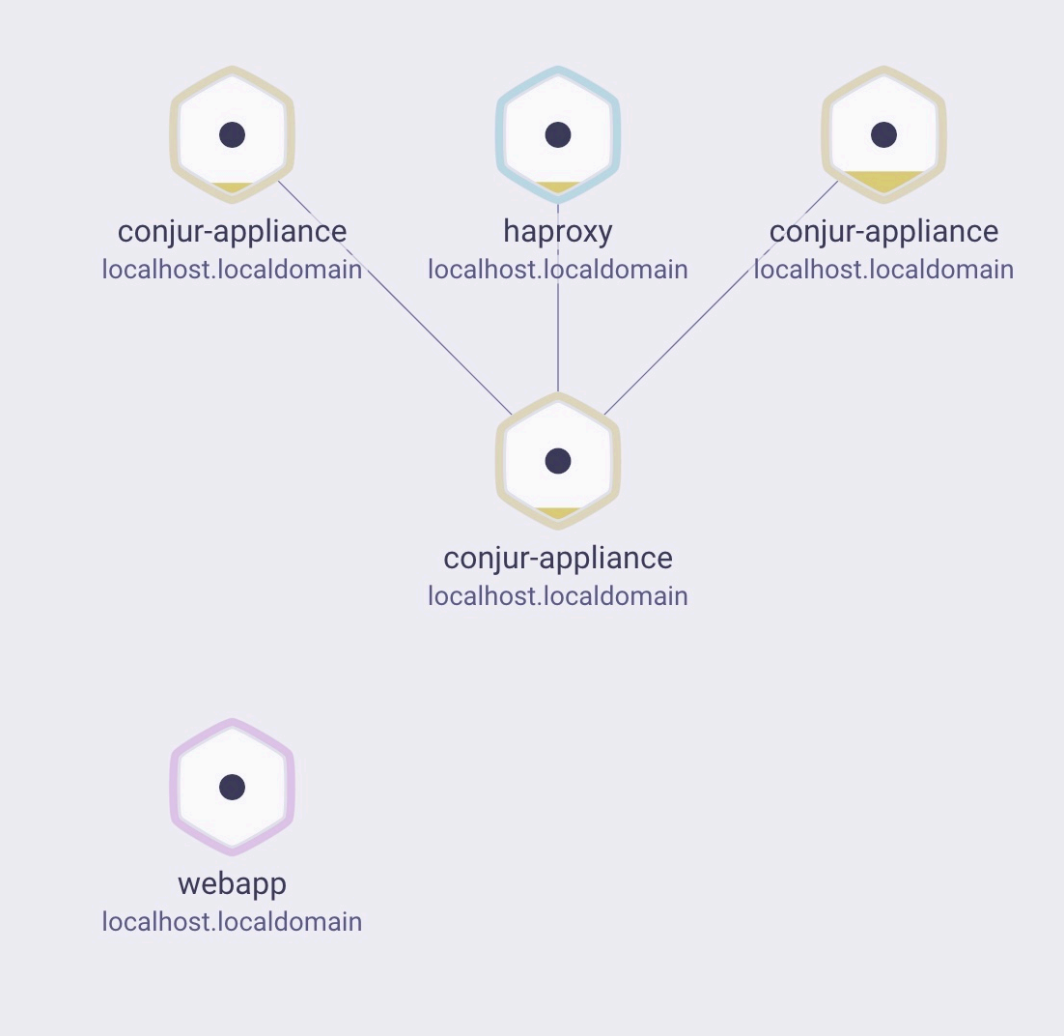
Deploying Conjur Cluster on OpenShift 3.3 (Origin 1.3)

This document is part of a repository that provides example code and instructions for deploying and configuring a Conjur cluster in OpenShift.
There are additional instructions for running a webapp for demonstation purposes.
The proposed architecture contains a master and two standbys.



0. Set up OpenShift Origin 1.3

We use a vagrant box since minishift does not support Origin 1.3.
Run the follow to set up the environment.

```
vagrant up --provision
vagrant ssh # then change directory to ./scripts
```

If not using Vagrant please modify utils with relevant credentials for login into OpenShift

1. Deployment

If using scripts please ensure `conjur-appliance:4.9-stable` is available in your Docker engine

Deploy a Conjur cluster in OpenShift can be broken down into the following steps:

- Ensure you're logged in `oc login $CLUSTER_URL -u admin -p admin`
- 0_init.sh - Setup the OpenShift environment and create the `conjur` project.
- 1_build_all.sh - Build required images `conjur-appliance:local` and `haproxy`.
- 2_start_cluster.sh - Run a master and 2 standbys, plus a `conjur-master` service which uses HAProxy.

