Commissaire Documentation

Release 0.0.6

See CONTRIBUTORS

Jun 28, 2017

Contents

Contents:

CHAPTER 1

Overview

Commissaire is a lightweight REST interface for performing system management tasks on network hosts in a cluster through Ansible.

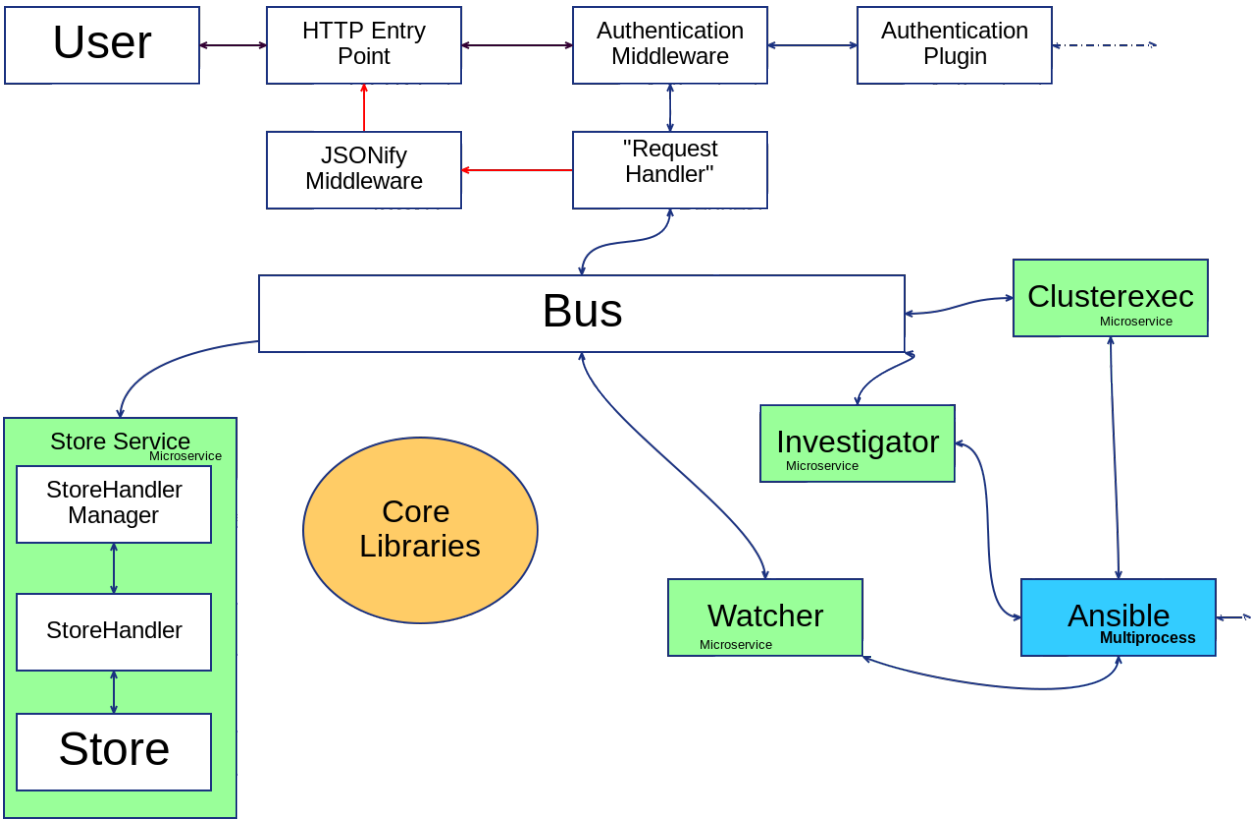
Current capabilities include rolling upgrades and restarts of traditional or "Atomic" hosts, and bootstrapping new hosts into an existing "container management" system such as OpenShift or Kubernetes.

Moving forward, Commissaire will expand the scope of its REST interface to provide centralized host inventory management and consoldate various Linux subsystems into a centralized API.

Feature Overview

- Restart hosts in an OpenShift or Kubernetes cluster
- Upgrade hosts in a OpenShift or Kubernetes cluster
- *Bootstrap new hosts into an existing OpenShift or Kubernetes cluster*
- No agent required for hosts: All communication is done over SSH
- *Simple REST interface for automation*
- Service status for health checking
- *Plug-in based authentication framework*
- *Command line interface for operators*
- Built in support for Atomic Host and Server variants of RHEL, Fedora, and CentOS

Logical Flow



What commissaire Is Not

There are a lot of overloaded words in technology. It's important to note what commissaire is not as much as what it is. commissaire is not:

- A Container Manager or scheduler (such as OpenShift or Kubernetes)
- A configuration management system (such as Ansible or Puppet)
- A replacement for individual host management systems

Example Uses

Note: This is an early list. More use cases will be added in the future.

- An administrator needs to upgrade an entire group of hosts acting as Kubernetes nodes
- An administrator needs to restart an entire group of hosts acting as Kubernetes nodes
- An organization would like new hosts to register themselves into a Kubernetes cluster upon first boot without administrator intervention

- An organization would like to keep groups of hosts used as Kubernetes nodes out of direct control of anything but Kubernetes and basic operations.

CHAPTER 2

Releases

Downloads

You can find the latest source releases via GitHub:

- commissaire
- commissaire-http
- commissaire-service
- commctl

Release Schedule

Commissaire follows semantic versioning. Releases occur on the following schedule:

- Minor or patch releases are released every 4th Monday of the month.
- Major releases occur when backwards incompatible changes are introduced.

Note: Until version `1.0.0` major changes may occur on the minor schedule.

CHAPTER 3

REST Configuration

Configuration File

```
$ cat /etc/commissaire/commissaire.conf
{
  "listen-interface": "127.0.0.1",
  "listen-port": 8000,
  "tls-pemfile": "/path/to/server.pem",
  "bus-uri": "redis://127.0.0.1:6379/",
  "authentication-plugins": [{
    "name": "commissaire_http.authentication.httpbasicauth",
    "filepath": "conf/users.json"
  }],
 "self-auths": ["/api/v0/secrets"]
}
```

Via CLI

```
usage: cli.py [-h] [--debug] [--config-file CONFIG_FILE] [--no-config-file]
              [--listen-interface LISTEN_INTERFACE]
              [--listen-port LISTEN_PORT] [--tls-pemfile TLS_PEMFILE]
              [--tls-clientverifyfile TLS_CLIENTVERIFYFILE]
              [--authentication-plugin MODULE_NAME:key=value,..]
              [--self-auth SELF_AUTHS] [--bus-exchange BUS_EXCHANGE]
              [--bus-uri BUS_URI]

optional arguments:
  -h, --help            show this help message and exit
  --debug               Turn on debug logging to stdout
  --config-file CONFIG_FILE, -c CONFIG_FILE
                        Full path to a JSON configuration file (command-line
                        arguments override)
```

```
  --no-config-file      Disregard default configuration file, if it exists
  --listen-interface LISTEN_INTERFACE, -i LISTEN_INTERFACE
                        Interface to listen on
  --listen-port LISTEN_PORT, -p LISTEN_PORT
                        Port to listen on
  --tls-pemfile TLS_PEMFILE
                        Full path to the TLS PEM for the commissaire server
  --tls-clientverifyfile TLS_CLIENTVERIFYFILE
                        Full path to the TLS file containing the certificate
                        authorities that client certificates should be
                        verified against
  --authentication-plugin MODULE_NAME:key=value,..
                        Authentication Plugin module and configuration.
  --self-auth SELF_AUTHS
                        URI paths which provide their own authentication.
  --bus-exchange BUS_EXCHANGE
                        Message bus exchange name.
  --bus-uri BUS_URI     Message bus connection URI. See:http://kombu.readthedo
                        cs.io/en/latest/userguide/connections.html

Example: commissaire -c conf/myconfig.json
```

Example

The following will run the same server as the above configuration file examples.

Note: `--no-config` is required when bypassing the configuration file!

```
$ commissaire-server --no-config \
    --bus-uri redis://127.0.0.1:6379/ \
    --bus-exchange commissaire \
    --tls-pemfile /path/to/server.pem \
    --listen-interface 8000 \
    --authentication-plugin commissaire_http.authentication.
↪httpbasicauth:filepath=conf/users.json \
    --self-auth /api/v0/secrets
```

Authentication

Multiple authentication plugins can be configured via the CLI. To do this use the `--authentication-plugin` switch multiple times.

```
...
--authentication-plugin commissaire_http.authentication.httbasicauth:filepath=conf/
↪users.json \
--authentication-plugin commissaire_http.authentication.keystonetokenauth:url=https://
↪example.com
```

CHAPTER 4

Walkthrough

This document walkthroughs a simple scenario with Commissaire.

Before We Start

Some commands sections talk about an ssh key. The clarify, the ssh key always meets these requirements:

- The key is a private ssh key
- A copy of the private key would be on the operators system
- The key would belong to a user on the remote host (IE: it would be listed in the authorized_keys file on the remote host)
- The user on the remote system would be privileged (easiest example: root)
- The key is used within Commissaire to access hosts

Configuring a ContainerManager (Optional)

If you will be using OpenShift, OCP, or Kubernetes then configuring a ContainerManager is the first thing to do. This essentially will tell commissaire how to communicate with your ContainerManager. When a cluster is associated to this ContainerManager new hosts will be automatically added into the the ContainerManager as nodes.

Let's say you wanted to add a ContainerManager called `ocp`, which has a url of `https://openshift.example.com`, and uses a token of `aaa` for authentication:

```
commctl container_manager create --type openshift --options='{"server_url": "https://
↪openshift.example.com", "token": "aaa"}' ocp
...
```

Note: *Adding Hosts To The Cluster*, later in this document, will show how the ContainerManager interacts binds with Clusters and Hosts.

Creating a Cluster

Clusters are groupings of hosts. These hosts are expected to be similar to each other in functionality. In other words, the configurations of hosts in a cluster should not differ. While the functionality provided by the hosts may differ the system itself should not. Take OpenShift nodes as an example. Some nodes may be hosting pods running different workloads, such as database services, web applications, or a mixture. However, the underlying hosts themselves are configured to be OpenShift nodes and are configured identical to each other.

To create a brand new cluster:

Note: If you did not create a ContainerManager you can omit `--container-manager`.

```
commctl cluster create --container-manager ocp mycluster
...
```

Adding Hosts To The Cluster

Adding new hosts to Commissaire comes in two forms. Automatic registration and manual additions.

Automatic Registration

First, you must create the `user-data` file. `commctl` provides a command, named `user-data`, which helps generate this file for you. Here is an example:

```
$ commctl user-data \
  --password \
  --username USER \
  --cluster CLUSTER \
  --endpoint https://example.com/ \
  CLUSTER.userdata
Password: <PASS>
$ # Let's check that the userdata file is indeed a multipart/mixed file
$ file CLUSTER.userdata
CLUSTER.userdata: multipart/mixed; boundary="==============8094544984785845936==",␣
↪ASCII text
```

Now provide the new `user-data` file when provisioning new hosts in your cloud provider. When the new host starts it will automatically register into Commissaire.

Manual Registration

You can also add hosts into Commissaire in a manual fashion. To do this you will need:

1. The host to have `sshd` running

2. The host to have `sshd` port open.
3. The private key to an administrative user on the host (EG: `root`)

Note: Jump to *Creating Keys* if you want to create a new key

Let's say you have a host called `192.168.152.110` which you'd like to add to the cluster `my_cluster`. You also have a private key of the remote root user for `192.168.152.110` at `/path/to/remote/hosts/priv/ssh_key`. The following command would add the host to the cluster:

Note: Remember, the ssh key references the operators copy of the key used when accessing the new host

```
$ commctl host create --cluster my_cluster 192.168.152.110 /path/to/remote/hosts/priv/
↪ssh_key
...
```

Operations

Now that you have at least one host registered in a cluster you can now do operations. Let's do a restart. The following command will start the restart process.

```
commctl cluster restart start my_cluster
...
```

Now let's see what the status of the process is:

For more operations via `commctl` see *commctl*

Optional Steps

The following are optional items which may prove useful for some users.

