OOM ONAP Offline Installer

2019-06-11 Stockholm ONAP DDF

Presenters:
Michal Ptacek
Mateusz Pilat
Agenda

❖ Introduction
  ➢ Motivation
  ➢ Architecture (as of 06/19)
  ➢ Build & installation procedure as of now

❖ What has changed since last DDF meeting
  ➢ Casablanca Installer fully based on Ansible
  ➢ Integration tests introduction

❖ Future plans for El-Alto and beyond
  ➢ Align closely offline installer with ONAP development
  ➢ Proposals for new features for Offline Installer
  ➢ Final goal
  ➢ Challenges, concerns

❖ Q&A
Part I. Introduction
Motivation

❖ Production deployments are not expected to be running from “online” source

❖ Need to have a control about what exactly is deployed (avoid surprises from latest code). Currently each online deployment can end-up with slightly different result.

❖ Should not be dependent on availability of all required online artifacts during installation or even runtime (e.g. nexus3.onap.org downtimes)

“We need to have a pre-populated platform, which can be used for any lab deployment in offline environments.”
Architecture of Offline Installer

❖ All artifacts* required during the deployment or even runtime must be accessible through the offline platform (i.e. pre-downloaded)

❖ Instead of patching whole ONAP and removing hardcoded URL’s, the main idea of our offline solution is to simulate all internet servers required (using own nexus accessible via nginx used as reversed proxy)

❖ Artifacts lists are stored in text files and there is manual effort in updating those data lists

* artifacts – docker images, npm packages, rpm packages, git repos, pip packages, files, ...
Simulated domains in Dublin

git: (git repos)
gerrit.onap.org
git.rancher.io
github.com

http: (maven artifacts)
git.onap.org
nexus.onap.org
repo.infra-server
www.getcloudify.org
www.springframework.org
repo.maven.apache.org
repo1.maven.org

nexus: (docker images, npm & pip)
docker.elastic.co
docker.io
index.docker.io
gcr.io
k8s.gcr.io
nexus.{{ ansible_nodename }}
nexus3.onap.org
pndareg.ctao6.net
quay.io
registry-1.docker.io
registry.hub.docker.com
registry.npmjs.org
Offline installer in Dublin – Build & Installation procedure

Packaging time

1. Download binaries
2. Populate nexus blob
3. Fetch & Patch OOM charts
4. Prepare SW package
   - Prepare Resource package
   - Prepare AUX package

Run time

5. Upload resources
6. Infra
7. RKE
8. ONAP

Alternative kubernetes
Alternative app

Not planned as of now (but can be added later)
Offline installation scheme

* during installation all artifacts are copied into infra node, resource host and install server are not needed in runtime
# Platform build procedure

<table>
<thead>
<tr>
<th>Preparations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure repos &amp; install SW for download/build/package scripts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download all artifacts based on collected data lists</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Populate nexus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build nexus blob</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Patching OOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch &amp; patch OOM helm charts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SW packaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build ansible image, create SW package, create resource package,..</td>
</tr>
</tbody>
</table>

**RHEL/Centos 7.6 build server**
Installation procedure - upload resources

- Resource host
- Ansible container or CHROOT
  - upload_resources.yml
- Infra node
  - /opt/onap/<resources>
- K8s node
- K8s node
- K8s node
Installation procedure - infrastructure deployment

- Resource host
- Ansible container or CHROOT
- Infra node
  - Dnsmasq
  - VNCserver
  - Nexus
  - Nginx
  - /opt/onap/<resources>

- K8s node
- K8s node
- K8s node

Optionally: Load additional/application artifacts into Nexus from aux package
Installation procedure - deploy k8s cluster

1. **Resource host**
2. **Ansible container or CHROOT**
3. **Infra node**
   - Dnsmasq
   - VNCserver
   - Nexus
   - Nginx
   - `/opt/onap/<resources>`
4. **rke.yml**
5. **K8s node**
   - K8s node
   - K8s node
   - K8s node
   - K8s node
   - K8s node

