OpenECOMP SDC

Installing on Rackspace

Product:	OpenECOMP SDC
Title:	Installing on Rackspace
Product version:	v1.0
Publication date:	01/30/2017
Reference number:	SDCOSIG021317
Revision number	01

Disclaimer

Contents

1 Overview

SDC is deployed using a HEAT template that bootstraps a VM machine on the Rackspace cloud. Assuming that the deployment runs as expected then the following System setup, prerequisites and requirements should automatically be put in place.

This document describes the manual processes required to perform any one of the automated steps, if the deployment did not run to completion as expected.

The document deals with several key topics:

- System setup and prerequisites
- Manual system setup
- SDC application setup
- SDCconsumers setup

2 System setup and prerequisites

Operating system requirements

- Ubuntu v16.04
- Volume allocated and named /data
- Docker service installed, up and running
- Docker is connected to the Docker hub repository

SDC application requirements

- <environment>.json located under:

  /data/ASDC/environments

- docker_health.sh script located under:

  /data/scripts

- docker_run.sh script located under:

  /data/scripts

3 Manual system setup

The manual system setup consists of :

- Installing the Docker service
- Updating Ubuntu to the latest version available on the Rackspace repository
- Allocating volume for /data
- Setting the connection to the Docker hub repository

Installing the Docker service

1. Start the Rackspace machine
   A. Login to Rackspace console
   B. Start the Rackspace VM
   C. Login to the Rackspace server with the credentials given by Rackspace
2. Check Ubuntu uname:
   `uname -r`
   The output needs to be grater then 3.13.0-91-generic

   For example:
   ```
   root@vm1-asdc:/data# uname -r
   4.4.0-51-generic
   ```
3. Install the Docker service:
   A. Login to the new machine as root and run:

   ```
   apt-get update
   apt-get install apt-transport-https ca-certificates (Press Y)
   apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
   echo "deb https://apt.dockerproject.org/repo ubuntu-trusty main" | tee /etc/apt/sources.list.d/docker.list
   apt-get update
   apt-cache policy docker-engine
   apt-get -y install linux-image-extra-$(uname -r) linux-image-extra-virtual
   apt-get -y install docker-engine
   service docker start
   ```

   B. Verify that Java is installed:

   ```
   java –version
   ```

   C. If Java is not installed then run:

```
apt-get install default-jre
apt-get update
```

4. A system restart is required; reboot the server.

Updating Ubuntu to the latest version available on the Rackspace repository

1. Login to the Rackspace server
   The output for a successful login will be displayed; check the version, for example:
   ```
   Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-51-generic x86_64)
   ```
2. Upgrade UBUNTU (only if version is below 16.04.1 LTS):
   - Run: **do-release-upgrade** (press y, press enter twice, y, select “install the package maintainer's version”, y, y - for reboot)
   - **Note**: this will update Ubuntu from 14.04.1 LTS to 16.04.1 LTS

Allocating a volume for data

1. vi /etc/fstab, add this line:
   ```
   /dev/xvdb1 /data ext4 errors=remount-ro,noatime,barrier=0 0 1
   ```
2. Check that the volume is mounted:
   ```
   df -h | grep data
   ```
3. The expected result should be in the form:
   ```
   /dev/xvdb1 99G 528M 93G 1% /data
   ```

Set connection to Docker hub repository

This section deals with connecting to the OpenECOMP SDC Docker repository and checking that the Docker Service is up and running.

To set the connection to the Docker hub repository:

1. add to /etc/hosts "`10.208.197.75 ecomp-nexus`"
2. Connect to OpenEcomp SDC docker-repo (only if needed):
   - keytool -printcert -sslserver ecomp-nexus:8443 -rfc > nexus.crt
   - cp nexus.crt /usr/local/share/ca-certificates/
   - update-ca-certificates

- service docker restart
    - docker stop/waiting
    - docker start/running, process 6518

3. Check that the Docker service is up and running by running the command:

    - service docker status

4 SDC application setup

This section runs a script to start the Dockers and boot the system. The script takes the latest Dockers from the Docker repository. The boot process takes approximately 1 minute.

Starting the SDC application

The SDC application is executed using several Dockers

To start the dockers:

1. Run the script in the format usage as described:
   Usage: /data/scripts/docker_run.sh -e <environment name> -r <release> -p <docker-hub-port>
   Note: the use of spaces and dashes in the script format is mandatory.
   Example usage as for the Orange environment:

   ```
   /data/scripts/docker_run.sh -e Orange -r 1610.2.12 -p 51220
   ```

2. A Health check to verify that OpenECOMP SDC is up and running is conducted as part of the docker_run.sh script. The expected output would look similar to that provided:

Health Check output:

```
root@vm1-asdc:~# /data/scripts/docker_health.sh
{
   "cluster_name" : "ASDC-ES-OS-ETE-DFW",
   "status" : "green",
   "timed_out" : false,
   "number_of_nodes" : 1,
   "number_of_data_nodes" : 1,
   "active_primary_shards" : 1,
   "active_shards" : 1,
   "relocating_shards" : 0,
   "initializing_shards" : 0,
   "unassigned_shards" : 0,
   "delayed_unassigned_shards" : 0,
   "number_of_pending_tasks" : 0,
   "number_of_in_flight_fetch" : 0,
   "task_max_waiting_in_queue_millis" : 0,
   "active_shards_percent_as_number" : 100.0
}
```