Creating Keys

If you want to create a new key pair for the remote host you can do the following:

```
This creates a new ssh public and private key as ``new_key.pub`` and ``new_key``.

.. code-block:: shell

   $ ssh-keygen -f new_key -C root
   Generating public/private rsa key pair.
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in new_key.
   Your public key has been saved in new_key.pub.
   The key fingerprint is:
   SHA256:YoFOXojY0tIkAQBRiPe00HWQdJ8zgOylJwDuQXXJfXc steve@bitfall
   The key's randomart image is:
   +---[RSA 2048]----+
```

```
   |O%oo=.=*Bo.       |
   |*.*+ Bo+*= . E    |
   |oo .o Eo=.. .     |
   |+.  + o .         |
   |...   o+S         |
   |oo oo  .    ..    |
   |+ . +             |
   | ..               |
   |                  |
   +----[SHA256]-----+
```

You could then use your cloud provider to inject the key into the host.

- AWS documentation
- GCE

CHAPTER 5

Operations

Preface

All operations via Commissaire are done via `REST`. `REST` calls are the prefered way to integrate with Commissaire. While any HTTP client can be used to directly interface with the `REST` server, many operators will feel more comfortable using *commctl*.

curl

Every call requires a username and password to be passed via HTTP Basic Auth. With curl this looks like:

```
curl ... -u "USERNAME:PASSWORD" ...
```

The proper headers must also be passed. Since all of the `REST` communication is done via JSON the `Content-Type` must be set to `application/json`.

```
curl ... -H "Content-Type: application/json" ...
```

Lastly, the type of operation must be specified. For example, *PUT* must be used when creating while *GET* must be used for retrieving.

```
curl ... -XPUT ...
```

Bootstrapping

Bootstrapping happens when a new host is added to commissaire via the Host endpoint.

```
curl -u "a:a" -XPUT -H "Content-Type: application/json" http://localhost:8000/api/v0/
↪host/192.168.1.100 -d '{"host": "192.168.1.100", "cluster": "datacenter1", "remote_
↪user": "root", "ssh_priv_key": "dGVzdAo="}'
...
```

It's important to remember *ssh_priv_key* must be base64 encoded without newlines. On many systems this can be done via that **base64** command and using the **-w0** switch.

```
$ cat path/to/key | base64 -w0 > encoded_key
```

For specifics on the endpoint see *Host*

Note: commissaire can help automate the bootstrapping of new hosts using cloud-init for early initialization. See *Cloud-Init Integration*.

Cluster Operations with curl

Note: Operators will probably want to use *commctl*

These operations are done across all hosts associated with a cluster.

Restart

Restarting a cluster is done by creating a new `restart` record for a specific cluster.

```
curl -u "a:a" -XPUT -H "Content-Type: application/json" http://localhost:8000/api/v0/
↪cluster/datacenter1/restart
...
```

To check on a `restart` record, a `REST` *GET* call on the same endpoint will show the current status.

```
curl -u "a:a" -XGET -H "Content-Type: application/json" http://localhost:8000/api/v0/
↪cluster/datacenter1/restart
...
```

For specifics on the `restart` endpoint see *Cluster Operations: Restart*

Upgrade

Upgrading a cluster is done by creating a new `upgrade` record for a specific cluster.

```
curl -u "a:a" -XPUT -H "Content-Type: application/json" http://localhost:8000/api/v0/
↪cluster/datacenter1/upgrade
...
```

To check on an `upgrade` record, a `REST` *GET* call on the same endpoint will show the current status.

```
curl -u "a:a" -XGET -H "Content-Type: application/json" http://localhost:8000/api/v0/
↪cluster/datacenter1/upgrade
...
```

For specifics on the `upgrade` endpoint see *Cluster Operations: Upgrade*

Deploy

Deploying to a cluster is done by creating a new `deploy` record for a specific cluster.

```
curl -u "a:a" -XPUT -H "Content-Type: application/json" --data='{"version": "7.2.1"}'␣
↪http://localhost:8000/api/v0/cluster/datacenter1/deploy
...
```

To check on a `deploy` record, a `REST` *GET* call on the same endpoint will show the current status.

```
curl -u "a:a" -XGET -H "Content-Type: application/json" http://localhost:8000/api/v0/
↪cluster/datacenter1/deploy
...
```

For specifics on the `deploy` endpoint see *Cluster Operations: Deploy*

CHAPTER 6

commctl

Preface

`commctl` is the official command line utility for Commissaire. `commctl` acts as a clean user interface between the operator and the commissaire-server allowing for a more traditional experience for operators.

Installation

Via Source

```
$ pip install git+https://github.com/projectatomic/commctl.git
...
```

Via Docker

From a checkout of the `commctl` repository:

```
$ sudo docker build -t commctl .  # Done once to build the image
...
$ sudo docker run -t -i --volume=`pwd`/.commissaire.json:/root/.commissaire.json:Z␣
↪commctl --help
```

Via RPM

If you want to roll your own RPM, the spec file can be found in the Fedora package repo.

On RHEL/CentOS/Fedora based systems you will also need to make sure to have an RPM build environment set up. This includes packages such as:

- rpm-build
- redhat-rpm-config

For further dependencies please see `BuildRequires` in the spec file.

Configuration

`commctl` requires a configuration file. The default path is `~/.commissaire.json` though it can be changed with the `--config`/`-c` option.

```
{
    "username": "a",
    "endpoint": "http://127.0.0.1:8000"
}
```

Note: At least one endpoint must be defined!

The password may be stored in the configuration file as well.

Warning: The configuration file is plain text. If you choose to keep a password in the file make sure to keep the file permissions locked down.

```
{
    "username": "a",
    "password": "a",
    "endpoint": "http://127.0.0.1:8000"
}
```

If you are using the Kubernetes authentication plugin you can opt to reuse the credentials from your kubeconfig like so:

Note: If you include username/password and kubeconfig items the username/password will be ignored in favor of the kubeconfig.

Multiple fallback endpoints may be specified as a list. The endpoints are tried in order until a successful connection is made.

```
{
    "username": "a",
    "endpoint": [
        "http://127.0.0.1:8000",
        "http://192.168.122.100:8000",
        "http://10.1.1.1:8000"]
}
```

Commands

cluster

Note: For API versions of these commands see the *Cluster API*

create

`create` will create a new cluster. It takes in two flags:

- `-t`/`--type`: Type of the cluster (Default: kubernetes)
- `-n`/`--network`: Name of the network (Default: default)

`create` requires one positional argument:

- `name`: The name to give the cluster

```
$ commctl cluster create --type kubernetes --network default my_cluster
```

delete

`delete` will delete a cluster from the server.

`delete` requires one positional argument:

- `name`: The name of the cluster to delete

```
$ commctl cluster delete my_cluster
```

get

`get` will retrieve a cluster from the server.

`get` requires one positional argument:

- `name`: The name of the cluster to retrieve

```
$ commctl cluster get my_cluster
```

list

`list` will provide a list of all configured clusters.

To list all clusters:

```
commctl cluster list
...
```

deploy start

`deploy start` will create a new deployment on an Atomic Host. This is an asynchronous action. See *deploy status* on checking the results.

`deploy start` requires two positional arguments:

- `name`: The name of the cluster to deploy upon
- `version`: The version with which to upgrade

```
$ commctl cluster deploy start mycluster 7.4.1
```

deploy status

`deploy status` will retrieve the status of an deploy

`deploy status` requires one positional argument:

- `name`: The name of the cluster to check

```
$ commctl cluster deploy status mycluster
```

restart start

`restart start` will create a new restart roll on a cluster of hosts. This is an asynchronous action. See *restart status* on checking the results.

`restart start` requires one positional argument:

- `name`: The name of the cluster to restart

```
commctl cluster restart start my_cluster
...
```

restart status

`restart status` will retrieve the status of an restart

`restart status` requires one positional argument:

- `name`: The name of the cluster to check

```
commctl cluster restart status my_cluster
...
```

upgrade start

`upgrade start` will create a new upgrade on a cluster of hosts. This is an asynchronous action. See *upgrade status* on checking the results.

`upgrade start` requires one positional argument:

- `name`: The name of the cluster to upgrade

```
commctl cluster upgrade start datacenter1 7.2.2
...
```

upgrade status

`upgrade status` will retrieve the status of an upgrade

`upgrade status` requires one positional argument:

- `name`: The name of the cluster to check

```
commctl cluster upgrade status datacenter1
...
```

host

Note: For API versions of these commands see the *Host API*

create

`create` will create a new host record. It takes in one flag:

- `-c`/`--cluster`: Adds the host to the specified cluster

`create` requires two positional arguments:

- `address`: The domain or address of the host to access and add
- `ssh_priv_key`: The full path to the **remote hosts** ssh private key for initial access

```
.. code-block:: shell

   $ commctl host create --cluster my_cluster 192.168.152.110 /path/to/remote/hosts/
↪priv/ssh_key
   ...
```

Note: When creating a new host record the remote host will need to have an ssh key already generated and available for commissaire. The host also will need to have ssh running and the `python` command must be available. If you want to bootstrap new hosts please see our *Cloud-Init Integration* documentation.

delete

`delete` will delete a host from the server.

`delete` requires one positional argument:

- `name`: The name of the host to delete

```
$ commctl host delete 192.168.152.110
```

get

`get` retrieves a host record from the server.

`get` requires one positional argument:

- `address`: The address or domain of the host to retrieve

```
$ commctl host get 192.168.152.110
```

list

`list` will provide a list of all configured hosts.

To list all hosts:

```
commctl host list
...
```

status

`status` retrieves status information for a specific host.

`status` requires one positional argument:

- `address`: The address or domain of the host to retrieve status

```
$ commctl host status 192.168.152.110
```

ssh

Note: For the api used for this commands see the *Host Creds API*

`commctl` provides a simple way to connect to your host node by pulling down the `ssh_priv_key` and `remote_user` from the server. The `ssh_priv_key` is stored temporarily and is removed upon the completion of the connection.

`ssh` requires one positional argument:

- `hostname`: The address or domain of th

`ssh` allows for N optional positional argument:

- `extra_args`: Extra arguments to pass to the ssh command

To connect to a host node:

```
commctl host ssh 192.168.1.100
...
```

To connect to a host node with extra ssh parameters:

```
commctl host ssh 192.168.1.100 -v -p 9876
...
```

network

Note: For API versions of these commands see the *Network API*

create

`create` will create a new network record. It takes in two flags:

- `-t`/`--type`: The type of the network: (Default: flannel_etcd)
- `-o`/`--options`: Additional options for the network (Default: "{}")

`create` requires one positional argument:

- `name`: The name to give the network

```
$ commctl network create --type flannel_server --options '{"address": "192.168.152.
↪100:8080"}' my_network
```

delete

`delete` will delete a network from the server.

`delete` requires one positional argument:

- `name`: The name of the network to delete

```
$ commctl network delete my_network
```

get

`get` will retrieve a network from the server.

`get` requires one positional argument:

- `name`: The name of the network to retrieve

```
$ commctl network get my_network
```

list

`list` will provide a list of all configured networks.

To list all hosts in a specific cluster:

```
commctl host list datacenter1
...
```

passhash

The `passhash` command provides an easy way to create bcrypt2 hashes.

