DCAE Controller Development Guide

Note these instruction refers to current master branch. The release-1.0.0 branch have development issues that have only been fixed in master branch.

And this does NOT refer to DCAEGEN2 which have a completely new setup/process and is getting released in the R1/Amsterdam Release.

Refer to DCAE MOD User Guide for latest

DCAE Controller Overview

Make sure your settings.xml includes the onap public repo for some snapshot jar downloads during the build.

DCAE Controller LF GIT repositories

1. dcae/controller
2. dcae/operation/utils
3. dcae/controller/analytics
4. dcae/demo

SOMF - Sirius Operational Management Framework

SOMF is a model-driven framework for building controllers. SOMF Is using the Eclipse Modeling Framework (EMF) for data modeling.

LF SOMF repositories

1. ncomp/maven - contains a few maven related artifacts
2. ncomp/utils - contains Java utilities and libraries that are used across DCAE Controller projects
3. ncomp/core - contains core models
4. ncomp/sirius/manager - contains the core SOMF implementation
5. ncomp/docker - contains SOMF adaptor used to communicate with Docker Engine API
6. ncomp/openstack- contains SOMF adaptor used to communicate with OpenStack API
7. ncomp/cdap- contains SOMF adaptor used to communicate with CDAP API+

Development Setup

GIT repositories and build order

1. ncomp/maven
2. ncomp/utils
3. dcae/operation/utils
4. ncomp/core
5. ncomp/sirius/manager
6. ncomp/cdap
7. ncomp/docker
8. ncomp/openstack
9. dcae/controller
10. dcae/controller/analytics

11. dcae/demo

Eclipse Setup

Since DCAE Controller/SOMF is build on top of EMF development is tightly tied to using Eclipse IDE with specific plugin installed. This section describe the process to setup a development Eclipse.

1. Install Neon.3 Java Developer Version http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/neon3
2. Install software from Neon update site
    a. Eclipse Plug-in Development Environment 3.12.2.v20161124-1400
    b. EMF - Eclipse Modeling Framework Xcore SDK 1.4.0.v20160526-0606
3. Install software from Eclipse Market Place
    a. YEdit Feature 1.0.20.201509041456-RELEASE
4. Install Groovy from http://dist.springsource.org/snapshot/GRECLIPSE/e4.6/
    a. Groovy-Eclipse Feature 2.9.2.xx-201703131833-e46
    b. Groovy-Eclipse M2E integration 2.9.2.xx-201703131833-e46
    c. Groovy Compiler 2.4 Feature 2.9.2.xx-201703131833-e46

Import Eclipse Projects

The GIT repositories include the Eclipse Project Metadata (e.g., .project, .classpath, and etc files) and thus can be importing into a workspace using the "Import Projects" "Import existing Eclipse projects". Only the projects that need to be worked on need to be in the workspace.

Working on DCAE Controller projects

There are 2 projects for most features.

1. A model project which contains the data model using XCORE file. These projects always have a name (and artifactId) that ends in `-model`.This file is located on the `src/main/xcore` directory. And the project is setup so Eclipse will update automatically the generated Java code in the `src/main/xcore-gen` directory. This project is a straight EMF type project and contain very little SOMF specifics.
2. A SOMF project which contains SOMF generated code and implementation of the APIs that the model defines.
    a. SOMF generator class `Generator.java` under the `src/main/java` folder. It contains the defined customization for creating the SOMF implementation. Running this class will generate new contents in the two generated folders.
    b. `src/main/sirius-gen` folder contains generated Java code for implementing Client and Server side functions.
    c. `src/main/server-gen` folder contains generated Bash and Groovy scripts for implementing Client and Server side functions.
    d. SOMF provider `*Provider.java` Java classes under the `src/main/java` folder. These implements the operation/methods that are defined in the model.

DCAE Configuration Setup

The DCAE configuration setup is trying to carefully handle Environmental and Non-Environmental configuration.

The ONAP Demo setup is creating another level of complexity to the setup since the overall ONAP Heat templates will deliver environmental information that need to be include in the Environmental configuration.

This is the various configuration entities.

1. DCAE Non Environmental ONAP Demo configuration. https://gerrit.onap.org/r/gitweb?p=dcae/demo.git;a=tree;f=OPENECOMP-DEMO;h=38b2f51a1be5c12f357e228b7a655c0ee97606b8;hb=HEAD with the main files
    a. location-types.yaml define the various DCAE entities (e.g., docker-host VMs, cdap cluster VMs, VES collector etc) that are part of the demo deployment
    b. vm-templates/vm-docker-host.yaml, vm-templates/vm-cdap-cluster.yaml, vm-templates/vm-postgresql.yaml . Define the deployment configuration for various VM deployments.
    c. docker-templates/docker-XX.yaml. Similar for docker deployments.
    d. cdap-templates/cdap-YY.yaml. Similar for docker deployments.
    e. steams.yaml. Define DCAE DMaaP setup.
    f. HEAT provided information. When the ONAP demo is getting deploy the vm1-dcae-controller VM which will come up with the DCAE controller Docker container. This container will use configuration attributes setup in the file /opt/app/dcae-controller/config.yaml to apply to DCAE controller environment file, which will have expressions like @{XXX} replaced with the value of XXX in the config.yaml file.

