Title: CloudGuard Edge Automation for KVM and OpenStack

Author John Michealson

Tech advisor Joe Dillig

**Overview**

This SK details how [cloud-init](https://cloudinit.readthedocs.io/en/latest/) automation within the CloudGuard Edge image for KVM may be leveraged to bootstrap a virtual machine built to run on the KVM hypervisor beginning with R80.20.

**Metadata Input Methods**

**Config Drive**

The most common method of providing metadata is a config drive. A config drive is a predefined file structure of VM configuration information that is contained in a specially labeled ISO then attached at VM creation time. Upon first boot, the OS has a bootstrapping service that searches for this mounted ISO via a ```blkid``` label and\or filesystem type and then reads the metadata from the file structure. It is possible to input basic VM meta information or complex multi-part scripts, service provider logos, etc. via this method. While Check Point references the standards of cloud-init, some variations may exist. The Check Point image with this updated bootstrapping have a naming convention that distinguishes the image automation within the image (openstack, generic).

**Http metadata service**

Virtual machine information may also be polled from an HTTP metadata service within OpenStack KVM platform providers. Be aware that not all OpenStack or KVM operators enable this service. The data available via the HTTP meta-service is the same as what is provided via config-drive (vendor, network, meta and user). The CG IaaS bootstrap service shows preference to a mounted config-drive at first boot; when an attached config-drive is not detected the bootstrap service will attempt to poll the http service for the configuration data. One benefit of Http metadata vs. config drive is that it is *dynamic,* presenting network interface changes to the VM post deployment. Conversely config drive data is only populated by the platform at VM instantiation.

**Requirements for generating config-drive ISO**

**FOLD HERE**

**Check Point OpenStack Labeled QCOW Image Requirements:**

1. ISO label is set as "config-2"
2. The file structure for an OpenStack deployment has the following requirements
   1. The configuration files (vendor, network, meta and user) must be placed in the ISO /openstack/latest/\*” directory by the platform.
   2. vendor\_data.json
      1. Check Point does not utilize this file
   3. network\_data.json
      1. May specify DHCP or static interface configurations per interface.
      2. If configured as DHCP, the bootstrapping automation polls for a DHCP provided address on specified interfaces, consumes that DHCP address, and statically assigns it to the interface.
   4. meta\_data.json
      1. SSH Key
      2. Virtual machine name is provided by the platform. This VM name is consumed as the OS hostname unless a hostname is specified in user\_data
   5. user\_data
      1. 16k file limit
      2. Description of additional data provided in **user\_data section**. ADD LINK TO “”user\_data” SECTION BELOW

**FOLD HERE**

**Check point Generic Labeled QCOW Image Requirements**

1. ISO label must be set to “cidata”
2. ISO must be type “ISO9660”
3. The ISO file structure has the following requirements
   1. user\_data file can be placed anywhere within the ISO (CP recommends the root directory)
   2. 16k file limit
   3. Description of additional data provided in user\_data section. ADD LINK TO “”user\_data” SECTION BELOW
4. Generic images poll for DHCP

FOLD HERE

**Check Point “user\_data” Customization Formats**

1. This section applies to both OpenStack and Generic images
2. User\_data may be provided via HTTP metadata service or config drive; not all operators enable platform HTTP metadata services
3. Config Drive ISO file structure has the following options
   1. The CP bootstrapping service will parse user\_data and determine the format of the scripts to parse automatically
   2. user\_data has no format requirement besides properly following the convention of the chosen file format
   3. The order of operation for the CP bootstrapping service ( aka “Just deplOy Everything” ) is MIME, cloud-config and bash
      1. In the event MIME or cloud-config are not detected, the user\_data is processed as a bash script.
   4. user\_data may be presented as cloud-config in a yaml format ADD LINK TO ”yaml example” SECTION BELOW
   5. user\_data may be presented as bash ADD LINK TO “”bash example” SECTION BELOW
   6. user\_data may be presented as multi-part mime format
      1. This method supports text/cloud-config and text/x-shellscript
      2. An example of when to use multi-part mime is terraform inputs. [See this link.](https://www.terraform.io/docs/providers/template/d/cloudinit_config.html)
   7. For complex configurations using a mixture of cloud-config and bash is it suggested to utilize multi-part mime.

FOLD HERE

**User\_data cloud-config gateway example**

The minimum group of settings required for proper cloud-config deployment are:

Blink\_config

Ssh\_authorized \_keys

The following example will be updated as features are added to the automation. Simply removing a stanza will cause the setting to remain at default.

#cloud-config

blink\_config:

configure: "true"

gateway\_cluster\_member: "false"

download\_info: "true"

upload\_info: "true"

ftw\_sic\_key: "vpn123"

ssh\_authorized\_keys:

- ssh-rsa AAAAB3NzV1zfVGaJD801Xt6EiQ2LWPEwVc3e5GsCkCgWyBb6HLkMyR0VLZzM7QLrXJgcC/ dev@dev

- ssh-rsa /8tiDGO2Ngq5O59igrKM/Q8X+yFKakISNbOg+iV8bZsPdLKfIjpb1HELFKe+eFijmbqzT

system:

hostname: CloudGuard

domainname: labnet.com

dns1: 1.1.1.1

dns2: 8.8.8.8

dns3: 4.2.2.2

ntp1:

address: ntp.checkpoint.com

version: 4

ntp2:

address: ntp2.checkpoint.com

version: 4

# Configure interfaces per OpenStack or Generic image instructions

interfaces:

- name: eth0

ipv4-address: 192.168.1.35

subnet-length: 24

- name: eth1

ipv4-address: DHCP

routing:

static:

- dst: 4.2.2.4/32

nexthop: 1.0.1.1

- dst: 1.1.1.1/32

nexthop: 10.0.2.1

- dst: 4.3.2.0/24

nexthop: eth0

- dst: 4.3.2.8/32

nexthop: blackhole

cpusers:

- name: admin

shell: /bin/bash

password-hash: $1$qgt/oJIioXKCY.

