Object Model / Flow of Operation

The Regional Controller's Object Model

The Regional Controller performs its work by maintaining an Object Model. The Object Model consists of a number of objects, described below, which model the machines, software, and lifecycles of the various Akraino deployments that the Regional Controller controls. The collection of all the objects that the Regional Controller maintains in this model, as well as their relationships is know as the *Akraino Universe*.

Most objects in the Object Model have the following in common:

- A *uuid*, meant to be a unique, internal identifier for the object. The *uuid* is assigned by the Regional Controller.
- A *name*, meant for human consumption. The *name* is required, and will normally be unique for any class of object.
- A *description*, also for human consumption, to provide more detail about the object. The *description* is optional.

Hardware Profiles

A Hardware Profile describes a specific model of hardware that will be used within an Akraino Universe (for example *Dell PowerEdge R740 Gen 14*). Because the amount and type of information that can describe a particular piece of hardware can vary widely, a Hardware Profile provide a *yaml* attribute where this information can be stored.

Nodes

A single, identifiable machine. A node may be unassigned, or it may belong to one and only one EdgeSite. A node must make reference to a hardware profile that describes the hardware of which the node consists. In addition, each Node has a *yaml* attribute, which can be used to store other information about the node (e.g. the nodes lat/long location for use in a GUI, or the rack ID of the rack the node is mounted on).

EdgeSites

A collection of one or more Nodes upon which a POD may be installed. Any specific Node may belong to one and only one Edge Site. Edge Sites in turn must belong to at least one, and potentially several *Regions*.

There is no requirement that an Edge Site consist of homogeneous hardware. However, most Blueprints are likely to require that all nodes in an Edge Site be the same hardware. This would be detected at the point in time where a POD is created (see below).

Regions

A region is a grouping mechanism to collect one or more Edge Sites, or other Regions, together for management purposes. Regions may be used to group Edge Sites by physical location (e.g. *US NorthEast*) together. They may also be used to perform logical groupings (e.g. *RAN Sites*).

Regions form a tree, similar to a UNIX filesystem, with the topmost Region being the *Universal Region*. A Region may have only one parent Region. The Universal Region has itself as its parent.

Blueprints

A collection of software that can be deployed to an *empty* Edge Site, as well as the supporting software that the Regional Controller uses to manage the lifecycle of the Edge Site. A Blueprint in the Object Model consists of:

- a *version* attribute, a string in *Semantic Versioning* (X.X.X) form, describing the version of the Blueprint. This is required.
- a *yaml* attribute describing everything else. This is required. This is very lightly defined at the moment, but is expected to include:
    - locations (URLs) of various pieces of software required to deploy the Blueprint
    - locations of downloadable workflows to run inside the Regional Controller
    - definitions of any hardware requirements that a specific Blueprint may have

PODs

(Point of Delivery) A specific deployment of a *Blueprint* upon an *Edge Site*. The act of creating a POD, causes the Blueprint to be deployed on the Edge Site. The Blueprint must be valid for the Hardware Profiles and other characteristics of the Edge Site, in order to be deployed. In addition, the Edge Site must not be in use by another POD at the time this POD is created. At the time of creation, the YAML that is uploaded by the user is verified against the schema that is specified in the Blueprint. The Regional Controller then tells the WorkFlow Engine to deploy the Blueprint on the Edge Site using, as parameters:

- the UUID of the newly created POD
- the Blueprint
- the definition of the Edge Site in the database
- the YAML content from the POST request
- any other parameters from the POST request

to control the deployment.

Updates to an existing POD are performed via PUT requests to the POD's URL. There is a separate section in the Blueprint specifying the input schema, workflow, and data file components required for each type of update.

Deleting a POD causes the the Blueprint to be removed from the Edge Site, and places the Edge Site back in an *unused* state.

If any operation (create/update/delete) is missing from the specification in the Blueprint, the corresponding operation is disallowed by the Regional Controller. Naturally, if *create* is missing, then the Blueprint can never be deployed.

User

An individual user of the Regional Controller. A user is identified by a user name, a password, and a list of roles. All Regional Controller API operations are logged with an indication of the user who requested the operation.

Because the user database is likely to be maintained externally (e.g. in an LDAP server shared with other services), there is no API to perform the CRUD operations on users.

Session

An authenticated instance of a user connection to the Regional Controller. There may be many Sessions for one User. Sessions have a limited lifetime.

