Controlling application start/stop using a profile concept

The proposal consists of two parts:

1. Associating (tagging) each application instance with zero or more profiles (which are strings chosen by the user) and being able to have the controller switch profiles by sending a default_profile string. If the default_profile is set by the controller, then only applications which are tagged with that profile will be running (and the applications must also be marked as Activate in the API.) This allows for stopping one set of applications and starting another set.
2. A new concept of a local_profile_server, which is a http/https API server (typlically deployed as an app instance on EVE). If this such a server is specified by the controller, then EVE will query its API endpoint for an override_profile. If the override_profile is not empty, then it will be used instead of the default_profile received from the controller. This allows for some emergency local change of the set of running application instances even when the controller is unreachable.

The draft API for this is specified in https://github.com/lf-edge/eve/pull/2073

Conceptual model

We introduce a tag/profile for each application instance, and in the EVE API we also introduce a default_profile field. The profile is a string which the user can pick. User's might want to use a handful of different profiles. Each application can be tagged with more than one profile.

The app instances which have a profile tag which match the received default_profile will be considered for running and those that do not match any will be halted. ("Considered" since we also take into account the individual stop/start in the UI aka the Activate boolean in the EVE API for the app instance as well.)

In addition to the above default_profile from the controller, EVE can be configured to talk to a local profile server, which is accessible on the network (it could be deployed in the enterprise, or could be an app instance deployed on the edge node). In that case EVE will request profile override configuration using a new API specified below. The response payload from that API will be a protobuf message containing an optional profile field.

When the local profile server is configured, and it responds with a non-empty string in the profile field, then EVE will use that as the profile to select which application instances to shutdown and start and ignore the default_profile received from the controller.

Hence the profile override can be disabled by either having the profile server return the empty string, or deconfigure the local profile server. In either case, EVE will switch to using the default_profile received from the controller. Note that a (temporarily or permanently) unreachable local profile server will not result in switching back.

EVE will report in info messages whether and which profile override is used.

Out of scope

The implementation of a local profile server is out of scope, as there might be different user interfaces to the local profile server. This work will merely define the GET API for the profile, and the method by which EVE is configured to access the local profile server. [But see Testing section below.]

Once the local profile server has asserted an override of the profile, it is up to the local profile server to remove that override when it is no longer needed, to ensure that the controller can fully control the application instances.

EVE behavior

The check for Activate being true will now be gated by (current profile not set, or current profile matches one of the app instance's profile.)

The current_profile is set to the override_profile if not empty, otherwise it is set to the defaut_profile from the controller.

If a local profile server has been configured, then EVE will attempt a GET request to that server with the same frequency as it requests configuration from the controller. It will retain the most recently received string as the override_profile (and report that in the info API message so that the UI can display the existence of an override in a prominent place.) Presumably the override_profile should be persisted across reboots so EVE can do the right thing on power up.

Should the local_profile_server field be cleared in the config API, then EVE will clear the override_profile.

EVE will verify that the server_token in the protobuf message matches the profile server token received from the controller.

EVE API Additions

A few items in the EVE API:

- A list of profiles for each app instance in the config API
- A default_profile in the device config API
- A local_profile_server string (with hostname or IP followed, then ":" then IP address in URL syntax) and a profile_server_token in the config API.
- A override_profile in the device info API

And a new Profile Override protobuf API defining one message:

```
message ProfileOverride {
   profile string = 1;
   server_token string = 2;
}
```

See **https://github.com/lf-edge/eve/pull/2073**

The local_profile_server string can be an IPv4, IPv6, or hostname followed by an optional ":" and a port number. Note that to handle IPv6 addresses it needs to check and allow for https://datatracker.ietf.org/doc/html/rfc3986#section-3.2.2 which means allowing syntax like

```
[fe80::1]:1234
10.1.1.1:1234
Hostname:1234
[fe80::1]
10.1.1.1
Hostname

Together with a local_profile_server one should be able to specify a profile_server_token.
```

The UI needs to display the overide_profile in a prominent place since this indicates that an override is in place even when connected to the controller. (Perhaps the case when override_profile == default_profile would be normal and need not be flagged.)

Depending on whether we reuse the tag capability in the UI or define some new profile setting for the app instances, zedcloud has different types of work to feed the application profiles to the EVE API.

Security

The communication to the local profile server will be done using http but on a specified port so that an outbound ACLs for port 80 will not accidentally allow such traffic to leave the edge-node on the Ethernet ports. Users will have to allow this port either out the Ethernet ports or to an app instance deployed locally on EVE. [TBD: the ip rules currently do not make the IP addresses on a local network instance reachable by EVE so need some tweak here.]

In addition, we require that the profile server be configured with a server_token (a random string) which EVE will verify matches the profile_server_token. That makes it harder for a network attacker to inject responses even when the ACLs have been configured to accept all traffic.

Open Issues

- [Closed] If the app instance is stopped from the UI i.e., Activate=false, the profile setting has no effect.
- [Closed] Each app instance can specify multiple profile tags, and if any one matches the current profile (and Activate is set) the app will run.
- [Closed] An empty profile string must match for all app instances to avoid having the introduction of the profile field in EVE result in all applications being halted when the controller sends no default_profile string.

Testing

To test the above it makes sense to implement a very basic local profile server app instance in the form of a container which each time the GET method is handled will look for a USB stick with a FAT filesystem and read a file called "profile" on that USB stick. If the file is not found it will return an empty string, otherwise the content of the file.