Variable Name	Sample Values	Notes
BASE	RACKSPACE,2-NIC, 1-NIC-FLOATING-IPS	RACKSPACE will only work in Rackspace, 2-NIC can work in both Rackspace and non Rackspace environment, and 1-NIC-FLOATING-IPS does not work in Rackspace.
DCAE-VERSION	1.1.0	Note only 1.1.0 version is working outside of Rackspace
OPENSTACK-AUTH-METHOD	password, api-key	Need to use password for non-Rackspace environments

DOCKER-VERSION	1.1-STAGING-latest	

2. DCAE Environmental ONAP Demo configuration. Currently 3 deployment scenarios are supported (See DCAE-7 - Getting issue details... STATUS for details about current status.)
    a. RACKSPACE. This scenario only work when the Cloud provider is Rackspace. In this each DCAE VM will be setup with a predefined fixed IP on a private network and it will get a random public IP on the public network. This scenario depends on the following values from the HEAT provided environment file: DCAE-VERSION, DOCKER-REGISTRY, DOCKER-VERSION, GIT-MR-REPO, HORIZON-URL, KEYSTONE-URL, NEXUS-PASSWORD, NEXUS-RAWURL, NEXUS-USER, OPENSTACK-KEYNAME, OPENSTACK-PASSWORD, OPENSTACK-PRIVATE-NETWORK, OPENSTACK-PUBKEY, OPENSTACK-REGION, OPENSTACK-TENANT-ID, OPENSTACK-TENANT-NAME, OPENSTACK-USER, POLICY-IP, STATE, ZONE.
    b. 2-NIC. This scenario works in most OpenStack environments and allows higher level of flexibility in assigning IPs etc. Similar to the RACKSPACE setup each DCAE VM will be setup with a predefined fixed IP on a private network and it will get a random public IP on the public network.This scenario depends on the following values from the HEAT provided environment file: BASE=2-NIC, DCAE-VERSION, DNS-IP-ADDR, DOCKER-REGISTRY, DOCKER-VERSION, FLAVOR-LARGE, GIT-MR-REPO, HORIZON-URL, KEYSTONE-URL, NEXUS-PASSWORD, NEXUS-RAWURL, NEXUS-USER, OPENSTACK-AUTH-METHOD, OPENSTACK-KEYNAME, OPENSTACK-PASSWORD, OPENSTACK-PRIVATE-NETWORK, OPENSTACK-AUTH-METHOD, OPENSTACK-PUBKEY, OPENSTACK-REGION, OPENSTACK-TENANT-ID, OPENSTACK-TENANT-NAME, OPENSTACK-USER, POLICY-IP, STATE, UBUNTU-1404-IMAGE, UBUNTU-1604-IMAGE, ZONE, dcae_cdap00_ip_addr, dcae_cdap01_ip_addr, dcae_cdap02_ip_addr, dcae_coll00_ip_addr, dcae_ip_addr, dcae_pstg00_ip_addr, public_net_id. The HEAT template demo/heat/OpenECOMP/onap_openstack_nofloat.yaml provides these variables.
    c. 1-NIC-FLOATING-IPS. This scenario works in most OpenStack environments and allows higher level of flexibility in assigning IPs etc. In this setup each DCAE VM will only have one NIC with an IP on the private network. In addition each VM will have a floating IP from the public network. In this case it is the floating VMs that can get predefined values and the IPs from the private network that are assigned to the NIC are randomly setup. This makes a few issues with the rest of the demo setup but should be working in the 1.1 release. This scenario depends on the following values from the HEAT provided environment file:BSAE=1-NIC-FLOATING-IPS, DCAE-VERSION, DNS-IP-ADDR, DOCKER-REGISTRY, DOCKER-VERSION, FLAVOR-LARGE, GIT-MR-REPO, HORIZON-URL, KEYSTONE-URL, NEXUS-PASSWORD, NEXUS-RAWURL, NEXUS-USER, OPENSTACK-AUTH-METHOD, OPENSTACK-KEYNAME, OPENSTACK-PASSWORD, OPENSTACK-PRIVATE-NETWORK, OPENSTACK-PUBKEY, OPENSTACK-REGION, OPENSTACK-TENANT-ID, OPENSTACK-TENANT-NAME, OPENSTACK-USER, POLICY-IP, STATE, UBUNTU-1404-IMAGE, UBUNTU-1604-IMAGE, ZONE, dcae_cdap00_float_ip_addr, dcae_cdap01_float_ip_addr, dcae_cdap02_float_ip_addr, dcae_coll00_float_ip_addr, dcae_float_ip_addr, dcae_pstg00_float_ip_addr. The HEAT template demo/heat/OpenECOMP/onap_openstack_float.yaml provides these variables. One requirement for this setup is that the floating IPs that is getting used are already associated with the Openstack Tenant used.