Almost all operations within the API require a session token, which identifies the user and the users' roles to the API. As such, the very first operation a user of the API will perform will be the *Create Session* (POST /api/v1/login) call in the Login API. This creates the Session and its corresponding token.

Role

A set of functionality that can be assigned to one or more Users. Roles allow users to perform specific functions within the API. The roles are hard-coded into the Regional Controller, and are not expected to change often, if at all; as such, there are no CRUD operations defined for the roles. A user of the API can discover what roles s/he has been given via the Login API.

Flow of Operation

An example of how the API would be used to deploy a POD on a bunch of new machines follows. These examples use *curl* as that is the most compact way to show all of the required headers and data elements required. Of course, any other programming language may also be used.

Get a Login Token

You will need an account that has permissions to create Nodes, Edgesites, and PODs (and possibly Blueprints too, if you are defining a new Blueprint). To do this use the Login API as follows (assuming the login is *admin* and password *abc123*):

```
$ curl -v -k -H 'Content-Type: application/yaml' --data '{ name: admin, password: abc123 }' https://arc.akraino.
demo/api/v1/login
```

The API will return a login token (which will expire in one hour) in the *X-ARC-Token* header. This should be passed to all subsequent API calls.

```
X-ARC-Token: YWRtaW4gICAgICAgIDE1NTY1NTgyMjExMzQ4N2NmMGUwNQ==
```

Enumerate the Machines

Assuming the machines to deploy on have not yet been made known to the RC, you would need to use the Node API to add them to the RC database.

Do this with the following API call, once per node:

```
$ YAML='{
  name: nodename,
  description: a description of the node,
  hardware: <hardware profile UUID>
}'
$ curl -v -k -H 'Content-Type: application/yaml' -H 'X-ARC-Token:
YWRtaW4gICAgICAgIDE1NTY1NTgyMjExMzQ4N2NmMGUwNQ==' \
  --data "$YAML" https://arc.akraino.demo/api/v1/node
```

Keep track of the UUID of each node that is returned from the API.

Create an Edge Site

Once the nodes are defined, you need to create an Edge Site (a cluster of nodes). Do this:

```
$ YAML='{
  name: edgesitename,
  description: description of the Edgesite,
  nodes: [ <list of node UUIDs> ],
  regions: [ <list of region UUIDs> ]
}'
$ curl -v -k -H 'Content-Type: application/yaml' -H 'X-ARC-Token:
YWRtaW4gICAgICAgIDE1NTY1NTgyMjExMzQ4N2NmMGUwNQ==' \
  --data "$YAML" https://arc.akraino.demo/api/v1/edgesite
```

Create/Verify the Blueprint

To get the UUID of the Blueprint, you would need to see which Blueprints are installed:

```
$ curl -v -k -H 'Accept: application/yaml' -H 'X-ARC-Token: YWRtaW4gICAgICAgIDE1NTY1NTgyMjExMzQ4N2NmMGUwNQ=='
https://arc.akraino.demo/api/v1/blueprint
```

If the Blueprint you want is missing, you may need to create it:

```
$ YAML='{
  blueprint: 1.0.0,
  name: my new blueprint,
  version: 1.0.0,
  description: description of the blueprint,
  yaml:  ....
}'
$ curl -v -k -H 'Content-Type: application/yaml' -H 'X-ARC-Token:
YWRtaW4gICAgICAgIDE1NTY1NTgyMjExMzQ4N2NmMGUwNQ==' \
  --data "$YAML" https://arc.akraino.demo/api/v1/blueprint
```

Start the Deployment (Create the POD)

Start the deployment by creating a POD:

```
$ YAML='{
  name: my new POD,
  description: description of this POD,
  blueprint: 827cfe84-2e28-11e9-bb34-0017f20dbff8,
  edgesite: 2d3533e4-3dcb-11e9-9533-87ac04f6a7e6
}'
$ curl -v -k -H 'Content-Type: application/yaml' -H 'X-ARC-Token:
YWRtaW4gICAgICAgIDE1NTY1NTgyMjExMzQ4N2NmMGUwNQ==' \
  --data "$YAML" https://arc.akraino.demo/api/v1/pod
```

Make note of the UUID that is returned. You will need it to monitor the deployment.

