trigger on CLI.

JSON file is an array of entries where every single entry in JSON file describe a task to run:

```
[
  {
      //task 1
  },
  {
      //task 2
  }
]
```

Current version of code support all the actions listed below:

- Create a container in CoudVision topology
- Move a list of devices to an existing container.
- Delete a container from CloudVision topology.

4.1 Create a container within CVP Topology:

You can create a container in CloudVision topology using a JSON like below.

4.1.1 JSON example:

```
{
    "name": "Create container",
    "type": "container",
    "action": "create",
```

(continues on next page)

(continued from previous page)

```
    "container": "Test Container",
    "parent": "Tenant"
}
```

Where **keys** have description below:

- `name`: A name for the task. it is only a local name and it is not used on CVP side.
- `type`: shall be **container**. It define what kind of entry to manage on CVP. in this case, we are talking about a container.
- `action`: Action to run on configlet. As we want to attach devices to container, action shall be **creation**
- `container`: Name of existing container where devices will be attached.
- `parent`: Name of parent container. It is value you have in your toipology. By default, container will be created under **Tenant**

Warning: This action execute task directly and there is no way to just provisionned and execute action later or manually.

4.1.2 Example outputs:

```
--------------------

2019-04-30 13:51:51 INFO     creation of container with name Test Container attached␣
↪to Tenant
2019-04-30 13:51:52 INFO     Connected to 54.219.174.143
2019-04-30 13:51:52 INFO     *************
2019-04-30 13:51:52 INFO     Start working with Test Container
2019-04-30 13:51:52 INFO     initializing a container object for Test Container
2019-04-30 13:51:52 INFO     Version [u'2018', u'2', u'2']
2019-04-30 13:51:52 INFO     Setting API version to v2
2019-04-30 13:51:54 WARNING  container Test Container not found
2019-04-30 13:51:54 INFO     create container on CVP server
2019-04-30 13:51:54 INFO     start creation of container attached to Tenant
```

4.2 Delete a container from CVP Topology:

You can delete a container in CloudVision topology using a JSON like below.

4.2.1 JSON example:

```
{
    "name": "Create container",
    "type": "container",
    "action": "destroy",
    "container": "Test Container",
    "parent": "Tenant"
}
```

Where **keys** have description below:

- `name`: A name for the task. it is only a local name and it is not used on CVP side.
- `type`: shall be **container**. It define what kind of entry to manage on CVP. in this case, we are talking about a container.
- `action`: Action to run on configlet. As we want to attach devices to container, action shall be **destroy**
- `container`: Name of existing container where devices will be attached.
- `parent`: Name of parent container. It is value you have in your toipology. By default, container will be created under **Tenant**

Note: To execute this action, your container should not contain any attached device. if some are still attached, process will stop.

Warning: This action execute task directly and there is no way to just provisionned and execute action later or manually.

4.2.2 Example outputs:

```
--------------------

2019-04-30 14:17:36 INFO     destruction of container with name Test Container
2019-04-30 14:17:37 INFO     Connected to 54.219.174.143
2019-04-30 14:17:37 INFO     *************
2019-04-30 14:17:37 INFO     Start working with Test Container
2019-04-30 14:17:37 INFO     initializing a container object for Test Container
2019-04-30 14:17:37 INFO     Version [u'2018', u'2', u'2']
2019-04-30 14:17:37 INFO     Setting API version to v2
2019-04-30 14:17:41 INFO     destroy container from CVP server
2019-04-30 14:17:41 INFO     start process to delete container Test Container
```

4.3 Move devices to an existing container:

Script provides a mechanism to move devices to an existing container. JSON syntax to support such operation is provided below:

4.3.1 JSON example:

```
{
      "name": "Change CVX to EVPN",
      "type": "container",
      "action": "attach-device",
      "container": "CVX",
      "apply": true,
      "devices": [
          "leaf1",
          "leaf2",
```

(continues on next page)

(continued from previous page)

```
        "cvx01"
    ]
}
```

Where **keys** have description below:

- `name`: A name for the task. it is only a local name and it is not used on CVP side.
- `type`: shall be **container**. It define what kind of entry to manage on CVP. in this case, we are talking about a container.
- `action`: Action to run on configlet. As we want to attach devices to container, action shall be **attach-device**
- `container`: Name of existing container where devices will be attached.
- `apply`: define wether or not we should deploy this configlet to devices. if set to **false**, then a change-control or manual action should be done later by user.
- `devices`: An array of devices hostname configured on CVP to move to `container`.

