Automated deployment and testing - using SMO package and ONAP Python SDK

Table of Contents

Subpages

Introduction

This wiki is meant to detail the approach and steps taken to automate the deployment and testing of the O-RAN-SC SMO package based on ONAP including validating sample use cases.

Driver

Automation is key nowadays and deployment and testing needs to be repeatable and portable.

In order to achieve a certain level of automation with the O-RAN-SC software we took the following steps :

1. Create a simple deployment method, reusing the work that was done by ONAP : see SMO package (https://jira.onap.org/browse/REQ-887)
2. Reuse existing testing automation tools that were successfully used in ONAP : see Python-SDK (https://python-onapsdk.readthedocs.io/en/master/)
3. Extend the deployment mechanics to provide a self contained, portable setup that can validate sample use cases on various type of deployment (called "flavors")

The End goal is to provide the community with a mean to deploy the SMO and its test environment with a minimum set of requirements, ultimately this setup can be used on a lab to automatically validate code changes and report issues directly in code review tools.

The setup as pictured in this wiki, is by no means closed, it can be easily extended thanks to the flexibility of all the tools that were chosen.

SMO Package based on ONAP

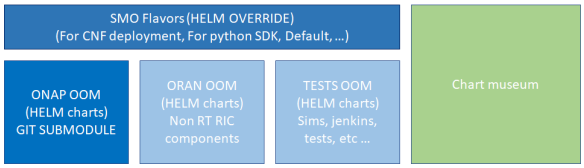
The SMO package is accessible on ORAN gerrit in the "it/dep" repo: https://gerrit.o-ran-sc.org/r/gitweb?p=it/dep.git;a=tree;f=smo-install;h=2e4539d6c3c2e2a274d1913c89df371c956f0793;hb=HEAD

It's based on the ONAP OOM repository because it is used as a git submodule. The ONAP charts are used unchanged nor redefined but instead obviously configured by using the helm override mechanism.

The ORAN charts are defined mainly for Non-RTRIC part but others charts could be added later.

The tests charts defined contains helm charts for network simulators (DU/RU/Topology server), jenkins or python SDK tests.

The chart museum is used to store the HELM charts built locally (as the ONAP and ORAN charts are currently not available remotely)



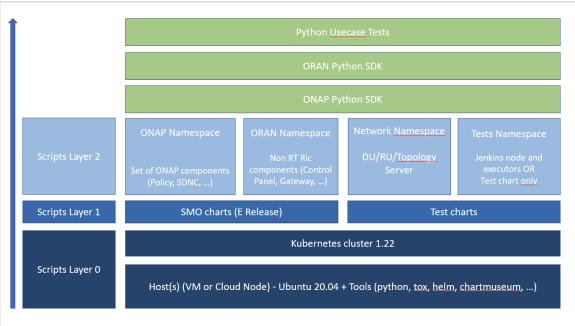
The SMO package contains some scripts to setup the node, install the smo/jenkins, start simulators, uninstall, etc ....

These scripts have been split into 3 different layers but depending of your setup, some scripts can be skipped.

- Layer 0: Setup the node (microk8s, helm, chartmuseum, tests tools, etc ...)
- Layer 1: Build the helm charts and upload them to chart museum
- Layer 2: Deploy SMO with specific flavors, deploy network simulators, deploy CI/CD tool, etc ...

Architecture

The picture below describes the environment from a high level and the various entities that are deployed with this setup



The basis is a standard VM or node running a recent Ubuntu release where a Kubernetes cluster is setup (that cluster can also be remote if needed as long as it is accessible from the host VM)

The deployment layout is explained in the readme file of the repository but in a nutshell it is composed of :

- helm charts to deploy the
  - SMO, the simulators
  - a dedicated embedded jenkins instance
  - an Helm Chart repository to store locally useful charts for the setup
- utility scripts to help setup the environments from a basic Ubuntu VM or node
- a set of Python SDK tests cases that are executed

The Jenkins instance will act as a Meta executor to orchestrate the deployment and execute the set of tests.

This layout provides all the needed configuration for the embedded Jenkins instance to interact with the cluster and connect to remote repositories

Deployment guide

Minimum Requirements

The target deployment environment is quite flexible thanks to the chart configurations and options.

The minimum setup requires :

1 VM or host with

- 6 or more CPU cores
- 20 G of RAM
- 60G of disk (mostly used to store container images
- Ubuntu 20.04 LTS (under test using 18.04 LTS)

Prerequisites

Be sure your user account has enough privileges to execute the commands that are listed below, some may imply to grant your user sudo privileges or manipulate users/groups so that these can be executed.

```
$ cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.6 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.6 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies
/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

```
$ git --version
git version 2.17.1
```

Installation

Download the IT/dep repository from gerrit at the following location :