The quickest way to use the command is to provide no flags. This will prompt you for the password and output the hash.

```
$ commctl passhash
Password:
$2a$12$tMz3FVwwwkXoXcTvCHdNnul1wC.sBX1KyRYEB.FZ42VCPZVc5.SyW
```

If you have a password in a file you can use the `--file`/`-f` switch to use it as the password.

```
$ commctl passhash --file my_password.txt
$2a$12$K5KtQ6woCJW5Y9gSC9W25eRu1rMWIT5WyLsLtauoZyB2bZQ8yjc1C
```

If you would like to change the strength of the hash via it's rounds you can use `--rounds`/`-r`.

```
$ commctl passhash --rounds 15
Password:
$2a$15$mTKz3Hl08AcJsK79YGk9G.RHe1P9ksr/whLyxZGsh92bvJt83mb8q
```

If you want to pass the password directly in the command you can use `--password`

> **Warning:** Generally this is a bad idea as the password may be kept in shell history and will be viewable by anyone else with access to the terminal.

```
$ commctl passhash --password bad_idea
$2a$12$BJZYMKFEvG1osE5YXBxwIOMEHCpvHu8IlSnVpE6L0JbuhNCa.Lj.C
```

CHAPTER 7

Components

Internal

The following are internal components of commissaire.

Commissaire Server

The commissaire server is the `REST` interface and is how an administrator works with commissaire. It attempts to follow `REST` as strictly as possible through the interpretation of commissaire developers.

Services

See *Commissaire Services*

External

The following are external components of commissaire.

etcd

etcd is used as the data store for commissaire. Any persistent data is kept within etcd as either traditional *key = value* pairs or as *key = JSON*. While any etcd instance will work it's recommended to use the same etcd cluster with Kubernetes.

Container Manager

OpenShift or Kubernetes can be used as the container manager. commissaire utilizes Kubernetes API to ensure that new host nodes register properly. From this point forward Kubernetes is able to use the host node to schedule pods, etc...

CHAPTER 8

Commissaire Services

Commissaire Service is a framework for writing long running services for the Commissaire management system. It provides a standard way to connect to Commissaire's message bus and provide/consume services.

Each service by default looks for a `.conf` file named after itself in `/etc/commissaire` for its configuration. For example, the Storage service looks for `/etc/commissaire/storage.conf`. The default location can be overridden with the `-c/--config` command-line option for any of the services.

For easier deployment on cloud services, each Commissaire service will also look to etcd for configuration if certain environment variables are defined, particularly `ETCD_MACHINES`.

Recognized environment variables for retrieving configuration from `etcd` are:

- `ETCD_MACHINES` : Comma-separated list of `etcd` service URLs
- `ETCD_TLSPEM` : Optional path to local TLS client certificate public key file
- `ETCD_TLSKEY` : Optional path to local TLS client certificate private key file
- `ETCD_CACERT` : Optional path to local TLS certificate authority public key file
- `ETCD_USERNAME` : Optional username used for basic auth
- `ETCD_PASSWORD` : Optional password used for basic auth

The configuration format, whether retrieved from a local file or from etcd, is a JSON object with some common and some service-specific members. JSON members that are recognized by all services include:

- `bus_uri` : Message bus connection URI, handed off to a Kombu Connection. The service's `--bus-uri` command-line option overrides this.
- `logging` : Logging configuration. This section is parsed into a Python dictionary and handed off to Python's logging.config.dictConfig function. Commissaire installs a default formatter and handler in the "root" logger which logs to stderr, so this section can usually be omitted.
- `debug` : A boolean option to enable debug-level logging messages in the "root" logger. The effect should be application-wide unless overridden in the `logging` section. Defaults to `False`.

Built-In Services

Commissaire Clusterexec

Commissaire's `Cluster Execution` service is a set of long running processes which handle rolling operations over hosts in a cluster.

- Local configuration in file `/etc/commissaire/clusterexec.conf`
- Remote configuration in `etcd` key `/commissaire/config/clusterexec`

Commissaire Container Manager

Commissaire's `Container Manager` service is a set of long running processes which provide a consistant API to work with container managers.

- Local configuration in file `/etc/commissaire/containermgr.conf`
- Remote configuration in `etcd` key `/commissaire/config/containermgr`

Commissaire Investigator

Commissaire's `Investigator` is a set of long running processes which connect to and bootstrap new hosts wanting to be managed by Commissaire.

- Local configuration in file `/etc/commissaire/investigator.conf`
- Remote configuration in `etcd` key `/commissaire/config/investigator`

Commissaire Watcher

Commissaire's `Watcher` is a set of long running processes which periodically connects to hosts that have already been bootstrapped and checks their status.

- Local configuration in file `/etc/commissaire/watcher.conf`
- Remote configuration in `etcd` key `/commissaire/config/watcher`

Commissaire Storage

Commissaire's `Storage` is a set of long running processes which broker storage and retrieval requests of persistent data.

Additionally, this service publishes notifications on the bus when creating, updating or deleting stored records. Other services can listen for and react to these notifications to automatically update internal state or kick off a long-running operation.

- Local configuration in file `/etc/commissaire/storage.conf`
- Remote configuration in `etcd` key `/commissaire/config/storage`

Configuration

The `Storage` service looks for a few additional JSON members in its configuration:

- `custodia_socket_path` : File path to the Unix domain socket opened by the Custodia service. The default path matches Custodia's default socket path (*/var/run/custodia/custodia.sock*), so this member need only be set if the Custodia service is using a different socket path.
- `storage_handlers` : This is a JSON object, or a list of JSON objects, describing which storage plugins to use, a list of model types to claim, and any other plugin-specific settings. Each object recognizes these members:
  - `type` : The module name of the storage plugin (for convenience, no dots in the name implies a prefix of `commissaire.storage`). This value is **required**, although it will usually be `etcd`, as no other plugins come built-in at this time.
  - `name` : Optional name to uniquely identify the plugin instance. If omitted, a unique name is derived from the plugin's module name, such as `commissaire.storage.etcd`.
  - `models` : A list of model type names such as `"Host"` or `"Cluster"`, or glob-style patterns like `"Host*"` or `"*"`. Storage requests for these types of models will be routed to the plugin instance specified by the `type` member. Defaults to `["*"]`, which is usually sufficient.

Writing a Service

See *Developing Services*

CHAPTER 9

Enums

OS's

- **atomic**: http://www.projectatomic.io/
- **rhel**: http://www.redhat.com/en/technologies/linux-platforms/enterprise-linux
- **centos**: https://www.centos.org/
- **fedora**: https://getfedora.org/
- **flannel_etcd**: Uses the configured etcd store handler for it's network configuration
- **flanneld_service**: Uses flannel in client/server mode. Requires options to have `address` of `host:port`.

Statuses

Cluster Statuses

- **ok**: The cluster is active as expected.
- **degraded**: The cluster has one more more nodes that are not active.
- **failed**: No nodes are currently active.

Host Statuses

- **investigating**: The host has passed credentials to commissaire which is now looking at the system.
- **bootstrapping**: The host is going through changes to become active.
- **active**: The host is part of the cluster and is registered with the Container Manager.
- **disassociated**: The host exists but is not associated with the Container Manager.

- **failed**: Unable to access the system.

Upgrade Statuses

- **in_process**: The cluster is currently upgrading hosts.
- **finished**: The cluster successfully upgraded.
- **failed**: The cluster could not upgrade.

ContainerManager Types

- **openshift**: A cluster with an OpenShift compatible API.

CHAPTER 10

REST Endpoints

REST stands for representational state transfer and is one of many ways to expose API's as a web service. REST allows "requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. [...] Using HTTP, as is most common, the kind of operations available include those predefined by the HTTP verbs GET, POST, PUT, DELETE and so on." (Wikipedia)

For more information on REST see the original dissertation.

Cluster

Endpoint: /api/v0/cluster/{NAME}

(Internal model name: Cluster)

Changed in version 0.1.0: type has been removed in favor of container_manager.

GET

Retrieve the status of the cluster.

```
{
    "name": string,
    "status" string,
    "network": string,
    "container_manager": str,
    "hosts": {
        "total": int,
        "available": int,
        "unavailable": int
    }
}
```

Example

```
{
    "name": "mycluster",
    "status": "ok",
    "network": "default",
    "container_manager": "prod_openshift",
    "hosts": {
        "total": 3,
        "available": 2,
        "unavailable": 1
    }
}
```

PUT

Creates a new cluster.

Deprecated since version 0.0.1: Provide a `network` when creating a new Cluster.

```
{
    "container_manager": string // (Optional) Name of the container manager to use
    "network": string     // The name of the network
}
```

Example

```
{
    "container_manager": "prod_openshift",
    "network": "default"
}
```

DELETE

Deletes an existing cluster.

Example Response

```
[]
```

Cluster Members

Endpoint: /api/v0/cluster/{NAME}/hosts

GET

Retrieve the host list for a cluster.

```
[
    host_address,...
]
```

Example

```
[
    "192.168.100.50",
    "192.168.100.51"
]
```

PUT

Replace the host list for a cluster. The "old" list must match the current host list.

```
{
    "old": [host_address,...]
    "new": [host_address,...]
}
```

Example

```
{
    "old": ["192.168.100.50"],
    "new": ["192.168.100.50", "192.168.100.51"]
}
```

Cluster Members (Individual)

Endpoint: /api/v0/cluster/{NAME}/hosts/{IP}

GET

Membership test. Returns 200 if host {IP} is in cluster, else 404.

Example Response

```
['192.168.100.50']
```

PUT

Adds host {IP} to cluster and returns the host added in a list. (Idempotent)

No body.

Example Response

```
['192.168.100.50']
```

DELETE

Removes host {IP} from cluster returning an empty list. (Idempotent)

No body.

Example Response

```
[]
```

Cluster Operations: Deploy

Endpoint: /api/v0/cluster/{NAME}/deploy

(Internal model name: `ClusterDeploy`)

GET

Retrieve the current status of an OSTree tree deployment.

```
{
    "status": string,
    "version": string,
    "deployed": HOST_LIST,
    "in_process": HOST_LIST,
    "started_at": string,
    "finished_at": string
}
```

Example

```
{
    "status": "in_process",
    "version": "7.2.6",
    "deployed": [{...}],
    "in_process": [{...}],
    "started_at": "2015-12-17T15:48:18.710454",
    "finished_at": null
}
```

PUT

Start a new OSTree tree deployment.

```
{
    "version": string  // Which OSTree tree to deploy
}
```

Example

```
{
    "version": "7.2.6"
}
```

Example Response

```
{
    "status": "in_process",
    "version": "7.2.6",
    "deployed": [{...}],
    "in_process": [{...}],
    "started_at": "2015-12-17T15:48:18.710454",
    "finished_at": null
}
```

Cluster Operations: Upgrade

Endpoint: /api/v0/cluster/{NAME}/upgrade

(Internal model name: `ClusterUpgrade`)

GET

Retrieve the current status of upgrades.

```
{
    "status": string,
    "upgraded": HOST_LIST,
    "in_process": HOST_LIST,
    "started_at": string,
    "finished_at": string
}
```

Example

```
{
    "status": "in_process",
    "upgraded": [{...}],
    "in_process": [{...}],
    "started_at": "2015-12-17T15:48:18.710454",
    "finished_at": null
}
```

PUT

Start a new upgrade.

No body.

Example Response

```
{
    "status": "in_process",
    "upgraded": [{...}],
    "in_process": [{...}],
    "started_at": "2015-12-17T15:48:18.710454",
    "finished_at": null
}
```

Cluster Operations: Restart

Endpoint: /api/v0/cluster/{NAME}/restart

(Internal model name: `ClusterRestart`)

GET

Retrieve the status of a restart.

```
{
    "status": string,
    "restarted": HOST_LIST,
    "in_process": HOST_LIST,
    "started_at": string,
    "finished_at": string
}
```

Example

```
{
    "status": "in_process",
    "restarted": [{...}],
    "in_process": [{...}],
    "started_at": "2015-12-17T15:48:18.710454",
    "finished_at": null
}
```

PUT

Create a new restart.