---

**THE LINUX FOUNDATION**

---

**ONAP**
Installation procedure - deploy onap using OOM charts

- Resource host
- Ansible container or CHROOT
- onap.yml
- Infra node
  - Dnsmasq
  - VNCserver
  - Nexus
  - Nginx
  - /opt/onap/<resources>
- K8s node
  - ONAP pods
- K8s node
  - ONAP pods
- K8s node
  - ONAP pods
Part II. What has changed since last DDF in Nozay
Installer redesign finished

➢ In Beijing we had working prototype of ansible offline installer
➢ In Casablanca complete solution was verified on top of 3.0.2 with vFWCL demo – notes published

https://gerrit.onap.org/r/gitweb?p=oom/offline-installer.git;a=tree;h=690b6dc9ec3c19cc225f6f495c8711b098f91008;hb=refs/heads/casablanca

Rebase from Casablanca to Dublin was much easier due to improved maturity of offline solution (we started testing with RC0 not after sign-off like it was in Casablanca)

New functionality:

➢ Role separation for each deployment stage (we benefit from that already in Dublin when replacing rancher with RKE)
➢ Kubernetes control plane can now be installed outside of infra node
➢ Added support for generic override file for helm charts
➢ Redesign of download scripting (more reliable, python based)
Automatic ONAP offline platform creation

- Tracked in OOM-1876 - Introduce fully automated ONAP offline platform builds in CI

Motivation “our goal in Dublin was to work more closely with community and start testing with RC0”, due to frequent changes of OOM charts building platform on top of that was like shooting moving target”

=> fully automated ONAP offline platform solution need to be adopted

Two types of inputs for offline platform:

good one’s – OOM charts, offline platform specific images & packages
bad ones’s – other artifacts required by various ONAP projects during install time & mostly runtime (we are trying to get rid of bad one’s together with community CCSDK-1117, SDNC-685, APPC-1441, DCAEGEN2-1088, POLICY-1576, SO-1917, OPTFRA-509, ... )
Automatically generated ONAP Offline Platform (Dublin)

- Get OOM charts (release specific)
- Get RKE specific docker images data list
- Get npm packages data list
- Get git repos data list
- Get rpm packages data list
- Get git repos
- Get rpm packages
- Get git repos
- Get rpm packages
- Get rpm packages
- Get git repos
- Get pip packages

Download

- Download RKE docker image
- Download application docker image
- Download npm packages
- Download git repos
- Download rpm packages
- Download http files

Bin utils are newly downloaded as http files

Build

- Build nexus blob
- Build ansible image
- Build nginx image
- ONAP offline createrepo

Packaging

- Create SW Package
- Create Resource Package
- Create AUX Package
- Empty

Legend

- ONAP app Inputs – OOM helm charts should be eventually the only application specific input for creating whole ONAP offline platform.
- ONAP app other Inputs – these parts are currently required for ONAP offline platform data build and we don’t expect them to be changed frequently. They are currently manually prepared. Our stretch goal is to get rid of all those inputs together with community. So at the end there will be just OOM charts and docker images will be produced out of this
- Offline infrastructure inputs & tasks – rpms and RKE is an example k8s cluster provider
- Tasks – needs to be processed fully automatically (in CI chain)
Offline deployments testing

Current community testing

- Molecule syntax & functional testing
  
  https://github.com/ansible/molecule

  Tests triggered by offline-installer-master-review job for every relevant offline installer patch in gerrit
  https://jenkins.onap.org/view/01-Recent/job/offline-installer-master-review/

CI nightly execution in Samsung Poland lab (upstreaming planned)

- Adding download into CI pipeline OOM-1849
- Adding nexus blob build into CI pipeline OOM-1848
- Execution of robot HC’s from offline installer OOM-1806
Part III. Future plans for El-Alto and beyond
Future plans for El-Alto and beyond

- Align closely offline installer with ONAP development
  - General concept
  - Offline Installer CICD processes
  - Fully automated platform builds
  - CI installer