- name: expert

password-hash: $1$qgtM4UIaoXKCY.

- name: john

password-hash: $1$qgtM4UcQaoXKCY.

shell: /bin/cli.sh

ssh\_authorized\_keys:

- ssh-rsa AAAAB3ISNbOg+iZsPdLKfIjpb1HELFKe+eFijmbqzT

- name: joe

password-hash: $1$qgtM4UZV$BJIiyaoXKCY.

shell: /bin/bash

ssh\_authorized\_keys:

- ssh-rsa AAAAB3ZKBGgWyBb6HLkMyR0VLZzM7QLrXJgcC/ dev@dev

clishcmd:

- 'set as 6408.45314'

- 'set as 6408.45314'

- 'set bgp external remote-as 6408.45312 on'

- 'set bgp external remote-as 6408.45312 local-address 100.100.1.3 on'

- 'set bgp external remote-as 6408.45312 description INSIDE'

- 'set bgp external remote-as 6408.45312 peer 100.100.1.1 on'

- 'set bgp external remote-as 6408.45312 peer 100.100.1.1 ping on'

- 'set bgp external remote-as 6408.45313 on'

- 'set bgp external remote-as 6408.45313 local-address 100.100.1.13 on'

- 'set bgp external remote-as 6408.45313 description OUTSIDE'

- 'set bgp external remote-as 6408.45313 peer 100.100.1.11 on'

- 'set bgp external remote-as 6408.45313 peer 100.100.1.11 ping on'

write\_files:

- path: /home/admin/write\_files

permissions: '0644'

owner: admin:root

content: |

My file content

- path: /home/admin/base64\_write\_files.jpeg

permissions: '0644'

owner: admin:root

encoding: b64

content: |

/9j/4AAQSkZJRgABAQIAJgAmAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB

AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEB

AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAARCAARACgDAREA

5peaMhIa91kjqGLqz1UsG9Kxak7yfQldMKzxb3XNmsxEEHZm3I7ihjxrlw0pLNDneVDlrVc0pNFW

hxtDiqSmnDXRVoIhQG9giAm1zPEcfUYEJZ4Y7EBmbL4fw4X9aJ+sS+Mus+8s+7P8ivNyL+FY/l5P

+YrxlNn4ir7yr/NGYYf7Uv8Asf0eHxkzvYj+bHnzwIZF/fGh/wA26/6WJ4ylyf6X+lvm7GDH1g/w

K4D/ACq4f9GuHjMvP//Z

**User\_data bash Gateway example**

While SK sk120193 describes blink operations in detail, the blink function within the Generic and OpenStack image accepts ONLY the options described in the following example.

~~~~ Code block

#!/bin/bash

adminHash='$1$YuBfLw54$4k6C/hGD6vdRspHSs59ny.'

sicKey="vpn123"

allowUpload="true"

allowDownload="true"

clusterMember="false"

#

echo "begin blink" >> /var/log/run-cloud-user-data

blink\_config -s "gateway\_cluster\_member=${clusterMember}&ftw\_sic\_key=${sicKey}&upload\_info=${allowUpload}&download\_info=${allowDownload}"

echo "end blink" >> /var/log/run-cloud-user-data

#

~~~~ Code block

Adding simple clish commands to the same user\_data allows more complex configurations

~~~~ Code block

clish -c "lock database override"

#clish -c "set user admin shell /bin/bash" -s

clish -c "set user admin password-hash ${adminHash}" -s

clish -c "set interface eth0 ipv4-address 120.120.120.113 mask-length 24" -s

clish -c "set interface eth0 state on" -s

clish -c "set static-route default nexthop gateway address 120.120.120.29 priority 1 on" -s

clish -c "set hostname CP\_Hostname" -s

echo "end config" >> /var/log/run-cloud-user-data

~~~~ Code block

User\_data cloud-config SMS example

At this time of writing, SMS images Generic and OpenStack are NOT blink enabled. These images the utilize the same deployment mechanism Check Point has embedded for some time. As such, this config\_system stanza will accept any input that is relevant via sk69701

Non-Blink All-In-One (SMS + GW) or Mgmt Images

The config\_system

**~~~~Code Block**

#cloud-config

config\_system:

  configure: "true"

  hostname: SMS

  mgmt\_admin\_name: "admin"

  mgmt\_admin\_passwd: "vpn123"

  mgmt\_gui\_clients\_radio: any

  install\_security\_managment: "true"

  install\_security\_gw: "false"

  install\_mgmt\_primary: "true"

  install\_mgmt\_secondary: "false"

  download\_info: "true"

  upload\_info: "true"

  mgmt\_gui\_clients\_radio: any

  primary: 1.1.1.1

  secondary: 8.8.8.8

  tertiary: 4.2.2.1

  ntp\_primary: ntp.checkpoint.com

  ntp\_primary\_version: 4

  ntp\_secondary: ntp2.checkpoint.com

  ntp\_secondary\_version: 4

**ISO Creation Example**

``` genisoimage -output /tmp/myUserData-gw.iso -volid config-2 -joliet -r ~/cloud-init/myUserData/openstack.gw/ ```

``` mkisofs -r -V config-2 -o /tmp/myUserData-gw.iso /tmp/cloud-init/myUserData/openstack.gw/

```xorriso -as mkisofs -o /tmp/myUserData-gw.iso /tmp/cloud-init/myUserData/openstack.gw/

**Employee only area:**

Flow diagram for 2020 CG IaaS bootstrapping