No body.

Example Response

```
{
    "status": "in_process",
    "restarted": [{...}],
    "in_process": [{...}],
    "started_at": "2015-12-17T15:48:18.710454",
    "finished_at": null
}
```

Clusters

Endpoint: /api/v0/cluster/

(Internal model name: `Clusters`)

GET

Retrieve a list of all clusters.

```
[
    string,...
]
```

Example

```
[
   "mycluster",
]
```

ContainerManagers

Endpoint: /api/v0/containermanagers/

(Internal model name: `ContainerManagerConfig`)

GET

Retrieve a list of all configured ContainerManagers.

```
[
    string,...
]
```

Example

```
[
   "prod_openshift",
]
```

ContainerManagerConfig

Endpoint: /api/v0/containermanager/{name}

(Internal model name: `ContainerManagerConfig`)

GET

Retrieve a specific ContainerManagerConfig record.

```
{
    "name": string,         // The name of the ContainerManagerConfig
    "type":  enum(string), // The type of the ContainerManagerConfig
    "options": dict         // Options to configure a ContainerManagerConfig
}
```

Note: See *ContainerManager Types* for a list and description of ContainerManager types.

Example

```
{
    "name": "prod_openshift",
    "type": "openshift",
    "options": {
        "apiserver": "https://192.168.152.101:8080/api/"
    },
}
```

PUT

Creates a new ContainerManagerConfig record.

```
{
    "name": str,            // Name of the ContainerManagerConfig
    "type":  enum(string), // The type of the ContainerManagerConfig
    "options": dict         // Options to explain a ContainerManagerConfig
}
```

Note: See *ContainerManager Types* for a list and description of ContainerManager types.

Example

```
{
    "type": "openshift",
    "options": {
        "apiserver": "https://192.168.152.101:8080/api/"
    },
}
```

DELETE

Deletes a ContainerManagerConfig record. (Idempotent)

No body.

Example Response

```
[]
```

Host

Endpoint: /api/v0/host/{IP}

(Internal model name: `Host`)

GET

Retrieve a specific host record.

```
{
    "address": string,        // The IP address of the cluster host
    "status":  enum(string), // The status of the cluster host
    "os": enum(string),       // The OS name
    "cpus": int,              // The number of CPUs on the cluster host
    "memory": int,            // The memory of the cluster host in kilobytes
    "space": int,             // The diskspace on the cluster host
    "last_check": string,     // ISO date format the cluster host was last checked
    "source": string          // (optional) External source for host information
}
```

Note: See *Host Statuses* for a list and description of host statuses.

Note: See *OS's* for a list and description of host statuses.

The `source` value, if defined, names a storage plugin which can provide information for this particular `Host` record. If omitted, host information is obtained from the default storage plugin defined by Commissaire's storage configuration.

Example

A host owned by Commissaire.

```
{
    "address": "192.168.100.50",
    "status": "active",
    "os": "atomic",
    "cpus": 4,
    "memory": 11989228,
    "space": 487652,
    "last_check": "2015-12-17T15:48:18.710454",
    "source": ""
}
```

A host owned by an external provider (note the "source" field).

```
{
    "address": "192.168.100.50",
    "status": "active",
    "os": "fedora",
    "cpus": 4,
    "memory": 2048,
    "space": 51475068,
    "last_check": "2016-11-28T22:10:11.851787",
    "source": "cloudforms"
}
```

PUT

Creates a new host record.

```
{
    "ssh_priv_key": string, // base64 encoded ssh private key
    "remote_user": string,  // Optional name of ssh user to use (default=root)
    "cluster": string       // Optional cluster the host should be associated with
}
```

Note: The rest of the host record will be filled out once the data has been pulled from the cluster host.

Note: As a convenience to hosts wishing to add themselves as part of a boot script, the endpoint /api/v0/host (without the {IP}) also accepts PUT requests. Here, the host address is inferred from the request itself but otherwise works the same: creates a new host record accessible at /api/v0/host/{IP}.

Example

```
{
    "cluster": "default",
    "remote_user": "root",
    "ssh_priv_key": "dGVzdAo..."
}
```

DELETE

Deletes a host record.

HostCreds

Endpoint: /api/v0/host/{IP}/creds

GET

Retrieve a specific hosts credentials.

```
{
    "ssh_priv_key": string, // base64 encoded ssh private key
    "remote_user":  string, // name of ssh user to use for connections
}
```

HostStatus

Endpoint: /api/v0/host/{IP}/status

(Internal model name: `HostStatus`)

GET

Retrieve a specific hosts status.

```
{
    "type":              string, // type of status
    "host":              dict,   // status elements from the Host instance
    "container_manager": dict,   // status elements reported from the Container␣
↪Manager
}
```

Example: Default

```
{
    "type": "host_only",
    "host": {
        "last_check": "2016-07-29T19:54:57.204671",
        "status": "active",
    },
    "container_manager": {...}
}
```

Hosts

Endpoint: /api/v0/hosts

(Internal model name: `Hosts`)

GET

Retrieve a list of hosts.

```
[
    {
        "address": string,       // The IP address of the cluster host
        "status":  enum(string), // The status of the cluster host
        "os": enum(string),      // The OS name
        "cpus": int,             // The number of CPUs on the cluster host
        "memory": int,           // The memory of the cluster host in kilobytes
        "space": int,            // The diskspace on the cluster host
        "last_check": string     // ISO date format the cluster host was last checked
    }...
]
```

Note: See *Host Statuses* for a list and description of host statuses.

Note: See *OS's* for a list and description of host statuses.

Example

```
[
    {
        "address": "192.168.100.50",
        "status": "active",
        "os": "atomic",
        "cpus": 4,
        "memory": 11989228,
        "space": 487652,
        "last_check": "2015-12-17T15:48:18.710454"
    },
    {
        "address": "192.168.100.51",
        "status": "active",
        "os": "atomic",
        "cpus": 3,
        "memory": 11989228,
        "space": 487652,
        "last_check": "2015-12-17T15:48:30.401090"
    }
]
```

Networks

Endpoint: /api/v0/networks/

(Internal model name: `Networks`)

GET

Retrieve a list of all networks.

```
[
    string,...
]
```

Example

```
[
   "mynetwork",
]
```

Network

Endpoint: /api/v0/network/{name}

(Internal model name: `Network`)

GET

Retrieve a specific network record.

```
{
    "name": string,         // The name of the network
    "type":  enum(string), // The type of the network
    "options": dict         // Options to explain a network
}
```

Note: See network-types for a list and description of network types.

Example

```
{
    "name": "mynetwork",
    "type": "flannel_server",
    "options": {
        "address": "192.168.152.101:8080"
    },
}
```

PUT

Creates a new network record.

```
{
    "type":  enum(string), // The type of the network
    "options": dict        // Options to explain a network
}
```

Note: See network-types for a list and description of network types.

Example

```
{
    "type": "flannel_server",
    "options": {
        "address": "192.168.152.101:8080"
    },
}
```

DELETE

Deletes a network record. (Idempotent)

No body.

Example Response

```
[]
```

CHAPTER 11

Community Meetings

The Commissaire Community Meeting is intended to bring all those who are interested in, currently working on, or using Commissaire together to discuss the project as a group.

The meeting starts at 3:00 PM UTC and is held the second and fourth Tuesday of each month in the `#atomic` Freenode IRC channel.

Timezone	Time
UTC	3:00 PM
EDT (US)	11:00 AM
IST (India)	8:30 PM
CST (China)	11:00 PM
AEST	1:00 AM

Rules

- Anyone can propose ideas
- There are no bad ideas
- Non-technical items are no more or less important than technical items
- It is healthy to disagree but it must be kept civil
- If you need help on a subject ask/If someone asks for help, help them!

Meeting Procedure

Each meeting is run by one of the core developers and the results channel is logged and posted to a gist for archival.

The meeting is broken into four sections: Presentation(s), Open PR Discussion, Open Issue Discussion, and Open Floor.

Presentation(s)

The Presentation section is used when a member would like to present slides, a demo, etc.. The slot for presentations will be available at every meeting but only meetings with proposed presentations will utilize the time.

Open PR Discussion

The Open PR section is used:

- List out the current open pull requests across all of the Commissaire repos
- As a group discuss PRs. This may be for clarification, requests for help, review, etc..
- Close any PRs which are finished/no longer needed

Open Issue Discussion

The Open Issue section is used for:

- List out the current open issues across all of the Commissaire repos
- As a group discuss issues. This may be for clarification, requests for help, review, etc..
- Close/Open issues based on discussion

Open Floor

The Open Floor section is used for general Commissaire discussion. Examples include:

- Proposing a presentation for next meeting
- Noting cancelation of a future meeting
- Discussion about meeting process
- Architecture/Design discussions