BE health-Check:


```
{
   "sdcVersion": "1610.2.12",
   "siteMode": "unknown",
   "componentsInfo": [
       {
          "healthCheckComponent": "BE",
          "healthCheckStatus": "UP",
          "version": "1610.2.12",
          "description": "OK"
       },
       {
          "healthCheckComponent": "ES",
          "healthCheckStatus": "UP",
          "description": "OK"
       },
       {
          "healthCheckComponent": "TITAN",
          "healthCheckStatus": "UP",
          "description": "OK"
       },
       {
          "healthCheckComponent": "DE",
          "healthCheckStatus": "UP",
          "description": "OK"
       }
   ]
}
```


FE health-Check:


```
{
   "sdcVersion": "1610.2.12",
   "siteMode": "unknown",
   "componentsInfo": [
       {
          "healthCheckComponent": "BE",
          "healthCheckStatus": "UP",
          "version": "1610.2.12",
```

```
            "description": "OK"
        },
        {
"healthCheckComponent": "ES",
            "healthCheckStatus": "UP",
            "description": "OK"
        },
        {
            "healthCheckComponent": "TITAN",
            "healthCheckStatus": "UP",
            "description": "OK"
        },
        {
            "healthCheckComponent": "DE",
            "healthCheckStatus": "UP",
            "description": "OK"
        },
        {
            "healthCheckComponent": "FE",
            "healthCheckStatus": "UP",
            "version": "1610.2.12",
            "description": "OK"
        }
    ]
}
[{"firstName":"Carlos","lastName":"Santana","userId":"cs0008","email":"de
signer@sdc.com","role":"DESIGNER","lastLoginTime":1483900163928,"status":
"ACTIVE","fullName":"Carlos Santana"}]
```

5 SDC consumers setup

This section describes the process for creating an SDC consumer that allows OpenECOMP components to communicate with SDC.

Creating SDC consumers

In order to integrate with other OpenECOMP components, consumers are configured per component. This operation is performed as an integral part of executing the Backend Docker.

The following users are predefined:

- appc
- vid
- dcae
- aai
- sdnc
- mso

For the purposes of this document the manual consumer configuration process is described.

To create new SDC consumers:

1. Login to the SDC Rackspace VM
2. Copy the file security-utils-1610.2.1.jar to the VM under /tmp
3. Run the following commands from the Docker, providing consumer specific values for the highlighted parameters:

```
Consumers creation, Run From Docker (Change the marked in yellow):
```

```
consumerName=<Consumer user name> (For example: appc )
```

```
user_pass=<Consumer password> (For example: appcos )
```

```
IP=localhost <OR Docker IP>
```

```
enc_pass=`java -cp /tmp/security-utils-*.jar
org.openecomp.sdc.security.Passwords $user_pass |tr '[]' ' '|awk '
{print $1}'`
```

```
salt=`echo $enc_pass |awk -F: '{print $1}'
```

```
pass=`echo $enc_pass |awk -F: '{print $2}'
```

```
curl -X POST -i -H "Accept: application/json; charset=UTF-8" -H
"Content-Type: application/json" -H "HTTP_CSP_ATTUID: jh0003"
http://$IP:8080/sdc2/rest/v1/consumers/ -d '{"consumerName":
'$consumerName', "consumerSalt": '$salt',"consumerPassword":
'$pass'}'
```

Note: Repeat the consumer creation process for each consumer, providing Consumer specific consumerName and user_pass

For example:

```
consumerName=appc

user_pass=appcos

enc_pass=`java -cp /tmp/security-utils-1702.0.11.jar
org.openecomp.sdc.security.Passwords $user_pass |tr '[]' ' '|awk '
{print $1}'`

salt=`echo $enc_pass |awk -F: '{print $1}'`

pass=`echo $enc_pass |awk -F: '{print $2}'`

curl -X POST -i -H "Accept: application/json; charset=UTF-8" -H
"Content-Type: application/json" -H "HTTP_CSP_ATTUID: jh0003"
http://localhost:8080/sdc2/rest/v1/consumers/ -d '{"consumerName":
'$consumerName', "consumerSalt": '$salt',"consumerPassword":
'$pass'}'
```

4. Check that the consumer was successfully created in SDC:

```
curl -X GET -i -H "Accept: application/json; charset=UTF-8" -H
"Content-Type: application/json" -H "HTTP_CSP_ATTUID: jh0003"
http://localhost:8080/sdc2/rest/v1/consumers/<consumerName>

HTTP/1.1 200 OK

Set-Cookie: JSESSIONID=1ahpyqpjjgfblahos4f03qun9;Path=/

Expires: Thu, 01 Jan 1970 00:00:00 GMT

Content-Type: application/json;charset=UTF-8

X-ECOMP-RequestID: 6e47cbde-44e8-4b82-8f17-c6a731bf0081

Vary: Accept-Encoding, User-Agent
```

```
Content-Length: 268

Server: Jetty(9.3.12.v20160915)
```

{"consumerName":"vid","consumerPassword":"3936abc03d50693c90ec68a4a60427d6bdde8b085d60314333c9e58a270ff6f3","consumerSalt":"eaa62d9681d8f803ac05db342e3c9cc0","consumerLastAuthenticationTime":0,"consumerDetailsLastupdatedtime":1481211500749,"lastModfierAtuid":"jh0003"}