Deploying a Cisco CSR 1000v VM Using Custom Data

When you deploy a Cisco CSR 1000v Virtual Machine instance on Google Cloud Platform, you can optionally choose to use the **Startup Script** section on the VM creation console to provide custom data. You can also use the CLI to access the custom data to achieve various automation goals. The custom data in GCP allows you to run Cisco IOS XE configuration commands, install Python packages in guestshell on Day0, run scripts in guestshell on Day0, and provide licensing information to boot the CSR 1000v instance with a desired technology package.

Releases Supported

You can deploy a Cisco CSR 1000v VM using a custom data only on Cisco IOS XE Gibraltar 16.12.1 or later releases.

Editing the Custom Data

To edit the custom data, configure the following properties:

- IOS configuration
- Scripts
- Script credentials
- Python package
- Licensing

You can place the properties in a file in any order. The following property descriptions specify dependencies between the properties, if any. See the example bootstrap files at: https://github.com/csr1000v/customdata-examples.

After defining the custom data properties, you can access the startup script or the custom data file using the CLI as described in the *Accessing the Custom Data* section.

Configuring the IOS Configuration Property

If you want to bootstrap the IOS configuration on Day0, configure the IOS Configuration property. See the following IOS configuration example:

```
Section: IOS configuration
hostname CSR1
interface  GigabitEthernet1
description “static IP address config”
ip address 10.0.0.1 255.255.255.0
interface GigabitEthernet2
description “DHCP based IP address config”
ip address dhcp
```

After the first line that reads `Section: IOS configuration`, you can enter a list of Cisco IOS XE configuration commands that you want to execute, on the Cisco CSR 1000v router.

When you run this command, the preceding IOS configuration is applied to the CSR 1000v router running on GCP, on Day0.

Configuring the Scripts Property

Scripts property helps you automate the deployment of your CSR1000v instance. If you want to run a Python or a Bash script on Day0 under the guestshell context, provide the public URL and arguments of the python or the bash script in Scripts property.

A script must include a piece of code that includes the shebang (!) character in the first line of the script. This line tells Cisco IOS-XE which script interpreter (Python or Bash) you must use to parse the script code. For example, the first line of a Python script can contain `#!/usr/bin/env python`, while the first line of a Bash script can contain `#!/bin/bash`. This line allows the Python or the Bash script to run as executable code in a Linux environment.

When you execute the script, the script runs in the guestshell container of the Cisco CSR 1000v instance. To access the guestshell container, use the **guestshell** EXEC mode command. For more information on guestshell command, see the Programmability Configuration Guide.

To configure the Scripts property, use the following format:

```
Section: scripts
public_url <arg1> <arg2>
```

In this script, the first line of the property should read `Section: Scripts`.

In the second line of the property, enter the URL of the script and the script's arguments. The script can be either a Python or a Bash script. The script is run in guestshell in the first boot when you upload the custom data file, when you create the CSR1000v instance.

To view more examples of the scripts, see "scripts" at: https://github.com/csr1000v/customdata-examples. Also, refer to the following examples:

Example 1

```
Section: Script
https://raw.githubusercontent.com/csr1000v/customdata-
examples/master/scripts/smartLicensingConfigurator.py --idtoken "<token_string>" --throughput
 <throughput_value>
```

The two lines in the scripts property retrieve the `smartLicensingConfigurator.py` script from the `customdata-examples` repository at the specified URL. The script runs in the guestshell container of the Cisco CSR 1000v with the arguments `idtoken` and `throughput`.

Example 2

```
Section: Scripts
ftp://10.11.0.4/dir1/dir2/script.py -a arg1 -s arg2
```

These two lines in the Scripts property retrieve the `script.py` script from the FTP server with the IP address 10.11.0.4, and runs the script with the `./script.py -a arg1 -s arg2` Bash command in the guestshell container of the Cisco CSR 1000v instance using arguments arg1 and arg2.

Note If a script in the Scripts property requires a Python package that is not included in the standard CentOS Linux release (the CentOS Linux release that is currently used by the guestshell is CentOS Linux release 7.1.1503), you must include information about the Python package in the Python package property. For more information, see: Configuring the Python package Property, on page 4.

Before you access the custom data and run the Bash or the Python script, Cisco recommends that you test the URL that you intend to use, using the Scripts property. You can test `ftp://10.11.0.4/dir1/dir2/script.py -a arg1 -s arg2` by first running the curl software tool to download the script file. In the guestshell, enter the curl command as shown in the following example:

```
curl -m 30 --retry 5 --user username:password
ftp://10.11.0.4/dir1/dir2/script_needs_credentials.py.
```

If the curl command is successful, a copy of the Python script is downloaded, which verifies whether the URL is correct.