CHAPTER 12

Licenses

Server License: GPLv3+

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
```

```
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
```

```
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
```

```
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
```

```
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
```

```
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
```

```
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
```

```
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
```

```
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
```

```
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.
```

```
  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.
```

```
  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
```

```
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

Client License: LGPLv2+

```
                  GNU LIBRARY GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1991 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the library GPL.  It is
 numbered 2 because it goes with version 2 of the ordinary GPL.]

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Library General Public License, applies to some
specially designated Free Software Foundation software, and to any
other libraries whose authors decide to use it.  You can use it for
your libraries, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if
you distribute copies of the library, or if you modify it.

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link a program with the library, you must provide
complete object files to the recipients so that they can relink them
with the library, after making changes to the library and recompiling
it.  And you must show them these terms so they know their rights.

  Our method of protecting your rights has two steps: (1) copyright
the library, and (2) offer you this license which gives you legal
permission to copy, distribute and/or modify the library.
```

```
  Also, for each distributor's protection, we want to make certain
that everyone understands that there is no warranty for this free
library.  If the library is modified by someone else and passed on, we
want its recipients to know that what they have is not the original
version, so that any problems introduced by others will not reflect on
  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that companies distributing free
software will individually obtain patent licenses, thus in effect
transforming the program into proprietary software.  To prevent this,
we have made it clear that any patent must be licensed for everyone's
free use or not licensed at all.
  Most GNU software, including some libraries, is covered by the ordinary
GNU General Public License, which was designed for utility programs.  This
license, the GNU Library General Public License, applies to certain
designated libraries.  This license is quite different from the ordinary
one; be sure to read it in full, and don't assume that anything in it is
the same as in the ordinary license.
  The reason we have a separate public license for some libraries is that
they blur the distinction we usually make between modifying or adding to a
program and simply using it.  Linking a program with a library, without
changing the library, is in some sense simply using the library, and is
analogous to running a utility program or application program.  However, in
a textual and legal sense, the linked executable is a combined work, a
derivative of the original library, and the ordinary General Public License
treats it as such.
  Because of this blurred distinction, using the ordinary General
Public License for libraries did not effectively promote software
sharing, because most developers did not use the libraries.  We
concluded that weaker conditions might promote sharing better.
  However, unrestricted linking of non-free programs would deprive the
users of those programs of all benefit from the free status of the
libraries themselves.  This Library General Public License is intended to
permit developers of non-free programs to use free libraries, while
preserving your freedom as a user of such programs to change the free
libraries that are incorporated in them.  (We have not seen how to achieve
this as regards changes in header files, but we have achieved it as regards
changes in the actual functions of the Library.)  The hope is that this
will lead to faster development of free libraries.
  The precise terms and conditions for copying, distribution and
modification follow.  Pay close attention to the difference between a
"work based on the library" and a "work that uses the library".  The
former contains code derived from the library, while the latter only
works together with the library.
  Note that it is possible for a library to be covered by the ordinary
General Public License rather than by this special one.
                    GNU LIBRARY GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
  0. This License Agreement applies to any software library which
contains a notice placed by the copyright holder or other authorized
party saying it may be distributed under the terms of this Library
General Public License (also called "this License").  Each licensee is
addressed as "you".
  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.
  The "Library", below, refers to any such software library or work
which has been distributed under these terms.  A "work based on the
Library" means either the Library or any derivative work under
```

```
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language.  (Hereinafter, translation is
included without limitation in the term "modification".)
  "Source code" for a work means the preferred form of the work for
making modifications to it.  For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control compilation
and installation of the library.
  Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it).  Whether that is true depends on what the Library does
and what the program that uses the Library does.
  1. You may copy and distribute verbatim copies of the Library's
complete source code as you receive it, in any medium, provided that
you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.
  You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.
  2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:
    a) The modified work must itself be a software library.
    b) You must cause the files modified to carry prominent notices
    stating that you changed the files and the date of any change.
    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.
    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.
    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application.  Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)
These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.
```

```
Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.
In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.
  3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library.  To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
instead of to this License.  (If a newer version than version 2 of the
ordinary GNU General Public License has appeared, then you can specify
that version instead if you wish.)  Do not make any other change in
these notices.
  Once this change is made in a given copy, it is irreversible for
that copy, so the ordinary GNU General Public License applies to all
subsequent copies and derivative works made from that copy.
  This option is useful when you wish to copy part of the code of
the Library into a program that is not a library.
  4. You may copy and distribute the Library (or a portion or
derivative of it, under Section 2) in object code or executable form
under the terms of Sections 1 and 2 above provided that you accompany
it with the complete corresponding machine-readable source code, which
must be distributed under the terms of Sections 1 and 2 above on a
medium customarily used for software interchange.
  If distribution of object code is made by offering access to copy
from a designated place, then offering equivalent access to copy the
source code from the same place satisfies the requirement to
distribute the source code, even though third parties are not
compelled to copy the source along with the object code.
  5. A program that contains no derivative of any portion of the
Library, but is designed to work with the Library by being compiled or
linked with it, is called a "work that uses the Library".  Such a
work, in isolation, is not a derivative work of the Library, and
therefore falls outside the scope of this License.
  However, linking a "work that uses the Library" with the Library
creates an executable that is a derivative of the Library (because it
contains portions of the Library), rather than a "work that uses the
library".  The executable is therefore covered by this License.
Section 6 states terms for distribution of such executables.
  When a "work that uses the Library" uses material from a header file
that is part of the Library, the object code for the work may be a
derivative work of the Library even though the source code is not.
Whether this is true is especially significant if the work can be
linked without the Library, or if the work is itself a library.  The
threshold for this to be true is not precisely defined by law.
  If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small inline
functions (ten lines or less in length), then the use of the object
file is unrestricted, regardless of whether it is legally a derivative
work.  (Executables containing this object code plus portions of the
Library will still fall under Section 6.)
  Otherwise, if the work is a derivative of the Library, you may
distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
```

```
whether or not they are linked directly with the Library itself.
  6. As an exception to the Sections above, you may also compile or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer's own use and reverse
engineering for debugging such modifications.
  You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License.  You must supply a copy of this License.  If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License.  Also, you must do one
of these things:
    a) Accompany the work with the complete corresponding
    machine-readable source code for the Library including whatever
    changes were used in the work (which must be distributed under
    Sections 1 and 2 above); and, if the work is an executable linked
    with the Library, with the complete machine-readable "work that
    uses the Library", as object code and/or source code, so that the
    user can modify the Library and then relink to produce a modified
    executable containing the modified Library.  (It is understood
    that the user who changes the contents of definitions files in the
    Library will not necessarily be able to recompile the application
    to use the modified definitions.)
    b) Accompany the work with a written offer, valid for at
    least three years, to give the same user the materials
    specified in Subsection 6a, above, for a charge no more
    than the cost of performing this distribution.
    c) If distribution of the work is made by offering access to copy
    from a designated place, offer equivalent access to copy the above
    specified materials from the same place.
    d) Verify that the user has already received a copy of these
    materials or that you have already sent this user a copy.
  For an executable, the required form of the "work that uses the
Library" must include any data and utility programs needed for
reproducing the executable from it.  However, as a special exception,
the source code distributed need not include anything that is normally
distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on
which the executable runs, unless that component itself accompanies
the executable.
  It may happen that this requirement contradicts the license
restrictions of other proprietary libraries that do not normally
accompany the operating system.  Such a contradiction means you cannot
use both them and the Library together in an executable that you
distribute.
  7. You may place library facilities that are a work based on the
Library side-by-side in a single library together with other library
facilities not covered by this License, and distribute such a combined
library, provided that the separate distribution of the work based on
the Library and of the other library facilities is otherwise
```

```
permitted, and provided that you do these two things:
    a) Accompany the combined library with a copy of the same work
    based on the Library, uncombined with any other library
    facilities.  This must be distributed under the terms of the
    Sections above.
    b) Give prominent notice with the combined library of the fact
    that part of it is a work based on the Library, and explaining
    where to find the accompanying uncombined form of the same work.
  8. You may not copy, modify, sublicense, link with, or distribute
the Library except as expressly provided under this License.  Any
attempt otherwise to copy, modify, sublicense, link with, or
distribute the Library is void, and will automatically terminate your
rights under this License.  However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.
  9. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Library or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.
  10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.
  11. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all.  For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Library.
If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.
This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.
  12. If the distribution and/or use of the Library is restricted in
```

```
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License may add
an explicit geographical distribution limitation excluding those countries,
so that distribution is permitted only in or among countries not thus
excluded.  In such case, this License incorporates the limitation as if
written in the body of this License.
  13. The Free Software Foundation may publish revised and/or new
versions of the Library General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number.  If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation.  If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.
  14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission.  For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this.  Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.
                            NO WARRANTY
  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU.  SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.
                     END OF TERMS AND CONDITIONS
           How to Apply These Terms to Your New Libraries
  If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change.  You can do so by permitting
redistribution under these terms (or, alternatively, under the terms of the
ordinary General Public License).
  To apply these terms, attach the following notices to the library.  It is
safest to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least the
```

```
"copyright" line and a pointer to where the full notice is found.
    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Library General Public
    License as published by the Free Software Foundation; either
    version 2 of the License, or (at your option) any later version.
    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    Library General Public License for more details.
    You should have received a copy of the GNU Library General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
Also add information on how to contact you by electronic and paper mail.
You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary.  Here is a sample; alter the names:
  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library `Frob' (a library for tweaking knobs) written by James Random Hacker.
  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice
That's all there is to it!
the original authors' reputations.
```

CHAPTER 13

Development

Getting Involved

Development Location

The code for commissaire lives on GitHub. The main repo can be found at https://github.com/projectatomic/commissaire.

Development Setup

See DEVEL.rst

Vagrant

A `Vagrantfile` is provided which will give you a full local development setup.

To run the vagrant development environment make sure you have a supported virtualization system, vagrant and vagrant-sshfs installed, and have all commissaire projects checked out in the parent folder as the commissaire vagrant box will attempt to mount them over SSH.

```
$ ls ../ | grep 'commissaire'
commissaire
commissaire-http
commissaire-service
$
```

To run the vagrant development environment make sure you have a support virtualization system as well as vagrant installed and execute `./tools/vagrantup`.

Warning: If you want to use the `vagrant` command directly note that you will have to follow the same start up process used in `./tools/vagrantup`

Note: If you decide to use the `vagrant` command directly, the `fedora-atomic` host will require a manual workaround to mount the shared folder at `/home/vagrant/sync`. After the box is up the first time, run `vagrant ssh fedora-atomic` to log into the virtual machine, then run `sudo rpm-ostree install fuse-sshfs`. Exit back out to the host machine and restart the virtual machine with `vagrant reload fedora-atomic`.

Note: You will need to add an ssh pub key to `/root/.ssh/authorized_keys` on nodes if you will not be using `cloud-init` for bootstrapping.

Server	IP	OS	AutoStart
Servers (etcd/redis)	192.168.152.101	Fedora Cloud 25	Yes
Fedora Node	192.168.152.110	Fedora Cloud 25	Yes
Fedora Atomic Node	192.168.152.111	Fedora Atomic 25	Yes
Commissaire	192.168.152.100	Fedora Cloud 25	No
Kubernetes	192.168.152.102	Fedora Cloud 25	No

For more information see the Vagrant site.

Getting Up To Speed

As you can see commissaire uses a number of libraries.

```
setuptools #license=MIT
sphinx_rtd_theme #license=BSD
python-etcd #license=MIT
requests #license=ASLv2.0
kombu #license=BSD
```

Of these, the most important to be up to speed on are:

- ansible: https://www.ansible.com/

Standards

Conventions

Code

Like most projects commissaire expects specific coding standards to be followed. pep8 is followed strictly with the exception of E402: module level import not at top of file.

Commissaire Proposal Document

A Commissaire Proposal Document (CPD) must be submitted and approved before significantly changing the current implementation. This applies to changes which break backward compatibility, replace a subsystem, change the user experience, etc.. For information on the CPD process see CPD-1: CPD Process and the CPD Template.

Ansible Templates

Variables are used with jinja2 templates and should always prefix **commissaire_**. Here is a current list variables in use as examples:

Name	Description
commissaire_targets	Host(s) to target
commissaire_install_libselinux_python	Command to install libselinux-python
commissaire_install_flannel	Command to install flannel
commissaire_flanneld_config_local	Local flannel config file to template to the target(s)
commissaire_flanneld_config	Remote path to the flannel config
commissaire_flannel_service	Name of the flannel service
commissaire_install_docker	Command to install docker
commissaire_docker_config_local	Local docker config file to template to the target(s)
commissaire_docker_config	Remote path to the docker config
commissaire_docker_service	Name of the docker service
commissaire_install_kube	Command to install kubernetes minion packages
commissaire_kubernetes_config_local	Local kubernetes config file to template to the target(s)
commissaire_kubernetes_config	Remote path to the kubernetes config
commissaire_kubeconfig_config_local	Local kubeconfig file to template to the target(s)
commissaire_kubeconfig_config	Remote path to the kubeconfig
commissaire_kubelet_service	Name of the kubelet service
commissaire_kubeproxy_service	Name of the kubernetes proxy service
commissaire_restart_command	Host restart command
commissaire_upgrade_command	Host upgrade command
commissaire_bootstrap_ip	The IP address of the host
commissaire_kubernetes_api_server_url	The kubernetes api server (scheme://host:port)
commissaire_kubernetes_client_cert_path	Path to the kubernetes client certificate
commissaire_kubernetes_client_key_path	Path to the kubernetes client key
commissaire_kubernetes_client_cert_path_local	Path to the local kubernetes client certificate
commissaire_kubernetes_client_key_path_local	Path to the local kubernetes client key
commissaire_kubernetes_bearer_token	The bearer token used to contact kubernetes
commissaire_docker_registry_host	The docker registry host
commissaire_docker_registry_port	The docker registry port
commissaire_etcd_server_url	The etcd server (scheme://host:port)
commissaire_etcd_ca_path	Path to the etcd certificate authority
commissaire_etcd_client_cert_path	Path to the etcd client certificate
commissaire_etcd_client_key_path	Path to the etcd client key
commissaire_etcd_ca_path_local	Path to the local etcd certificate authority
commissaire_etcd_client_cert_path_local	Path to the local etcd client certificate
commissaire_etcd_client_key_path_local	Path to the local etcd client key
commissaire_flannel_key	The flannel configuration key

Testing

Unit Testing

commissaire uses TravisCI to verify that all unit tests are passing. **All unit tests must pass and coverage must be above 80% before code will be accepted. No exceptions.**.

