CANONICAL

Greengrass Snap Release Notes

Canonical Devices and IoT, Field Engineering

Version: 1.0
April 1 2019
Ian Johnson <ian.johnson@canonical.com>

Overview

AWS IoT Greengrass version 1.8.0 is the first release to support snaps as a distribution method. The Greengrass snap supports all the same features as a native distribution and is compatible with existing Greengrass groups. However, some deployments may require configuration changes.

This document assumes knowledge of AWS IoT Greengrass in general, and is meant as a supplement to the official AWS IoT Greengrass documentation and walkthroughs available here.

Initial system configuration

Core snap channel

The Greengrass snap requires various interfaces to be connected before it can operate normally. One of these interfaces, `greengrass-support`, was modified to accommodate the 1.8 release of Greengrass and as such, the necessary changes are only available in snapd 2.38 and later.

System user and group configuration

If you require the use of Greengrass groups/lambdas that do not run as root, you will need to add the user/group you wish to use with the snap to the system. The default user and group configured for groups and lambdas are `ggc_user` and `ggc_group` respectively. Note that this configuration is optional; you can always configure your Greengrass groups/Lambdas to run as root such that no additional users or groups are necessary on the system. See AWS documentation for more information on running lambda functions here and for configuring the default user and group for all lambdas in a Greengrass group here.

To add the `ggc_user` user to an Ubuntu Core system use:

```
sudo adduser --extrausers --system ggc_user
```

To add the `ggc_user` user to an Ubuntu classic system omit the `--extrausers` flag and use:

```
sudo adduser --system ggc_user
```

To add the `ggc_group` group to an Ubuntu Core system use:

```
sudo adduser --extrausers --system ggc_user
```

To add the `ggc_group` to a classic Ubuntu system omit the `--extrausers` flag and use:

```
sudo adduser --system ggc_user
```

Installation

Installing the snap

The snap can be installed from the snap store using the `snap install` command:

```
snap install aws-iot-greengrass
```

If you are installing locally from a file and do not have the associated assertions you need to use the `--dangerous` flag:

```
snap install --dangerous aws-iot-greengrass*.snap
```

The snap has one daemon/service that is disabled on installation because the service needs to have certificates and configuration specific to a user's account before it can run successfully. You can see the service disabled with the `snap services` command:

```
$ snap services aws-iot-greengrass
Service                          Startup   Current   Notes
aws-iot-greengrass.greengrassd  disabled  inactive  -
```

Connecting interfaces

If the snap was not installed from the store directly and was instead installed through a file with the `--dangerous` flag, the relevant interfaces for Greengrass will not be automatically connected and will need to be manually connected. This can be done with the `snap connect` command or via a `POST` request to the `/v2/interfaces` REST endpoint in the snapd REST API. The minimal necessary interfaces for the snap to function correctly are:

- `greengrass-support`
- `process-control`
- `system-observe`
- `network`
- `network-bind`
- `network-control`

Regardless of whether the snap was installed from a snap file or from the store, the `network` and `network-bind` interfaces will be automatically connected.

Note that the `greengrass-support` interface is critical to the operation of the snap such that it needs to be connected before any other interfaces are connected. If the `greengrass-support` interface is ever disconnected from the snap, the interface will need to be re-connected and the system rebooted in order for the Greengrass snap to work again. As such, it is recommended never to disconnect the `greengrass-support` plug from the Greengrass snap while the Greengrass snap is being used.

To connect an interface plug via the `snap connect` command, perform the following command for all relevant interfaces:

```
snap connect aws-iot-greengrass:greengrass-support
```

To instead connect these interfaces using the snapd REST API, perform a `POST` request against the `/v2/interfaces` endpoint from a program the `snapd-control` interface with the following JSON body content:

```
{
       "action": "connect",
       "plugs": [{"snap": "aws-iot-greengrass", "plug": "greengrass-support"}],
       "slots": [{"snap": "core", "slot": "greengrass-support"}]
}
```

For example, to perform this POST request with `curl` run:

```
$ sudo curl -X POST --unix /run/snapd.socket http://localhost:/v2/interfaces --data
'{"action": "connect","plugs": [{"snap": "aws-iot-greengrass",   "plug":
"greengrass-support"}],"slots": [{"snap":"core","slot": "greengrass-support"}]}'
```

Other interfaces that may be useful or convenient for the operation of the snap but are not necessary include:

- `camera`
- `dvb`
- `gpio`
- `gpio-memory-control`
- `hardware-random-control`
- `home`
- `i2c`
- `iio`
- `joystick`
- `opengl`
- `optical-drive`
- `raw-usb`
- `removable-media`

- `serial-port`
- `spi`

Many of these additional interfaces are necessary in order for Lambda functions to access devices. This is detailed more in the section Local Resource device access below.

Configuration of the snap

After the snap is installed it must be configured with a user's Greengrass configuration and certificates specific to their account. This mechanism is provided as a `snap set` call, accessible either using the snap command or using the snapd REST API. When a greengrass group is initially created, a user is provided with the opportunity to download the certificates and configuration file as a gzipped tar file with either a tar.gz or tgz file extension. The name of this file is to be provided to the `snap set` command or REST API endpoint. Specifically, the file contents are not stored in the snap configuration for security reasons; the certificates are stored in $SNAP_DATA and only accessible by root or from processes in the snap and not to other users. Storing the key data directly inside the snap configuration would expose these keys to other users on the system that are authenticated with snapd and may or may not have root access.

Note that both the `home` and `removable-media` interfaces are used only for convenience when configuring the snap with this archive file. If you do not connect these interfaces, the archive file will first need to be copied into a snap-readable location such as `$SNAP_DATA` or `$SNAP_COMMON`. If you connect these interfaces, note that the `home` interface is connected with the `read: all` attribute specified, which allows a process running as root inside the snap to read from the home folders of all other users on the system. This allows the configure hook, which runs as root, to read non-root home folders and thus read the archive file even if the file is not strictly in the root user's home interface.

