# ONAP Offline Deployment

Architecture of current Samsung solution


v0.2, 9.7.2018

Content:

# Version Control

| Version | Date | Modified by | Comment |
|---|---|---|---|
| 0.1 | 27.06.2018 | Michal Ptacek | Initial draft |
| 0.2 | 09.07.2018 | Michal Ptacek | Format improvements |
| | | | |
| | | | |

# Contributors

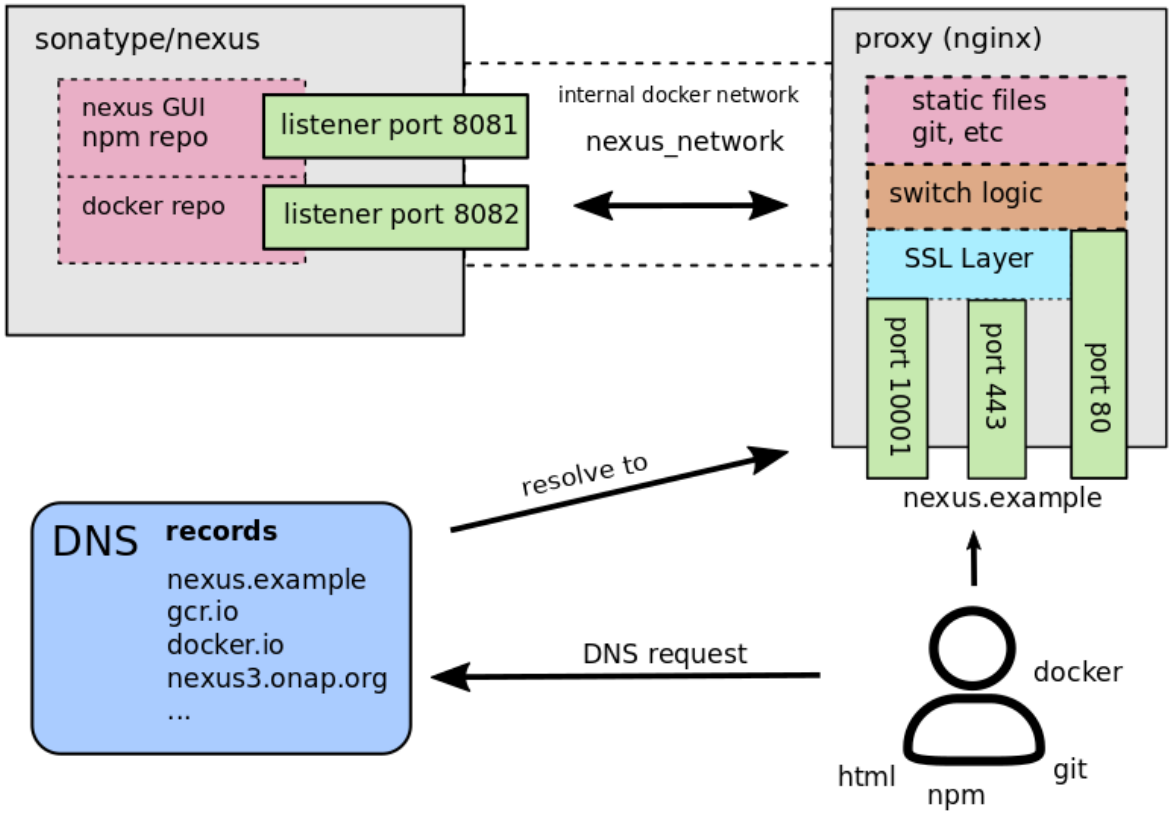
| Name | Mail |
|---|---|
| Michal Ptáček | m.ptacek@partner.samsung.com |
| Jaroslav Šafka | j.safka@partner.samsung.com |
| Milan Verešpej | m.verespej@partner.samsung.com |
| Vladyslav Malynych | v1.malynych@partner.samsung.com |
| | |

# 1. Concept description

Purpose of this document is to describe concept of current design for offline ONAP deployments. In our solution we are using local nexus repository. Docker containers are accessing repositories over https, the nginx is in there as reverse proxy to arrange that.

*"The main principle in place is that all docker images / pip packages / deb/ rpm packages / cloudify blueprints / git repos / files which are needed during ONAP OOM installation are downloaded during "online deployment" and pulled into local nexus accessible via nginx proxy. Original DNS Domains remains unchanged in original deployment scripts and are simulated by nginx."*

Nginx is fast proxy which provides also https support and distinguish to which port it should direct requests (e.g. Port 443 is shared with Nexus GUI and docker repository). Actually all traffic is directed to nginx which provide https access and then send data to nexus over private docker network (named nexus_network). Nginx is used also for git repos and normal http data based on host name.