1.1 Setup OpenShift environment

Please consult `0_init.sh`.

1.1.1 Context creation

Isolate the Conjur cluster by creating a dedicated OpenShift project for it.

1.1.2 Openshift Permissions

Appropriate privileges should be granted to ensure relevant operations can be carried out e.g. Conjur seed files can be unpacked.

Below are some privilege considerations:

In order to unpack Conjur seed files processes in the Conjur container need to run as root. Addition of the anyuid privilege grant is one way in which this could be achieved.

```
oc adm policy add-scc-to-user anyuid -z default
```

HAproxy needs to be able to list master/standby pods to update its config.

```
oc adm policy add-cluster-role-to-user cluster-reader
system:serviceaccount:$CONJUR_CONTEXT:default
```

User "developer" needs the edit role on a project.

```
oc policy add-role-to-user edit developer
```

Visit your Appliance URL

1.2 Build container images

Please consult `1_build_all.sh`.

This section assumes you have the appliance image `conjur-appliance:4.9-stable` in your OpenShift Docker Engine.

- Build the appliance, this adds `./etc/conjur.son` to the appliance. This file is used to specify the amount of memory to allocate to postgres. Consult `./build/conjur_server`
- Build HAproxy. Consult `./build/haproxy`

1.3 Deploy the Conjur cluster

Please consult `2_start_cluster.sh`.
The following steps should be carried out within the Conjur project

1.3.1 Create Conjur Cluster

Please consult `./conjur-service/conjur-cluster.yaml`.

- Create Conjur cluster from manifest
  `oc create -f ./conjur-service/conjur-cluster.yaml`

1.3.2 Configure Conjur Cluster

- Get list of the master/standby candidates and select first pod to be master,
  `$(oc get pods -l app=conjur-node --no-headers | awk '{ print $1 }')`
- Label chosen pod with role=master,
  `oc label --overwrite pod $MASTER_POD_NAME role=master`
- Configure Conjur master using evoke

```
oc exec $MASTER_POD_NAME -- evoke configure master \
 -j /etc/conjur.json \
 -h $CONJUR_MASTER_DNS_NAME \
 --master-altnames conjur-master \
 --follower-altnames conjur-follower \
 -p $CONJUR_ADMIN_PASSWORD \
 $CONJUR_CLUSTER_ACCOUNT
```

- Prepare standby seed files by running the following on the master pod
  `evoke seed standby > standby-seed.tar`
- Loop through standby candidates
  `$(oc get pods -l role=unset --no-headers | awk '{ print $1 }')`
  - Label each pod with standby role
    `oc label --overwrite pod $pod_name role=standby`
  - Copy standby seed file to pod
  - Unpack seed file and configure standby

    ```
    oc exec $pod_name evoke unpack seed /tmp/standby-seed.tar
    oc exec $pod_name -- evoke configure standby -j /etc/conjur.json -i
    $MASTER_POD_IP
    ```

- Start synchronous replication by running the following on the master pod
  `evoke replication sync`
- If weave scope is running, visit the URL and observe the Conjur master connected to 2 standbys.

1.3.3 Create and start HAProxy

- Create HAproxy from manifest
  `oc create -f ./conjur-service/haproxy-conjur-master.yaml`
- Update HAproxy by creating new config to reflect running Conjur cluster and restart daemon. Consult `./etc/update_haproxy.sh` for a working example

2. Example webapp to verify it all works!

This section demonstrates an example app consuming the the Conjur cluster running on OpenShift for the purposes of machine identity and secrets retrieval.

2.1 Configure and login the local command-line interface.

Please consult `./0_webapp_init.sh`

- Initilize host Conjur CLI for Conjur cluster on OpenShift
- Build app. Consult `./webapp_demo/build/build.sh`

2.2 Load policies and set secret values

Please consult `./1_load_policies.sh` and `webapp_demo/policy`

- Load the users policies, which would be managed by the Ops team.
- Load the OpenShift app policies, which are performed by the OpenShift admin.
- Load the database policies, which would be managed by a DBA team.
- Load the application policies, which would be managed by an application team.

2.3 Deploy webapp

Please consult `./2_deploy.sh`

- Grab the Conjur SSL cert by running the following on the Conjur master pod `cat /opt/conjur/etc/ssl/conjur.pem`. Store this in a ConfigMap.
- Rotate the webapp host API key `conjur host rotate_api_key –h $host_id` and store it in a Secret.
- Deploy the webapp application by running `oc create –f ./conjur-service/conjur-cluster.yaml`
- Visit the OpenShift console, navigate to the webapp project and observe in the logs that secrets are being fetched from the Conjur cluster.