4.3.2 Example outputs:

```
--------------------

2019-04-30 10:21:54 INFO     device leaf1 is going to be moved to CVX
2019-04-30 10:21:54 INFO     device leaf2 is going to be moved to CVX
2019-04-30 10:21:54 INFO     device cvx01 is going to be moved to CVX
2019-04-30 10:21:55 INFO     Connected to 54.219.174.143
2019-04-30 10:21:55 INFO     *************
2019-04-30 10:21:55 INFO     Start working with CVX
2019-04-30 10:21:55 INFO     initializing a container object for CVX
2019-04-30 10:21:55 INFO     Version [u'2018', u'2', u'2']
2019-04-30 10:21:55 INFO     Setting API version to v2
2019-04-30 10:21:59 INFO     check is devices are already part of container
2019-04-30 10:21:59 INFO     device is not part of that container -- moving forward
2019-04-30 10:21:59 INFO     device is not part of that container -- moving forward
2019-04-30 10:21:59 CRITICAL device is already part of that container -- skipping
2019-04-30 10:21:59 INFO     >---
2019-04-30 10:21:59 INFO     starting process to attach a list of device to CVX
2019-04-30 10:21:59 INFO     >---
2019-04-30 10:21:59 INFO     create change to move leaf1 to CVX
2019-04-30 10:22:03 INFO     task created on CVP: 250
2019-04-30 10:22:03 INFO     >---
2019-04-30 10:22:03 INFO     create change to move leaf2 to CVX
2019-04-30 10:22:06 INFO     task created on CVP: 251
2019-04-30 10:22:06 INFO     >---
2019-04-30 10:22:06 CRITICAL device already attached to CVX
2019-04-30 10:22:06 INFO     >---
2019-04-30 10:22:06 INFO     run pending tasks to related to container CVX
2019-04-30 10:22:06 INFO      -> execute task ID: 250
2019-04-30 10:22:08 INFO       * Wait for task completion (status: ACTIVE) / waiting␣
↪for 0 sec
2019-04-30 10:22:09 INFO       * Wait for task completion (status: ACTIVE) / waiting␣
↪for 1 sec
2019-04-30 10:22:10 INFO       * Wait for task completion (status: ACTIVE) / waiting␣
↪for 2 sec
2019-04-30 10:22:12 INFO       * Wait for task completion (status: COMPLETED) /␣
↪waiting for 3 sec
```

(continues on next page)

(continued from previous page)

```
2019-04-30 10:22:12 INFO      -> task 250 status : COMPLETED
2019-04-30 10:22:12 INFO      -> execute task ID: 251
2019-04-30 10:22:13 INFO       * Wait for task completion (status: ACTIVE) / waiting␣
↪for 0 sec
2019-04-30 10:22:14 INFO       * Wait for task completion (status: ACTIVE) / waiting␣
↪for 1 sec
2019-04-30 10:22:15 INFO       * Wait for task completion (status: ACTIVE) / waiting␣
↪for 2 sec
2019-04-30 10:22:17 INFO       * Wait for task completion (status: COMPLETED) /␣
↪waiting for 3 sec
2019-04-30 10:22:17 INFO      -> task 251 status : COMPLETED
```

CHAPTER 5

aristaCvpContainerManager module

CHAPTER 6

bin

CHAPTER 7

Container manager for CoudVision

Generic script to manage containers on Arista Cloudvision server. It is based on cvprac library to interact using APIs calls between your client and CVP interface.

Supported Features

- **Check** if container exists on CVP.
- **Create** a container on CVP topology
- **Delete** a container from CVP topology.
- Get **information** from Cloudvision based on container's name.
- Collect list of attached devices.
- Move a devices to an existing container.

Complete documentation available on read the doc

7.1 Known Issue

Due to a change in CVP API, change-control needs to get snapshot referenced per task. Current version of `cvprack` does not support it in version 1.0.1

Fix is available in develop version. To install development version, use pip:

```
$ pip install git+https://github.com/aristanetworks/cvprac.git@develop
```

7.2 Getting Started

```
$ pip install git+https://github.com/titom73/arista-cvp-container-management.git

# Update your credential information
$ cat <<EOT > env.variables.sh
export CVP_HOST='xxx.xxx.xxx.xxx'
export CVP_PORT=443
export CVP_PROTO='https'
export CVP_USER='username'
export CVP_PASS='password'
export CVP_TZ='France'
export CVP_COUNTRY='France'
EOT

# run script (assuming VLANs configlet is present on CVP)
$ arista-cvp-container-management -j actions.json
```

7.3 License

Project is published under BSD License.

7.4 Ask question or report issue

Please open an issue on Github this is the fastest way to get an answer.

7.5 Contribute

Contributing pull requests are gladly welcomed for this repository. If you are planning a big change, please start a discussion first to make sure we'll be able to merge it.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search