Configuring the Script Credentials Property

If you have specified an FTP server in the Script property, and the server requires a username and password credentials, specify the credentials using the Script credentials property.

Note If you can access the FTP server anonymously, you need not use the Script credentials property.

Configure the Scripts property with a URL and parameters that match those in the Script credentials property. To configure the Script credentials property, use the following format:

```
Section: Script credentials
public_url <username> <password>
```

Example

```
Section: Script credentials

ftp://10.11.0.4/dir1/dir2/script1.py userfoo foospass
```

The second line in the Script credentials property specifies the values of the username (`userfoo`) and password (`foospass`) credentials for the python script `script1.py`.

Include the name of the FTP server that is also in the Scripts property. An example line in the Scripts property is: `ftp://10.11.0.4/dir1/dir2/script1.py -a arg1 -s arg2`. See example 2 in Configuring the Scripts Property, on page 2.

Configuring the Python package Property

If a Python package is required by a script in the Scripts property and it is not part of the standard CentOS Linux release 7.1.1503, you must include information about the package in the Python package property. By including the Python package property in the bootstrap file, you ensure that the Cisco CSR 1000v downloads and installs the required Python package before the custom data file that you specified in the Scripts property.

Configure Python Package Property

To configure the Python package property, use the following format:

```
Section: Python package
package_name [ version ] [ sudo ] { [ pip_arg1 [ ..[ pip_arg9] ] ] }
```

The arguments: *version*, **sudo**, and *pip_arg1* to *pip_arg9* are optional. You must put the arguments to the pip command between "{" and "}" braces.

If you specify the *version* argument, a specific version number is downloaded.

If you specify the *sudo* argument, the package is downloaded as a sudo user.

Configuration Examples

Example 1

```
Section: Python package
```

```
ncclient 0.5.2
```

In this example, the second line of the Python package property specifies that the *package_name* is "ncclient" and the *version* is "0.5.2". When the bootstrap file is uploaded, version 0.5.2 of the ncclient package is installed in the guestshell container of the Cisco CSR 1000v.

Example 2

```
Section: Python package
```

```
csr_gcp_ha 3.0.0 sudo {--user}
```

In this example, the second line of the Python package property specifies that the *package_name* is "csr_gcp_ha" and the *version* is "3.0.0". When the bootstrap file is uploaded, version 3.0.0 of the csr_gcp_ha package is installed in the guestshell container of the Cisco CSR 1000v. The following command is executed as a sudo user: `pip install csr_gcp_ha=3.0.0 --user`.

Configuring the License property

Configure the license property to specify the license technology level for the Cisco CSR 1000v instance.

- Enter the first line of the property in the format: `Section: License`.
- Enter the second line of the property, which specifies the tech level of the license, using the following format: **TechPackage:***tech_level* .

Note Ensure there are no spaces between "TechPackage:" and the *tech_level*. The possible *tech_level* values include: ax, security, appx, or ipbase.

Ensure that *tech_level* is in lowercase.

Configuration Example

```
Section: License

TechPackage:security
```

Accessing the Custom Data

To run the custom data as a file by using the CLI, execute the following script:

Accessing the custom data file using the CLI

To run the custom data as a file by using the CLI, execute the following script:

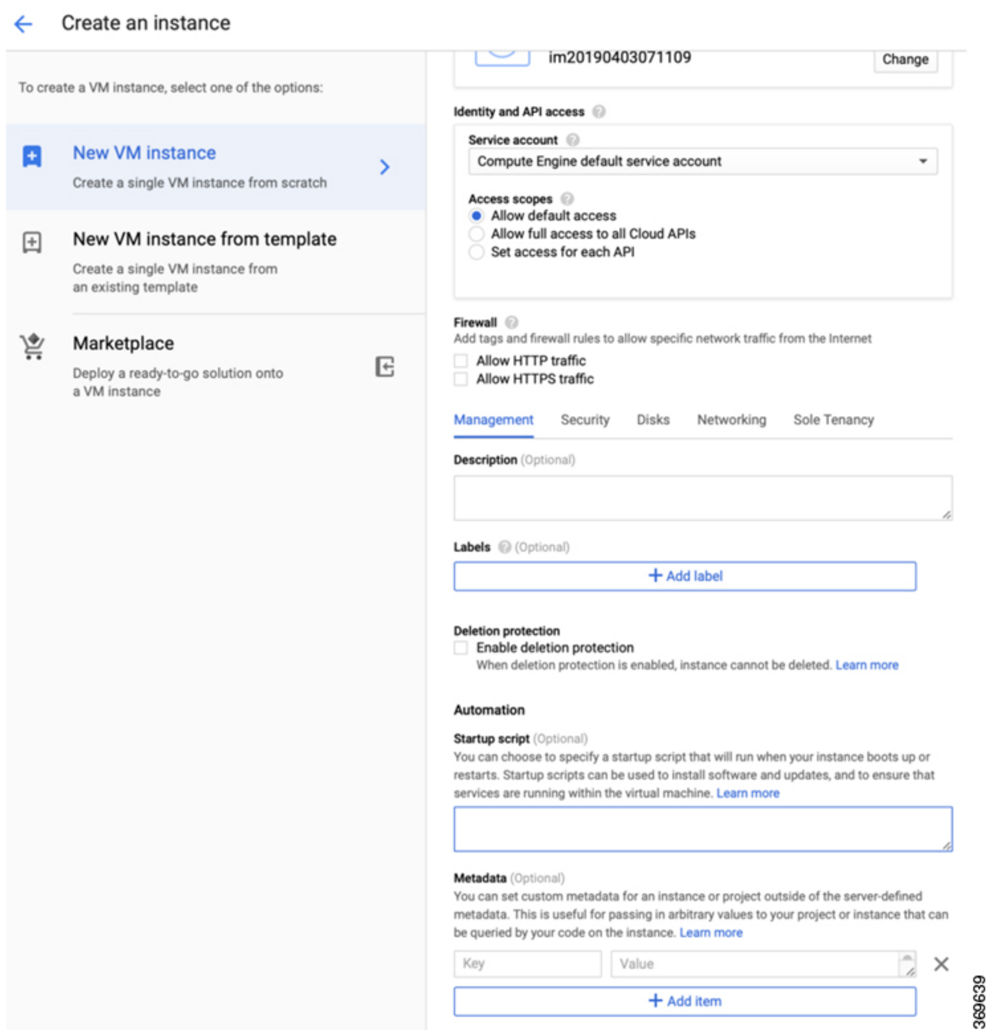
```
gcloud compute instances create <vm_name> --metadata-from-file=startup-script=Customdata.txt
 --image  <image_name>
```

When you execute this command, a Cisco CSR 1000v VM is created. The router is configured using the commands in the file: "Customdata.txt".

Accessing the custom data from the console

To access the custom data from the console, log in to the GCP console. Click **Compute Engine**, and select **Create an Instance**.

On the New VM instance screen, click **Management** > **Startup Script**.


Create an instance
To create a VM instance, select one of the options:
New VM instance
Create a single VM instance from scratch
New VM instance from template
Create a single VM instance from an existing template
Marketplace
Deploy a ready-to-go solution onto a VM instance
Change
Management
Security
Disks
Networking
Sole Tenancy
+ Add label
Key
Value
+ Add item
369639

im20190403071109

Identity and API access

Service account: Compute Engine default service account

Access scopes
- [✓ Allow default access]
- Allow full access to all Cloud APIs
- Set access for each API

Firewall
Add tags and firewall rules to allow specific network traffic from the Internet
- Allow HTTP traffic
- Allow HTTPS traffic

Description (Optional)

Labels (Optional)

Deletion protection
- Enable deletion protection
When deletion protection is enabled, instance cannot be deleted. Learn more

Automation

Startup script (Optional)
You can choose to specify a startup script that will run when your instance boots up or restarts. Startup scripts can be used to install software and updates, and to ensure that services are running within the virtual machine. Learn more

Metadata (Optional)
You can set custom metadata for an instance or project outside of the server-defined metadata. This is useful for passing in arbitrary values to your project or instance that can be queried by your code on the instance. Learn more

The startup script specified in this field runs every time you bootup or restart your CSR 1000v instance.

Verifying the Custom Data Configuration

After you run the custom data script, the VM is created and the configuration commands are executed. To verify the same, use the following commands and scripts:

- **show version**: To help determine if the license property worked, in Cisco IOS XE CLI on the CSR 1000v, enter the **show version** command. For example, the output displays a reference to the security license.
- To see if errors occurred after running commands in the scripts property, look at the `customdata.log` file in the `/bootflash/<cloud>/` directory. The *scriptname*.log file stores any output that is sent to STDOUT by the script.
- To verify whether the Python property worked, enter the `pip freeze | grep <package-name>` command from the Guestshell to view the currently installed Python packages. Here, *package-name* refers to the package that you are specifically searching for.

- To verify the Cisco IOS XE commands in the IOS Configuration property, run the **show running-configuration** command.