To run unit tests locally and see where your code stands:

```
$ tox
...
```

End-to-End/BDD Testing

commissaire uses Behave to execute end to end/BDD tests. You will need to have the following in your parent directory to properly be able to execute tests locally.

```
$ ls ../ | grep 'comm'
commctl
commissaire
commissaire-http
commissaire-service
$
```

To run e2e/bdd tests locally and see where your code stands:

```
(virtualenv)$ behave -D start-all-servers
...
```

Note: you can pass `-D commissaire-server-args=""` to append server arguments when starting the server from behave.

or via `tox`

```
(virtualenv)$ tox -e bdd
...
```

You can also run the tests against any commissaire/etcd instance directly.

Warning: Do **not** point to a real instance of commissaire. e2e/BDD tests will simulate real usage on a running server and will probably cause damage.

See manual_installation for how to set up commissaire.

```
# Set up ...
(virtualenv)$ behave \
    -D commissaire-server=http://127.0.0.1:8000 \
    -D etcd=http://127.0.0.1:2379 \
    -D bus-uri=redis://127.0.0.1:6379
...
```

If you are using our vagrant set up you can use the `use-vagrant` argument.

```
(virtualenv)$ ./tools/vagrantup
...
(virtualenv)$ behave -D use-vagrant
...
```

Here are all of the user arguments supported by using the `-D` options:

```
- commissaire-server: The URI of the server to use.
- etcd: The URI of a running etcd to use.
- bus-uri: The URI of a bus service to use.
- use-vagrant: If vagrant is in use. Ignores start-* items.
- start-all-servers: Starts everything (like setting all start-* items).
- start-etcd: If etcd should be started.
- start-custodia: If custodia should be started.
- start-redis: If redis should be started. Also sets BUS_URI.
- start-storage-service: If commissaire-storage-service should start.
- start-investigator-service: If commissaire-investigator-service should start.
- start-watcher-service: If commissaire-watcher-service should start.
- start-commissaire-server: If the commissaire-server should start.
```

There are a number of tags within the tests. Using these tags can target specific parts of the codebase without running the full suite. Use `-t` to specify tags. `-k` is also helpful as it will suppress showing the tests that did not run. Using a `~` before the tag will disable all test with that tag. See behave `--tags-help` for more details

Tag	Description
anonymous	Tests without authentication
clientcert	Tests that use a client certificate
cluster	Tests that are specific to cluster functionality
clusterexec	Tests that use the clusterexec code
create	Tests that create a resource
delete	Tests that delete a resource
deploy	Tests which use the deploy functionality
hosts	Tests that are specific to the hosts functionality
list	Tests that list a resource
recreate	Tests that recreate a resource
restart	Tests which use the restart functionality
retrieve	Tests the get a resource
slow	Tests that are known to run slow
ssh	Tests which use the ssh related functionality
status	Tests that are specific to the status functionality
upgrade	Tests which use the upgrade functionality

The following command shows how to run all the create tests that are not marked slow:

```
# Set up ...
(virtualenv)$ behave -k -t create,~slow \
    -D commissaire-server=http://127.0.0.1:8000 \
    -D etcd=http://127.0.0.1:2379 \
    -D bus-uri=redis://127.0.0.1:6379
...
```

The same thing using the `./tools/behave` script:

```
# Set up ...
(virtualenv)$ ./tools/behave -t create,~slow
...
```

Authentication Plugins

commissaire's authentication is handled by a simple WSGI based plugin based system. To create a new authentication plugin you must:

- subclass `commissaire_http.authentication.Authenticator`
- name the class `PluginClass`
- override the `authenticate` method

If you need to have configuration items passed when used you will also need to override `__init__` adding in keyword arguments.

Note: The `authenticate` should always return `True` for success, `False` for general failure, or handle responses itself as a WSGI application.

Examples

Basic

```
from commissaire_http.authentication import Authenticator

class AlwaysAllowOnSSL(Authenticator):
    """
    Example: Allows anyone if they use https.
    """

    def authenticate(self, environ, start_response):
        """
        Allows access if https is in use.

        :param environ: WSGI environment instance.
        :type environ: dict
        :param start_response: WSGI start response callable.
        :type start_response: callable
        :returns: True on success, False on failure
        :rtype: bool
        """
        if environ.get('wsgi.url_scheme', 'http') == 'https':
            return True
        return False

#: Alias AlwaysAllowOnSSL
PluginClass = AlwaysAllowOnSSL
```

As a WSGI Application

```
from commissaire_http.authentication import Authenticator

class AlwaysAllowOnSSL(Authenticator):
    """
    Example: Allows anyone if they use https but pretends to be a teapot
    if they use http.
    """

    def authenticate(self, environ, start_response):
        """
        Allows access if https is in use.
```

```
        :param environ: WSGI environment instance.
        :type environ: dict
        :param start_response: WSGI start response callable.
        :type start_response: callable
        :returns: True on success, False on failure
        :rtype: bool
        """
        if environ.get('wsgi.url_scheme', 'http') == 'https':
            return True
        start_response("418 I'm a teapot", [('content-type', 'text/plain')])
        return [bytes('Whiiiiieee', 'utf8')]

#: Alias AlwaysAllowOnSSL
PluginClass = AlwaysAllowOnSSL
```

Real Code

See httpauthclientcert

Writing a Commissaire Service

High Level

- Subclass `commissaire_service.service.CommissaireService`
- Define all `on_{{ method }}` methods to exposed them on the message bus
- Define how to run the service (Directly or via a ServiceManager)

Specifics

Create the Subclass

All Commissaire Services must subclass (or reimplement the functionality of...) `commissaire_service.service.CommissaireService`.

```
from commissaire_service.service import CommissaireService


class MyService(CommissaireService):
    """
    This is MyService.
    """
    pass
```

Define Exposed Methods

`CommissaireService` uses the `on_{{ method }}` convention for exposing methods to remote callers. If you wanted to expose a method as `ping` you would define a method on your service called `on_ping`. `on_{{ method`

}}'s expect to take 1 or more arguments where the first required argument is `message` which is the message itself in the case the method needs extra information.

To return results back to the caller via the message bus simply use the `return` statement as if it was a normal method. If there is an error, `raise` the proper exception. These will be transformed into proper messages and returned to the message bus and passed to the caller.

Note: message must always be the **first** argument!

```
def on_ping(self, message):
    """
    Exposed as ping. Takes no bus arguments.
    """
    return 'pong'

def on_echo(self, message, words):
    """
    Exposed as echo. Takes one bus argument of words.
    """
    return words

def on_fail(self, message):
    """
    Exposed as fail. Takes no bus arguments.
    """
    raise NotImplementedError('I was never created')
```

Storage Integration

Most services will want to interact with Commissaire's `Storage` service in some way; perhaps to read or update records in permanent storage or just be notified of changes. This is such a common case that Commissaire provides a convenience class named `StorageClient` to easily interact with the `Storage` service.

The `StorageClient` API uses `Model` instances as inputs and outputs, and handles all the JSON encoding and decoding for you.

```
from commissaire import models
from commissaire.storage import client


class MyService(CommissaireService):
    """
    This demonstrates how to use StorageClient.
    """

    def __init__(self, exchange_name, connection_url, config_file=None):
        ...
        # Chain up to CommissaireService.__init__()
        ...

        # StorageClient requires a BusMixin interface, which
        # our parent class -- CommissaireService -- provides.
        self.storage = client.StorageClient(self)

        # Invoke a method when a new Host record is created.
```

```
        #
        # Can also listen for: client.NOTIFY_EVENT_DELETED
        #                      client.NOTIFY_EVENT_UPDATED
        #                      client.NOTIFY_EVENT_ANY
        self.storage.register_callback(
            self.host_created_cb, models.Host,
            client.NOTIFY_EVENT_CREATED)

    @client.NotifyCallback
    def host_created_cb(self, event, model, message):
        # "event" will always be "created" since we specified
        # client.NOTIFY_EVENT_CREATED when registering this
        # callback (see above).
        #
        # "model" will always be a models.Host instance since
        # we specified it when registering this callback (see
        # above). We could have also passed None to catch the
        # creation of any type of record in permanent storage.
        pass

    def on_do_something_cool(self, message, host):
        # Fetch a cluster for some reason.  storage.get_cluster()
        # returns a models.Cluster instance instead of a bunch of
        # JSON to decode.
        cluster = self.storage.get_cluster('my_cluster')
        ...
        # Do something cool with the requested host.
        ...

        # Say we updated the state of the models.Host instance.
        # This writes the updated state back to permanent storage.
        self.storage.save(host)
```

Running the Service

The simplest way to run a `CommissaireService` is to create an instance and use it's `run` method.

```
#: The arguments used to create new kombu.Queue instances
queue_kwargs = [
    {'name': 'my_queue', 'routing_key': 'queues.my_queue.*'},
]

try:
    MyService(
        exchange_name='my_exchange',
        connection_url='redis://127.0.0.1:6379/',
        qkwargs=queue_kwargs
    ).run()
except Exception as error:
    # Handle it ;-)
    pass
```

A more likely pattern is to run multiple instances of a service on the same queue to be able to handle more requests. This can be done by wrapping the service in a `ServiceManager`. As you can see it follows a similar pattern as the `CommissaireService` prepending a few inputs required for running multiple processes.

Note: Debugging with multiple processes can be much harder. If you need to debug a service it is recommend to use the `CommissaireService` directly to ensure no `Exception` information gets eaten up between the process pool and service.

```
#: The arguments used to create new kombu.Queue instances
queue_kwargs = [
    {'name': 'my_queue', 'routing_key': 'queues.my_queue.*'},
]

try:
    ServiceManager(
        service_class=MyService,
        process_count=3,
        exchange_name='my_exchange',
        connection_url='redis://127.0.0.1:6379/',
        qkwargs=queue_kwargs
    ).run()
except Exception as error:
    pass
```

Code Example

See simpleservice.

Developing on Commissaire HTTP

Commissaire HTTP provides a multithreaded REST interface into Commissaire functionality. The server is broken up into 5 main parts: `Router`, `Dispatcher`, `Function Handler`, `Class Handler`, and the `CommissaireHttpServer` itself.

Router

The `Router` maps URI paths to controllers. The following example would route the path `/hello/` to the controller at `commissaire_http.handlers.hello_world` if the HTTP method is `GET`.

```
mapper = Router()
mapper.connect(
    '/hello/',
    controller='commissaire_http.handlers.hello_world',
    conditions={'method': 'GET'})
```

Dispatcher

The `Dispatcher` uses a `Router` and, as it's name suggests, dispatches the requests to the proper controller. It also takes care of loading handlers. The following example creates a new `Dispatcher` instance using a previously created `Mapper`. It will load handlers found in the `commissaire_http.handlers` and `mypackage.handlers` packages.

```
dispatcher = Dispatcher(
    mapper,
    handler_packages=[
     'commissaire_http.handlers',
     'mypackage.handlers'])
```

Handlers

`Handlers`, also called controllers, do the majority of the business logic. A `Handler` can be a function or a class, but must follow a specific convention so the `Dispatcher` knows it's valid during loading.

Function Handler

Function Handlers must take two parameters: `message` and `bus`. The first input, `message`, is the jsonrpc message. The second input, `bus` will either be a valid connection to the bus or, if the bus is not enabled, `None`.

When referencing a Function Handler as a controller use the full package path to the function. If the function is `hello_world` and it lives under `commissaire_http.handlers` then the controller would be `commissaire_http.handlers.hello_world`.

The following example would show the user `{"Hello": "there"}` or `{"Hello", "{{ name }}"}` depending on parameters. Remember, the return of the handler must be a valid jsonrpc message as well!

Note: The `method` in the incoming jsonrpc message is hijacked and filled with the HTTP method that was used to call the handler.

```
def hello_world(message, bus):
    """
    Example function handler that simply says hello. If name is give
    in the query string it uses it.

    :param message: jsonrpc message structure.
    :type message: dict
    :returns: A jsonrpc structure.
    :rtype: dict
    """
    response_msg = {'Hello': 'there'}
    if message['params'].get('name'):
        response_msg['Hello'] = message['params']['name']
    return {
        'jsonrpc': '2.0',
        'id': message['id'],
        'result': response_msg,
    }
```

Class Handler

A Class Handler is not much different than a Function Handler. Instead of defining a single function, a class is declared with methods that take three parameters: `self`, `message`, and `bus`. If the method should not be considered a handler it must start with an underscore.