To provide the combined configuration and certificates archive file with the `snap set` command run:

```
snap set aws-iot-greengrass gg-certs=/path/to/the/file.tgz
```

To provide the file using the snapd REST API from a management agent snap using `snapd-control`, perform a `PUT` request against the `/v2/snaps/aws-iot-greengrass/conf` endpoint with the following JSON content as the HTTP request body:

```
{
   "gg-certs" : "/path/to/the/file.tgz"
}
```

For example, to set this with the `curl` command run:

```
$ sudo curl -X PUT --unix /run/snapd.socket
http://localhost:/v2/snaps/aws-iot-greengrass/conf --data
'{"gg-certs":"/path/to/the/file.tgz"}'
```

After this is done, the `greengrassd` service will start running automatically if the configuration is valid. This can be validated with `snap services`:

```
$ snap services aws-iot-greengrass
Service                         Startup  Current  Notes
aws-iot-greengrass.greengrassd  enabled  active   -
```

Deploying Lambda Functions

After the snap is configured and running on the device, the associated Greengrass group must be deployed using either the AWS REST API, the AWS CLI, or the web UI. See AWS documentation for more details on how to do this here using the web UI and here for using the CLI or the REST API.

Additionally, some Lambda functions can use Local Resources as documented here. There are additional steps that must be taken in order to use Local Resources with the Greengrass snap.

Local Resource device access

Some Lambda functions may require access to direct linux devices such as `/dev/gpio0` or `/dev/ttyUSB0`. By default the Greengrass Lambda will not have access to these devices and access must be provided to the Lambda in AWS as documented here. In addition, the Greengrass snap is confined by AppArmor to not allow these accesses by default, but the snap can be configured to allow these accesses by connecting an interface. Only devices which are supported with a snapd interface that is declared by the snap can be configured for access. Some of these interfaces are generic and connect to the system's default slot such as the `raw-usb` interface. This would allow access to `/dev/ttyUSB0` as well as any other device matching the pattern "`/dev/tty{USB,ACM}[0-9]*`". For these devices, connect the interface to the implicit core snap slot using the `snap connect` command as in:

```
snap connect aws-iot-greengrass:raw-usb
```

Or using the snapd REST API by performing a post as shown previously in Connecting Interfaces.

Some devices, however, need a specific system slot in order to allow access, in which case the command is slightly different. In order to use a device such as `/dev/gpio0`, a slot exposing `/dev/gpio0` must be exposed by the gadget snap on an Ubuntu Core system and the connection will only allow access to that specific device node. To access another gpio device

such as `/dev/gpio1` another slot must be exposed by the gadget snap and connected independently from the `/dev/gpio0` connection. To allow access to a specific gpio device such as BCM pin 0 (which is exposed as `/dev/gpio0`) on a Raspberry Pi with a gadget snap named `pi3`, the gpio interface plug that Greengrass specifies would be connected as follows:

```
snap connect aws-iot-greengrass:gpio pi3:bcm-gpio-0
```

Lastly, these connections can be specified in a gadget snap using the connections stanza in the `gadget.yaml` of the gadget snap. See documentation of the gadget snap here for more details.

Local Resource volume access

Some Lambda functions may require access to other files not already contained in the Lambda. Thus, Lambda functions can be configured using Local Resources to access file content using volumes. Snaps are also confined to not access arbitrary files from the host filesystem, so local resource files must also be delivered to the system using snaps. The snap exposes a `content` interface which can be connected to another snap to share the other snap's files with the Greengrass snap, thus enabling the Greengrass snap to provide access to the other snap's files to Lambda function using local resources.

The first task to provide volume access is to build a snap with the files in a snap. Then the snap needs to declare a `content` interface slot named `local-resources` that specifies the file or directory to share from the snap to the Greengrass snap. For example, with a snap that has files in a `myfiles` directory that should be accessed by the Greengrass snap, declare a slot in the `snapcraft.yaml` of your other snap using:

```
slots:
  local-resources:
    interface: content
    content: local-resources
    source:
      read: [$SNAP/myfiles]
```

The source specification is specifically important because otherwise, all of the files from inside the `myfiles` directory will be placed into the root folder of the target and potentially conflict with other snaps sharing files with the Greengrass snap. Using the source specification allows the `myfiles` directory to be placed verbatim inside the target as explained below.

After installing this snap on the same system as the system running Greengrass as a snap the interface plug in, the Greengrass snap must be connected to the interface slot in the other

snap. This can be done with the `snap set` command like this:

```
snap connect aws-iot-greengrass:local-resources other-snap:local-resources
```

This can also be connected any other way interfaces can be connected such as using the snapd REST API or using auto-connection rules in the gadget snap. After this has been connected, the next step is to configure the Lambda function in AWS.

Due to the usage of the `content` interface, the location of the files from the perspective of the Greengrass snap is not exactly the same as the location of the files in the other snap. Thus, a different path needs to be used when configuring the local resource volume access for the Lambda function. Files shared via the `content` interface with the Greengrass snap will reside inside the `/var/snap/greengrass/common/ggc-local-resources` directory such that the `myfiles` directory will appear inside the Greengrass snap as `/var/snap/greengrass/common/ggc-local-resources/myfiles`. When configuring the local resource access for a Lambda to access this directory, you would specify the volume source path as `/var/snap/greengrass/common/ggc-local-resources/myfiles`.

For more information and help using Greengrass as a snap, please visit the snapcraft forum at forum.snapcraft.io. For more information about AWS IoT Greengrass, please visit aws.amazon.com/greengrass.