```
$ mkdir workspace && cd workspace
$ git clone --recurse-submodules https://gerrit.o-ran-sc.org/r/it/dep.git
$ cd dep
```

Note that you need to add the recurse sub modules flag as some parts are git submodules pointing to existing related charts (ONAP)

The installation is quite straight forward, several utility scripts are available from the repository to allow the user to setup all/some of the components independently

Please have a look at the embedded readme file located under dep/smo-install, it provides a full description of the content and how to setup various flavors of the environment

https://gerrit.o-ran-sc.org/r/gitweb?p=it/dep.git;a=blob_plain;f=smo-install/README.md;hb=refs/heads/master

User guide

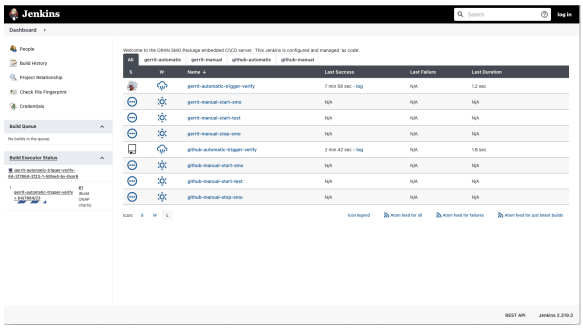
This section describes how to access and use the setup that has been deployed, it allows the user to configure, run and analyze tests results on O-RAN-SC package, it can also serve as a validation tool for incoming code changes

1) accessing

Once the system is boot strapped, a new namespace called 'test' should appear on your Kubernetes environment.

you should be able to access the jenkins instance by opening a browser to :

http://<your k8s host>:32080/



All Jobs are auto provisioned by the deployment to be able to :

- Connect to a remote repository (sample for Github and Gerrit provided)
- Auto discover branches and open pull requests / Gerrit reviews
- Download changes from the repository, build deploy and tests
- It is also possible to manually trigger Start / Stop / Test the SMO package (Job with the Manual keyword in the name)

It is possible to login to the Jenkins instance (to see/modify the configuration) by using the default login /password (test/test) - this can be overriden in overrides files (see above at installation part)

2) Running the tests

To execute the tests, there are 2 options :

1) Auto Scheduled checks

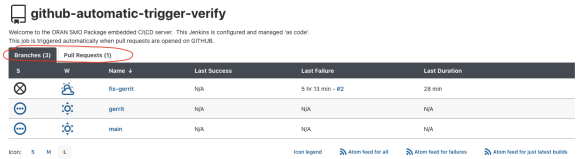
configure the access to the github/gerrit repositories and let the system scan for available branches, you can see what has been retrieved by clicking on the "-verify" jobs.

Once the access to the repositories is configured, new jobs will be automatically created to validate available branches, they can be triggered by launching the jobs

OR

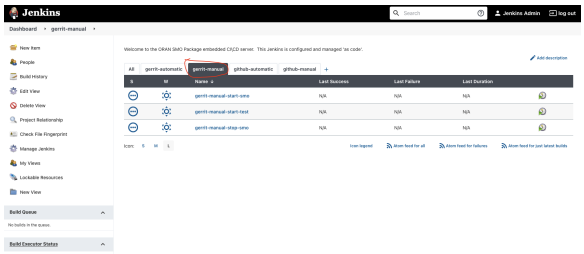
Jobs are going to be created automatically on Gerrit reviews(gerrit) / Pull Requests (github) and will check for new changes automatically, or can be launched manually

see jobs below



2) Manual startup

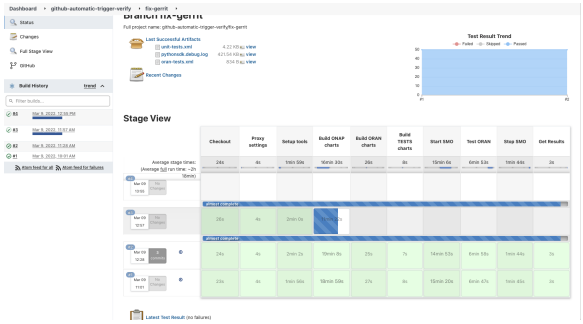
A set of jobs is automatically created to allow for a manual startup (regardless of branches or current activity on the repository), they can be accessed from the main dashboard or using the tabbed view



In order to execute the manual jobs, you need to be logged in (test user or the user defined in the override files at installation

when running a manual job, you can provide the branch which will be pulled from the repository and the location of the override files (Flavor) which defines which components and tests will be run

Once the tests are executed, you can follow the pipeline execution through the Jenkins UI



Logs/results can be accessed through the workspace available in the job execution UI, execution history, trends and step details can also be accessed through Jenkins interface

3) Understanding the tests

There are 2 sets of tests being run by the tool :

Unit tests

They are typically used to validate the ORAN SDK methods at a unit level, every time a new method is added to the ORAN SDK it must be validated by a simple unit test.

The pyTest framework is used to execute them.

Integration tests

These tests make use the ONAP/ORAN Python SDK to validate specific SMO usecases. They can initialize, provision, verify, trigger, etc ... the different ONAP/ORAN components started by the SMO package. They need to be executed within the Kubernetes cluster or directly on a Kubernetes node as the tests need access to the virtual networks created by the kubernetes cluster.

4) Python SDK tests and how to add/enhance tests

Demo Recording

Plan:

- What is the SMO package

- What is ONAP Python SDK

- What is ORAN Python SDK
- How to execute the SMO usecase tests
- SMO Nomad/ephemeral CICD jenkins

Other resources