One major difference between a Class Handler and Function Handler is that Class Handlers are instantiated when they are loaded!

When referencing a Class Handler as a controller, use the full package path to the class and the method. If the class is `ClassHandlerExample`, the method is `hello`, and it lives under `commissaire_http.handlers` then the controller would be `commissaire_http.handlers.ClassHandlerExample.hello`.

The following example exposes `hello` in the same way as the Function Handler example above. It then uses `hello_world` to do the heavy lifting.

```
class ClassHandlerExample:
    """
    Example class based handlers.
    """

    def hello(self, message, bus):
        """
        Example method handler that simply says hello. If name is given
        in the query string it uses it.

        :param message: jsonrpc message structure.
        :type message: dict
        :returns: A jsonrpc structure.
        :rtype: dict
        """
        return hello_world(message, bus)

    def _ignored(self):
        """
        This method would not be loaded as a handler but could be used by
        handlers in this class.
        """
        return 'I am ignored.'
```

CommissaireHttpServer

In the following example, a `CommissaireHttpServer` is created which binds to address 127.0.0.1 and port 8000 and uses a previously created `Dispatcher`. It then is set to run (block) until killed.

```
server = CommissaireHttpServer('127.0.0.1', 8000, dispatcher)
server.serve_forever()
```

Code Example

See http_server.

Building Packages

RPM

commissaire's spec file is located in the Fedora package repo.

Generate the Source Distribution

```
(virtualenv)$ ./setup.py sdist
...
```

Move Source Distribution To RPM Source

Note: Your rpmbuild root may be different! Check with your distribution.

```
(virtualenv)$ mv dist/*.tar.gz ~/rpmbuild/SOURCES/`./setup.py --version`.tar.gz
```

Build The Package

```
(virtualenv)$ rpmbuild -ba contrib/package/rpm/commissaire.spec
...
```

Your package should now be output in your rpmbuild root's `RPMS/noarch/` and `SRPMS` directories.

Cloud-Init Integration

Commissaire provides a `commctl` command to generate a `user-data` file for `cloud-init` that automatically registers hosts to the Commissaire server during bootup. This command is aptly named `user-data`.

commctl user-data command

```
usage: commctl user-data [-h] -e ENDPOINT [-c CLUSTER] [-u USERNAME]
                                  [-p] [-r REMOTE_USER] [-s SSH_KEY_PATH]
                                  [-a AUTHORIZED_KEYS_PATH] [-C CLOUD_INIT]
                                  [-o OUTFILE]

optional arguments:
  -h, --help            show this help message and exit
  -e ENDPOINT, --endpoint ENDPOINT
                        Commissaire endpoint to use during bootstrapping
  -c CLUSTER, --cluster CLUSTER
                        Name of the cluster for new hosts to join
  -u USERNAME, --username USERNAME
                        Commissaire user to use when bootstrapping
  -p, --password        Prompts for a Commissaire password to use when
                        bootstrapping
  -r REMOTE_USER, --remote-user REMOTE_USER
                        Remote user to provide to Commissaire for ssh access
  -s SSH_KEY_PATH, --ssh-key-path SSH_KEY_PATH
                        Path to the private key of the remote user
  -a AUTHORIZED_KEYS_PATH, --authorized-keys-path AUTHORIZED_KEYS_PATH
                        Path to the authorized_keys file of the remote user
  -C CLOUD_INIT, --cloud-init CLOUD_INIT
                        cloud-init.txt file to use
```

```
  -o OUTFILE, --outfile OUTFILE
                        Output file. If omitted STDOUT is used

Example: commctl user-data -p -c my_cluster -o cluster.userdata
```

Create the User-Data File

Let's say you have the following properties:

- A Commissaire username of USER
- A Commissaire password of PASS
- A Commisaire cluster you want new hosts to join called CLUSTER
- A Commissaire REST Server listening at https://example.com/
- The expectation of having the `user-data` file at `./CLUSTER.userdata`

You would create the `user-data` file like so:

```
$ commctl user-data \
  --password \
  --username USER \
  --cluster CLUSTER \
  --endpoint https://example.com/ \
  --outfile CLUSTER.userdata
Password: <PASS>
$ # Let's check that the userdata file is indeed a multipart/mixed file
$ file CLUSTER.userdata
CLUSTER.userdata: multipart/mixed; boundary="===============8094544984785845936==,␣
↪ASCII text
```

CPDs

CPD-1: CPD Process

Metadata

- CPD Version: 1
- Status: Accepted

Description

Commissaire Proposal Documents (CPD) provide a consistent way to propose large changes to the project for review.

Rationale

Most changes to Commissaire are small, iterative enhancements and bug fixes. When a larger change to the project may make sense a CPD provides a formalized way to propose the change, request review, and refine the idea until it is either accepted or rejected.

Design

CPD Process

The following process should be followed when a CPD is needed:

1. Open up an Issue with a brief description
2. Note in the Issue that a CPD will be created
3. Create an initial CPD
4. Update the Issue with a link to the CPD and request feedback
5. Update the CPD as needed and ask for feedback
6. Accepted/Closed Phase
   - If 75% or more of the active development team give the CPD a `:+1:` it is Approved
   - If 50% or more of the active development team disagrees with the CPD it is Closed
   - If the person proposing the CPD no longer wishes to continue they can request it to be Closed
   - If none of the above is met the cycle can continue to 5.
7. The current development lead(s) pull in the CPD to the docs folder and update the status
8. The current development lead(s) update the issue with the result

Naming

Each CPD will have a unique number associated with it. As an example, this CPD will have the number `1` and should be referenced as `CPD-1`. The CPD number shall be the same as any issue number opened. As an example, if there is an issue #10 that needs a larger design then the CPD would be `CPD-10`.

Outline of a CPD

Label	Parent Label	Description
Name	None	CPD name. Ex: `CPD-1`.
Metadata	Name	CPD CPD Version, Status (`Open`, `Closed`, `Accepted`)
Description	Name	Short description of the CPD.
Rational	Name	Why there is a need to make the change.
Design	Name	Deep dive into changes. May have subsections.
Checklist	Name	Important items to note.
User Story	Name	User story that would be used to implement the change.
Acceptance Criteria	Name	Criteria that must be met for the change to be considered complete.
References	Name	Any helpful external links.

CPD Status

The status of a CPD will be changed to Accepted if and when 75% or more of the active core development team gives the CPD a `:+1:`. It is the job of at least one of the development leads to update the CPD to Accepted status and note it is accepted in the related issue(s).

Checklist

- breaks API backward compatibility
- breaks user interaction backward compatibility
- requires new or replaces current libraries

User Story

As a developer on Commissaire I want a formalized way to propose large changes so that the larger group can help refine the ultimate solution.

Acceptance Criteria

- Verify that a template for proposals is created
- Verify that a proposal that describes the process is created
- Verify that documentation is updated

References

- Kubernetes Proposals
- Golang Proposals
- PEP Proposal
- JSR

CPD-101: Key Storage Encryption

Metadata

- CPD Version: 1
- Status: Accepted

Description

Today we are holding keys the same way that secrets are used in some container managers. Instead of holding keys in base64 and assuming that the `Storage` instance is used only for Commissaire, we could encrypt keys, credentials and other secrets to add another layer of safety.

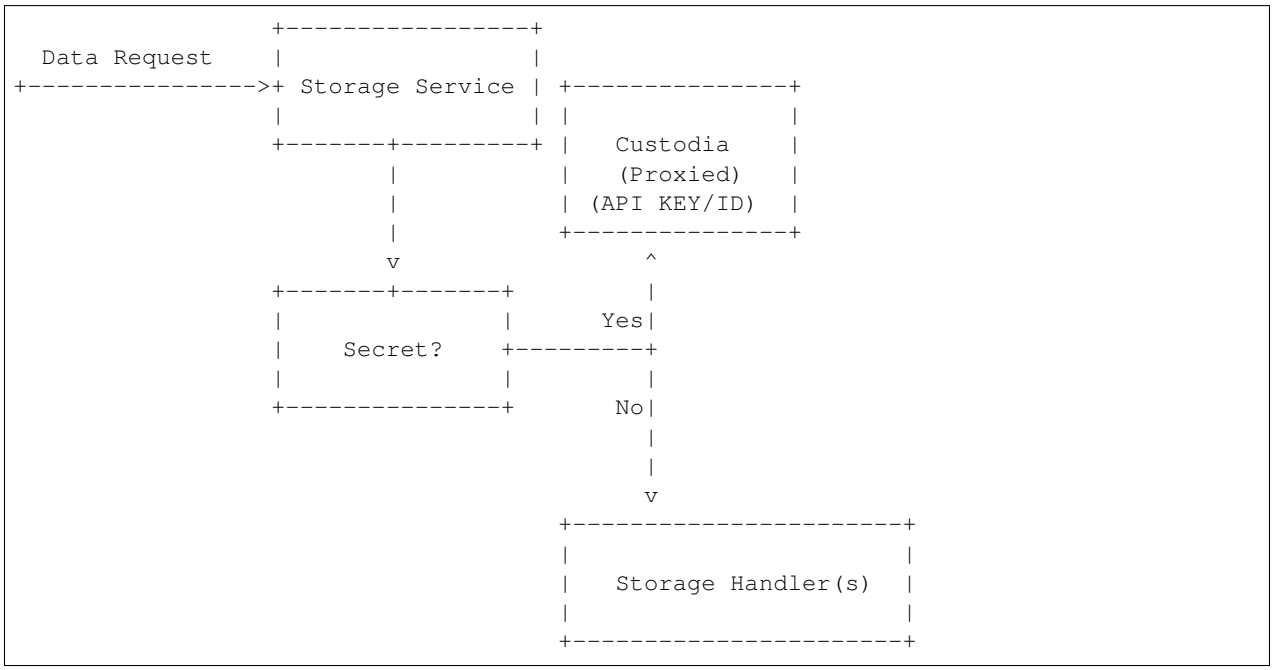
Rationale

The likelihood of having a `Storage` system that is used only for Commissaire seems low. More than likely the same instance will be used for other applications as well. By adding encryption to sensitive data we could mitigate access from those with direct access to data dumps and storage systems.

Design

The `StorageService` would be updated to know what data would be backed through Custodia rather than the other storage handler(s).

- Install and setup of a Custodia instance would be part of a Commissaire install and configuration.
- Custodia would be configured to use an etcd backend.
- Custodia would be configured to use unix socket communication.
- Commissaire would proxy access to Custodia and enforce it's authentication
- Commissaire would generate an api id and api key for StorageService access
- StorageService would have an api id and api key in it's config to access Custodia
- Commissaire's Storage Service would be updated to store credentials and ssh keys via Custodia.



Additional Libraries

Custodia would be a required subsystem. Custodia would be installed as part of Commissaire.

StorageService Updates

The StorageService would need to to know when to use Custodia versus the configured `StorageHandler``(s)`. It would look at the ``_secrets` attribute on the instance and, if set to `True` would use the secrets handler.

The secrets handler would be automatically added to the `StorageHandlerManager` and would require no special configuration by the operator. However, additional configuration keys **would** be added so that the secrets handler could authenticate to the secrets store.

Lastly, the `StorageService` would need to have a way to query for the secrets api endpoint. There are many possible designs for this. The decision is left up to the implementation.

New HTTP Handler

A new handler called `secrets` could be added. This would proxy requests back to the Custodia instance.