- Proposals for new features for Offline Installer

- Challenges, concerns
Current offline installer delivery process:

- Focused on major releases
- Partially manual gathering and maintaining list of artefacts
- Always behind latest changes
- Slow adaptation to changes
Goals for Offline Installer delivery process

- Planned improvements:
  - Closely tied up with oom repository (at least to the “latest stable”)
  - Frequent builds to quickly detect problems
  - Quick reaction to any detected issues
  - Automated artefacts gathering (where possible)
  - Continuing effort to remove online dependencies (where possible)
Offline Installer CI/CD processes - status overview

Package preparation

- Unit Test: To do
- Integration Test: To do
- System Test: Partially done
- Acceptance Test

Installation

- Done

Parallel to online deployment
Offline Installer CI/CD processes - System Test overview

- gathering and downloading artefacts
- building and packaging
- installing
- testing
Fully automated platform builds

- There are manually inserted inputs for the offline installer platform (usually for not often changing parts). Goal is to remove as many of them as possible.

**ONAP Offline Platform should continuously evolve, most of the manual inputs for builds should turn to void, what remains should be directly obtainable from the helm charts**
Automatically generated ONAP Offline Platform (Dublin)

Get OOM charts (release specific)
Get RKE specific docker images data list
Get npm packages data list
Get git repos data list
Get rpm packages data list

Get ONAP Dublin patch role
Automatically generate docker images data list

Download RKE docker image
Download application docker image
Download npm packages
Download git repos
Download rpm packages
Download http files

Get git repos data list
Get npm packages data list
Get pip packages data list
Http files data list

ONAP app Inputs – OOM helm charts should be eventually the only application specific input for creating whole ONAP offline platform.
ONAP app other Inputs – these parts are currently required for ONAP offline platform data build and we don’t expect them to be changed frequently. They are currently manually prepared. Our stretch goal is to get rid of all those inputs together with community. So at the end there will be just OOM charts and docker images will be produced out of this

Offline infrastructure inputs & tasks – rpms and RKE is an example k8s cluster provider

Tasks – needs to be processed fully automatically (in CI chain)

Download pip packages

Patch OOM helm charts
Build nexus blob

Build ansible image
Build nginx image
ONAP offline createrepo

Create SW Package
Create Resource Package
Create Package
Create AUX Package
Automatically generated ONAP Offline Platform (El-Alto and beyond)

Get OOM charts (release specific)
Get RKE specific docker images data list
Automatically generate docker images data list
Download RKE docker image
Download application docker image
Get rpm packages data list
Download rpm packages
Build nexus blob
Build ansible image
Build nginx image
ONAP offline createrepo
Create SW Package
Create Resource Package
Create AUX Package

Legend:
- ONAP app Inputs – OOM helm charts should be eventually the only application specific input for creating whole ONAP offline platform.
- Offline infrastructure inputs & tasks – rpms and RKE is an example k8s cluster provider
- Tasks – needs to be processed fully automatically (in CI chain)
Removing runtime dependencies along with community

- By having fast feedback from offline installer CI nightly, all attempts to introduce additional runtime dependency will be detected and analyzed together with appropriate ONAP project.

  *Subsequently decisions will be taken whether it’s feasible to do it better (e.g. adding pip package to dockerfile instead of adding shell script for runtime with pip install) OR offline platform has to be readjusted.*

  *All such issues will be visible earlier and pressure for “making it more production ready” will be raising.*

Example of solved issues:
APPC-1441, DCAEGEN2-1088, POLICY-1576, SO-1917

Example of currently opened issues:
CCSDK-1117, SDNC-685, OPTFRA-509
Offshore Installer CI/CD processes
- One of our biggest goals for El-Alto is to upstream to the community whole CI nightly builds solution for offline installer (inc. parts which are now developed for Dublin)

Open question:
- At which stage integrate it with ONAP life cycle?
- Under which project should it be stored?
- HW requirements are considerable (storage, bandwidth, RAM, CPU) - will there be room for it?
Motivation:

ONAP is supposed to be commercialized by many vendors, each will have its own add-ons, standards and customizations. It would be beneficial if Offline Installer could prepare framework to deliver and deploy such components.