The overall deployment flow is the following.

1. HEAT template
    a. Create the vm1-dcae-controller VM.
    b. Setup /opt/app/dcae-controller/config.yaml
    c. Start the Docker Container for the DCAE Controller
2. The DCAE controller will (running /opt/app/dcae-controller-platform-server/bin/controller-startup.sh)
    a. Determine the BASE based on the BASE attribute in config.yaml
    b. Determine the ZONE based on the ZONE attribute in config.yaml
    c. Substitute the HEAT delivered attributes from config.yaml into /opt/app/dcae-controller-platform-server/OPENECOMP-DEMO-$BASE and produce a /opt/app/dcae-controller-platform-server//opt/app/dcae-controller-platform-server/OPENECOMP-DEMO-$ZONE.
        i. `bin/dcae-controller.sh rackspace-substitute --from OPENECOMP-DEMO-$BASE --to OPENECOMP-DEMO-$ZONE --file /opt/app/dcae-controller/config.yaml`
    d. Merge the Non-environmental configuration OPENECOMP-DEMO with the environmental configuration OPENECOMP-DEMO-$ZONE to create the complete configuration setup for this specific environment which will be placed in GITLINK/OPENECOMP-DEMO-$ZONE
        i. `java -cp 'lib/*' org.openecomp.dcae.controller.operation.utils.GenControllerConfiguration $ZONE . GITLINK OPENECOMP-DEMO`
    e. The DCAE Controller will be started up and synced with the configuration
        i. `bin/dcae-controller.sh start`
        ii. bin/dcae-controller.sh sync-configuration --environment OPENECOMP-DEMO-$ZONE
    f. The DCAE Controller can then Deploy the various DCAE components define in the demo environment
        i. `bin/dcae-controller.sh deploy-service-instance -i $ZONE -s vm-docker-host-1`
        ii. `bin/dcae-controller.sh deploy-service-instance -i $ZONE -s vm-postgresql`
        iii. `bin/dcae-controller.sh deploy-service-instance -i $ZONE -s vm-cdap-cluster`
        iv. `bin/dcae-controller.sh deploy-service-instance -i $ZONE -s docker-databus-controller`
        v. `bin/dcae-controller.sh deploy-service-instance -i $ZONE -s cdap-helloworld`
        vi. `bin/dcae-controller.sh deploy-service-instance -i $ZONE -s cdap-tca-hi-lo`
        vii. `bin/dcae-controller.sh deploy-service-instance -i $ZONE -s docker-common-event`

Common Deployment Issues Seen

{"auth":{"RAX-KSKEY:apiKeyCredentials":{"username":"<some-user-here>","apiKey":"<some-key-here>"}}}

Using Rackspace authentication in non Rackspace. Make sure your are using 1.1 DCAE controller container and that OPENSTACK-AUTH-METHOD = 'password', DOCKER-VERSION =1.1-STAGING-latest and DCAE-VERSION = 1.1.0

java.lang.IllegalArgumentException: !Absolute URI: null/servers

Connectivity to OpenStack APIs is not working look in logs/*.err for the error returns by the keystone API call.

Reporting Issues

Due to the large number of possible issues in a deployment please run the following script and include the output in the Ticket.

```
#!/bin/bash

PW=$(grep OPENSTACK-PASSWORD /opt/app/dcae-controller/config.yaml | sed s/OPENSTACK-PASSWORD:.//)

set -e

echo ======= config.yaml

cat /opt/app/dcae-controller/config.yaml | sed "s/$PW/XXXXXX/"

echo ======= docker

docker images

docker ps -a

ID=$(docker ps | grep dcae-controller: | cut -c1-12)

echo ======= docker logs

docker logs $ID 2>&1

echo ======= dcae-controller.sh.log

docker exec $ID cat /opt/app/dcae-controller-platform-server/logs/dcae-controller.sh.log

echo ======= reports

docker exec -e GROOVY_HOME=/opt/app/groovy-2.4.6 $ID /opt/app/dcae-controller-platform-server/bin/dcae-controller.
sh report -n /reports/dcae/vms

docker exec -e GROOVY_HOME=/opt/app/groovy-2.4.6 $ID /opt/app/dcae-controller-platform-server/bin/dcae-controller.
sh report -n /reports/dcae/service-instances

echo ======= logs err

docker exec $ID cat /opt/app/dcae-controller-platform-server/logs/controller-platform-server-controller.err |
head -10000 | sed "s/$PW/XXXXXX/"

echo ======= logs out

docker exec $ID cat /opt/app/dcae-controller-platform-server/logs/controller-platform-server-controller.out |
head -10000
```