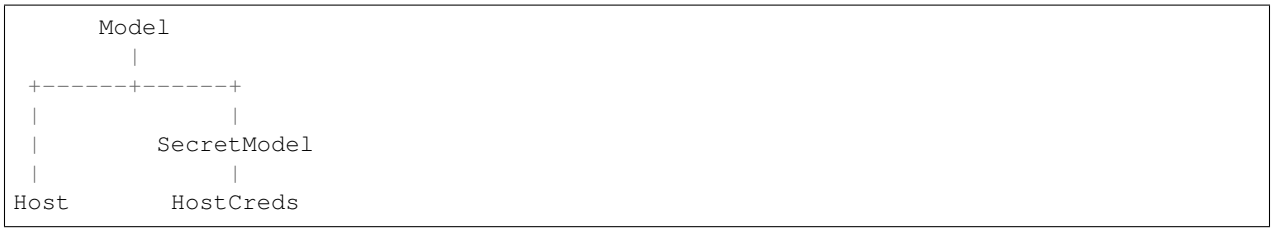
AuthenticationManager Updates

A way to allow for proxied authentication will be required. This can be done by providing a list of self authenticated endpoints which bypasses the authentication stack and sends the request directly through to the handler.

Model Updates

Sensitive items would be pulled out from the `Host` model into it's own model. For simplicity, the model should be named after the REST endpoint that has traditionally returned the data: `HostCreds`. The models would match based on their primary keys: `address`.

The `Model` would add a subclass which would be used to house secrets. This new subclass would be called `SecretModel` and would always have it's contents stored in the secrets store.



Example Code

These are examples and likely will not work without modification.

Model Updates

```
class SecretModel(Model):
    """
    Parent class for all models which must be stored in the secrets store.
    """
    pass

class Host(Model):
    """
    Representation of a Host.
    """
    _json_type = dict
    _attribute_map = {
        'address': {'type': str},
        'status': {'type': str},
        'os': {'type': str},
        'cpus': {'type': int},
        'memory': {'type': int},
        'space': {'type': int},
        'last_check': {'type': str},
        'source': {'type': str},
    }
    _attribute_defaults = {
```

```
        'address': '', 'status': '', 'os': '', 'cpus': 0,
        'memory': 0, 'space': 0, 'last_check': '', 'source': ''}
    _primary_key = 'address'


class HostCreds(SecretModel):
    """
    Representation of Host credentials.
    """
    _json_type = dict
    _attribute_map = {
        'address': {'type': str},
        'ssh_priv_key': {'type': str},
        'remote_user': {'type': str},
    }
    _attribute_defaults = {
        'ssh_priv_key': '',
        'remote_user': 'root',
    }
    _primary_key = 'address'
```

StorageHandlerManager Updates

```
def _get_handler(self, model):
    """
    Looks up, and if necessary instantiates, a StoreHandler instance
    for the given model.  If the model stores secrets the secrets
    handler is used. Raises KeyError if no handler is registered
    for that type of model.
    """
    if issubclass(model.__class__, models.SecretModel):
        handler = self._handlers.get('secret')  # Just an example
    else:
        handler = self._handlers.get(type(model))

    if handler is None:
        # Let this raise a KeyError if the registry lookup fails.
        handler_type, config, model_types = self._registry[type(model)]
        handler = handler_type(config)
        self._handlers.update({mt: handler for mt in model_types})
    return handler
```

Secrets Handler

```
def _register(router):
    """
    Sets up routing for secrets.

    :param router: Router instance to attach to.
    :type router: commissaire_http.router.Router
    :returns: The router.
    :rtype: commissaire_http.router.Router
    """
    from commissaire_http.constants import ROUTING_RX_PARAMS

    router.connect(
        R'/api/v0/secrets/',
        controller=proxy_secrets,
```

```
        conditions={'method': ['GET', 'PUT', 'POST', 'DELETE']})

@BasicHandler
def proxy_secrets(message, bus):
    """
    Proxy secrets back to Custodia

    :param message: jsonrpc message structure.
    :type message: dict
    :param bus: Bus instance.
    :type bus: commissaire_http.bus.Bus
    :returns: A jsonrpc structure.
    :rtype: dict
    """
    try:
        # Use unix socket to proxy
    except:
        # ...
```

AuthenticationManager Update

```
def __init__(
        self, app, authenticators=[], self_auths=['/api/v0/secrets']):
    """
    Initializes a new AuthenticationManager instance.

    :param app: A WSGI app to wrap.
    :type app: instance
    :param authenticators: Configured Authenticator instances to utilize.
    :type authenticators: list
    :param self_auths: List of endpoints which have their own authentication
    :type self_auths: list
    """
    self._app = app
    self.authenticators = authenticators
    self.self_auths = self_auths

def __call__(self, environ, start_response):
    """
    ...
    """
    # If the endpoint self authenticates then pass directly
    # to the handler
    if environ['PATH'] in self.self_auths:
        return self._app(environ, start_response)
    # ...
```

Example Configuration

StorageService

```
{
    "custodia_api_id": "storage_service",
    "custodia_api_key": "$API_KEY",
    "storage_handlers": [
      {
```

```
        "name": "etcd",
        "server_url": "http://127.0.0.1:2379",
        "models": ["*"]
      }
    ],
    "debug": false
}
```

Custodia

```
[DEFAULT]
libdir = /var/lib/commissaire/custodia/
logdir = /var/log/commissaire/
rundir = /var/run

[global]
debug = false
server_socket = ${rundir}/custodia.sock
auditlog = ${logdir}/custodia-audit.log

[store:etcd]
etcd_server = {{ etcd_server }}
etcd_port = {{ etcd_port }}
handler = EtcdStore
namespace = custodia_commissaire_data

[store:encrypted_etcd]
handler = EncryptedOverlay
backing_store = etcd
master_key = ${libdir}/secrets.key
master_enctype = A256CBC-HS512
autogen_master_key = true

[auth:creds]
handler = SimpleAuthKeys
id_header = CUSTODIA_AUTH_ID
key_header = CUSTODIA_KEY_ID
store = etcd
store_namespace = custodia_commissaire_api

[authz:paths]
handler = SimplePathAuthz
paths = /. /secrets

[/]
handler = Root

[/secrets]
handler = Secrets
store = encrypted_etcd
```

Documentation Updates

Documentation would need to be updated to clarify the following:

- Sensitive data is stored encrypted

- How to access the secrets store
- The bus component will need to be considered secure
- Some bus backends will need to use stunnel (and include an example)
- Information pointing to Custodia

Migration Tool

A migration tool to push secrets into the secrets store.

Future Considerations

- Commissaire could use Custodia for authentication/authorization
- Commissaire could provide a backend for Custodia to use it as authentication

Checklist

- breaks API backward compatibility
- breaks user interaction backward compatibility
- **requires new or replaces current libraries**

User Story

In order to increase security I would like encryption to be added to secrets storage so that those with access to the data do not get direct access to sensitive data.

Acceptance Criteria

- Verify a card for installing Custodia is created
- Verify a card is created for adding/updating models and updating model usage
- Verify a card is created for updating commissaire-service
- Verify a card is created for updating commissaire-http
- Verify a card is created for allowing commissaire-storage-service to query for the http endpoint

References

- Kubernetes Secrets
- Custodia

CPD-61: Host Abstraction

Metadata

- CPD Version: 1
- Status: Accepted

Description

Today the `Host` abstraction is a simple model. It is saved and retrieved via persistent storage with the `StorageService`. This CPD makes the `Host` abstraction into a model which may have its data pulled from another system (EX: CloudForms).

Rationale

Most environments Commissaire will be used in will not be greenfield. They will likely have at least one other system which is storing host information. Even if the environment is greenfield there is a good chance that other systems which require storing host data themselves will be brought in.

Design

Model Changes

The current `Host` model has the following fields:

Name	Description
address	IP address or hostname of the `Host`
status	Status of the host. Decided within Commissaire itself.
os	The Operating System the `Host` utlizes.
cpus	Number of CPU's in the `Host`
memory	Amount of memory the `Host` has at it's disposal.
space	Amount of storage the `Host` has at it's disposal.
last_check	The last time the host was checked by `Watcher`. Set by Commissaire itself.
ssh_priv_key	The ssh private key to use for accessing the `Host`.
remote_user	The username to use with the `ssh_priv_key` when accessing the `Host`.

To accommodate the possibility of external `Host` instances the following field would be added.

Name	Description
source	Name of the external system of record that can authoritatively provide `Host` information.

The use of `source` will determine if the `Host` instance should be populated from the general store defined by commissaire, or a specific store.

- When `source` is not defined, the `Host` record is considered native to Commissaire.
- If the `source` is defined it should be the name of the StoreHandler which can provide

the `Host` information. For instance, if `commissaire.storage.cloudforms` should be used, then the `source` would be `cloudforms`.

Example Host Model Owned by Commissaire

```
{
    "space": 51475068,
    "status": "active",
    "address": "192.168.155.150",
    "os": "fedora",
    "memory": 2048,
    "cpus": 4,
    "last_check": "2016-11-28T22:10:11.851787",
    "source": ""
}
```

Example Host Model Owned by An External Provider

Note: All data but `status`, `last_check`, and `source` would come from the source.

```
{
    "space": 51475068,
    "status": "active",
    "address": "192.168.155.150",
    "os": "fedora",
    "memory": 2048,
    "cpus": 4,
    "last_check": "2016-11-28T22:10:11.851787",
    "source": "cloudforms"
}
```

Accessing Hosts

`StorageService` will still be the authoritative service for retrieving `Host` data. For `StorageService` to be able to make these external calls a `StoreHandler` would need to be available and configured for any `source` in use. As an example of a `StoreHandler` see the etcd StoreHandler.

Changes to StorageService

`StorageService` currently only allows one `StoreHandler` to be configured per model (See this code chunk). This restriction would need to be changed so that multiple `StoreHandlers` can be configured with a model. The first `StoreHandler` linked to a model should be consider that models default.

The `StoreHandler` precedence would work as follows:

- If the model has an `source` then the provided `source` is used
- If the model has no `source` then the default `StoreHandler` for said model is used.

The `StoreHandler` would also need to be extended in a way to denote a difference between a traditional `StoreHandler` and an `source` `StoreHandler`. This exercise is left up to the implementer.

Example StorageService Configuration

Note: In this example `etcd` is the default for all models.

```
{
  "storage_handlers": [{
      "name": "commissaire.storage.etcd",
      "server_url": "http://127.0.0.1:2379",
      "models": ["*"],
  }, {
      "name": "commissaire.storage.cloudforms",
      "server_url": "https://example.org/api/",
      "models": ["Host"],
      "username": "commissaire_service_account",
      "password": "abetteronethanthis",
      "version": "2.0.0"
  }]
}
```

Future Considerations

When a `Host` uses an external provider we may be able to remove the load from the `Watcher` and have the provider let us know upon major status change.

The `cloud-init` script and bootstrapping will probably benefit by adding a new optional field which defines `source`.

The `Host` creation endpoints will probably benefit by adding a new optional field which defines `source`.

The `Watcher`, or another long running service, could be extended to periodically pull `Host` information from all configured `source` `StoreHandler` instances.

An `ExternalProviderService` may make sense in the future if remote control ends up being a need.

The same patterns could be used with `Cluster`.

Checklist

- **breaks API backward compatibility**
- breaks user interaction backward compatibility
- requires new or replaces current libraries

User Story

In support of allowing other systems to provide host data in a brownfield environment I would like `Host` to be abstracted in such a way that it may be from N number of horizontal systems so that I do not have to have multiple copies of host inventories.

Acceptance Criteria

- Verify that a design document is created

- Verify the document is reviewed by at least one other developer
- Verify implementation card(s) are created

References

- etcd StoreHandler
- CloudForms
- Greenfield
- Brownfield

CHAPTER 14

Indices and tables

- genindex
- modindex
- search