Example use case:

Custom dashboard for monitoring ONAP components

Proposition of implementation:

Add to Offline Installer functionality to create application consisting ONAP and additional packages. Prepare packaging scripts to build such product and installer to installation it.
Upgrades in ONAP offline deployment

❖ Motivation:
Currently there is no procedure to upgrade ONAP installed with Offline Installer. The goal of this feature is to deliver a framework for generating and delivering upgrade package.

❖ Propositions of implementation:
- Add additional functionality to packaging that will compare two configurations, extract differences and create artefacts to be updated and removed.
  
  or

- Generate new package

In both cases Installation process will be extended with feature to deliver new package, process upgrade, rollback or remove outdated artefacts, depending on the upgrade result.
The Vision for Offline Installer

2018

Year of Start

First offline ONAP deployments
vFWCL functional check added
Create offline platform from data lists

2019

Year of Evolution

Offline installer officially part of ONAP community under OOM project umbrella
Evolving from manually created to auto-generated data lists
Working closely with community to remove runtime/install time add-ons

2020

Year of Success

Offline deployment compatibility feature adopted by ONAP projects
Footprint of offline artifacts decreased the bare minimum (just automatically generated docker image list is needed for offline deployments) – no git repos, http files, npm packages
Violation of offline compatibility checked by upstream jenkins jobs – jobs are maintained, people notified
Challenges, concerns

❖ There is no straightforward way how to analyze all online dependencies, our direction is to speed-up detection of such violations, but still it require reverse engineering if new dependency is introduced.

❖ Avoid patching OOM helm charts and do offline installer specific setup directly in OOM (e.g. root certificate distribution, application specific stuff related to offline setup, ..)
https://gerrit.onap.org/r/gitweb?p=oom/offline-installer.git;a=blob_plain;f=patches/offline-changes.patch;hb=refs/heads/master

❖ Capacity - Team size
<table>
<thead>
<tr>
<th>Top 10 contributors</th>
<th>Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tomas Levora</td>
<td><a href="mailto:t.levora@partner.samsung.com">t.levora@partner.samsung.com</a></td>
</tr>
<tr>
<td>Petr Ospaly</td>
<td><a href="mailto:p.ospaly@partner.samsung.com">p.ospaly@partner.samsung.com</a></td>
</tr>
<tr>
<td>Samuli Silvius</td>
<td><a href="mailto:s.silvius@partner.samsung.com">s.silvius@partner.samsung.com</a></td>
</tr>
<tr>
<td>Michal Ptacek</td>
<td><a href="mailto:m.ptacek@partner.samsung.com">m.ptacek@partner.samsung.com</a></td>
</tr>
<tr>
<td>Michal Zegan</td>
<td><a href="mailto:m.zegan@samsung.com">m.zegan@samsung.com</a></td>
</tr>
<tr>
<td>Bartlomiej Grzybowski</td>
<td><a href="mailto:b.grzybowski@partner.samsung.com">b.grzybowski@partner.samsung.com</a></td>
</tr>
<tr>
<td>Milan Verespej</td>
<td><a href="mailto:m.verespej@partner.samsung.com">m.verespej@partner.samsung.com</a></td>
</tr>
<tr>
<td>Mateusz Pilat</td>
<td><a href="mailto:m.pilat@partner.samsung.com">m.pilat@partner.samsung.com</a></td>
</tr>
<tr>
<td>Ronan Keogh</td>
<td><a href="mailto:ronan.keogh@est.tech">ronan.keogh@est.tech</a></td>
</tr>
<tr>
<td>Lasse Kaihlavirta</td>
<td><a href="mailto:l.kaihlavirt@partner.samsung.com">l.kaihlavirt@partner.samsung.com</a></td>
</tr>
</tbody>
</table>
Questions?
THANK YOU

Presenters:
Michal Ptacek
Mateusz Pilat