Monitor the deployment by monitoring the *POD Event* URL for the newly created POD. This will return a list of events related to the POD similar to:

```
$ curl -v -k -H 'Accept: application/yaml' -H 'X-ARC-Token: YWRtaW4gICAgICAgIDE1NTY1NTgyMjExMzQ4N2NmMGUwNQ==' \
  https://arc.akraino.demo/api/v1/podevent/56b365a0-d6a2-4d12-8f02-e2fc2671573e
events:
- {level: INFO, time: '2019-04-29 18:15:28.0', message: Pod created.}
- {level: INFO, time: '2019-04-29 18:15:28.0', message: 'Starting workflow: create'}
- {level: INFO, time: '2019-04-29 18:15:28.0', message: 'Workflow directory created:
    $DROOT/workflow/create-56b365a0-d6a2-4d12-8f02-e2fc2671573e'}
- {level: WARN, time: '2019-04-29 18:17:38.0', message: 'Could not fetch the workflow
    file http://example.com/blueprints/create.py'}
```

Flow of Operation using CLI commands

The equivalent of the previous section, using the CLI command *rc_cli* would be:

Enumerate the Machines

Assuming the machines to deploy on have not yet been made known to the RC, you would need to use the Node API to add them to the RC database.

Do this with the following API call, once per node:

```
$ cat > node.yaml <<EOF
name: nodename
description: a description of the node
hardware: <hardware profile UUID>
EOF
$ rc_cli -u admin -p abc123 node create node.yaml
```

Keep track of the UUID of each node that is returned from the API.

Create an Edge Site

Once the nodes are defined, you need to create an Edge Site (a cluster of nodes). Do this:

```
$ cat > es.yaml <<EOF
name: edgesitename
description: a description of the Edgesite
nodes: [ <list of node UUIDs> ]
regions: [ <list of region UUIDs> ]
EOF
$ rc_cli -u admin -p abc123 edgesite create es.yaml
```

Create/Verify the Blueprint

To get the UUID of the Blueprint, you would need to see which Blueprints are installed:

```
$ rc_cli -u admin -p abc123 blueprint list
```

If the Blueprint you want is missing, you may need to create it:

```
$ cat > blueprint.yaml <<EOF
blueprint: 1.0.0
name: my new blueprint
version: 1.0.0
description: description of the blueprint
yaml: ....
EOF
$ rc_cli -u admin -p abc123 blueprint create blueprint.yaml
```

Start the Deployment (Create the POD)

Start the deployment by creating a POD:

```
$ cat > pod.yaml <<EOF
name: my new POD
description: description of this POD
blueprint: 827cfe84-2e28-11e9-bb34-0017f20dbff8
edgesite: 2d3533e4-3dcb-11e9-9533-87ac04f6a7e6
EOF
$ rc_cli -u admin -p abc123 pod create pod.yaml
```

Make note of the UUID that is returned. You will need it to monitor the deployment.

Monitor the deployment by displaying the newly created POD. Note: the *rc_cli* command does not presently interface to the POD event API, so this will not be precisely equivalent to the *curl* calls shown above. This will return a list of events related to the POD similar to:

```
$ ./rc_cli -H rc -p admin123 pod show d72e1901-b9b3-4137-b217-fc3cae4575ac
---
blueprint: 690450c0-776a-11e9-ae9b-f3cee2e49e42
description: CD of blueprint on OE
edgesite: 60ab1298-7769-11e9-92b3-373d9b2f2476
events:
- level: INFO
  message: Pod created.
  time: '2019-06-13 16:56:19.0'
- level: INFO
  message: 'Starting workflow: create'
  time: '2019-06-13 16:56:19.0'
- level: INFO
  message: 'Workflow directory created: $DROOT/workflow/create-0-d72e1901-b9b3-4137-b217-fc3cae4575ac'
  time: '2019-06-13 16:56:19.0'
- level: INFO
  message: 'Workflow fetched: http://www.example.org/blueprints/work-flow-v0.1.sh'
  time: '2019-06-13 16:56:19.0'
- level: INFO
  message: Workflow template created.
  time: '2019-06-13 16:56:19.0'
- level: INFO
  message: Starting create workflow for POD d72e1901-b9b3-4137-b217-fc3cae4575ac
  time: '2019-06-13 16:57:02.0'
name: pod_on_oe1
state: WORKFLOW
url: /api/v1/pod/d72e1901-b9b3-4137-b217-fc3cae4575ac
uuid: d72e1901-b9b3-4137-b217-fc3cae4575ac
workflows:
- create
yaml:
  input_yaml: http://www.example.org/mtlab/aknode201-user_config.yaml
  iso_primary: http://www.example.org/iso/latest/install.iso
  iso_secondary: http://www.example.org/iso/latest/bootcd.iso
```