# 2. Preparations

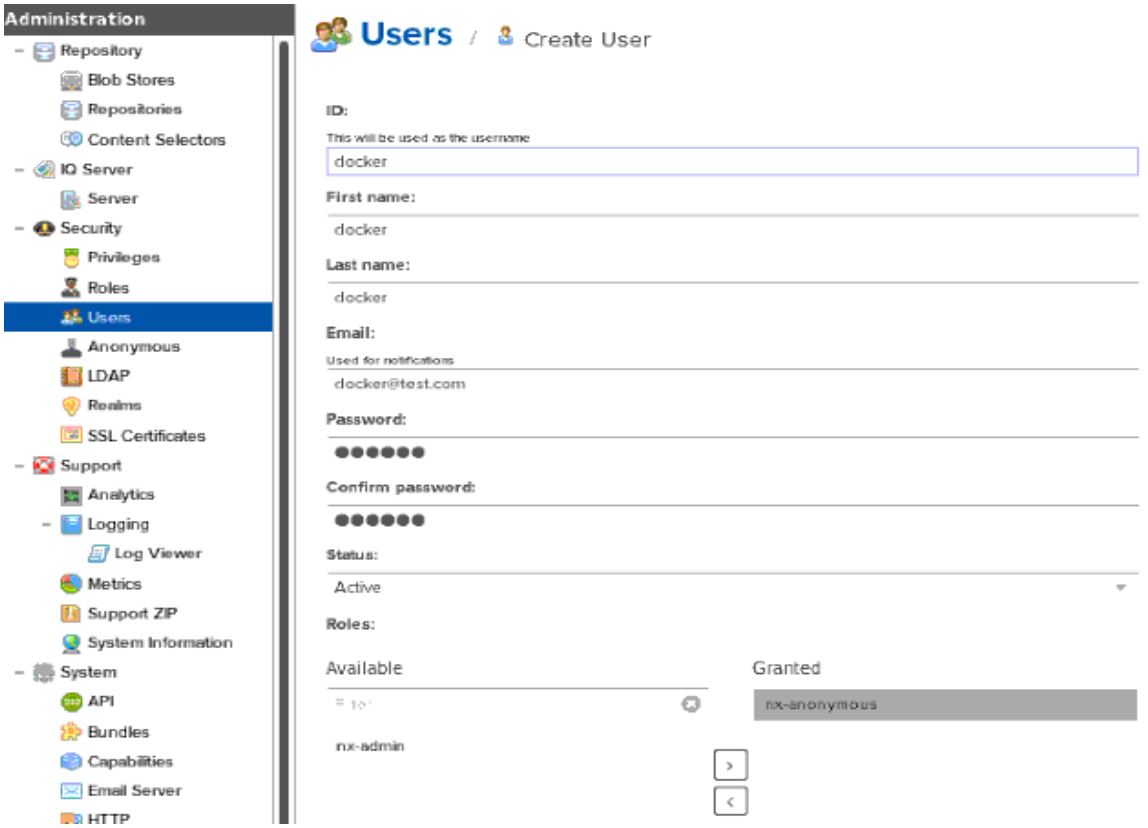
Essential part of our solution is in local nexus, which has to be populated by all required artifacts. We're using standard sonatype/nexus3 docker container.

```
 sonatype/nexus3          "sh -c ${SONATYPE_DIR}/start-nexus-repository-manager.sh"
19 hours ago        Up 3 hours          8081/tcp                                                nexus
```

## 2.1 Configure nexus - npm hosted repo, docker hosted repo, realms, users

Nexus must be accessed by his FQDN, like http://nexus.student12/
(Because on port 80 is more services/domains, and it is distinguished by his fqdn on proxy)

- Login (default: admin / admin123)
- Settings -> repositories > create repository #docker
  - Select docker (hosted)
  - Name: anything you want (ex: onap)
  - Repository connectors: HTTP set to 8082
  - Force basic authentication: UNchecked
  - Enable docker V1 API: checked
- Settings -> Repositories > create repository #npm
  - Select npm (hosted)
  - Name: npm-private
  - Deployment policy: TODO (Allow redeploy)
- Settings -> Security -> Realms
  - Add Docker Bearer Token Realm
  - Add npm Bearer Token Realm
- Settings -> Security -> Users
  - Add user docker / pwd docker
  - Rights same as anonymous docker




## 2.2 Populate nexus with docker images

When you want add images to nexus, then you must login as admin to each domain.
Ex:
$ docker login // login to default docker.io
$ docker login gcr.io // login to gcr.io repository

Images are saved from “online deployment” using “
# docker image save -o $filename".tar" $image:$tag
also tag is preserved (noted: in some cases more versions of same image are required by ONAP components)

Subsequently images are loaded into docker cache by
# docker load -i $filename
And eventually pushed into local nexus by
# docker push $image_name:$tag

## 2.3 Populate nexus with npm packages

There are multiple containers in ONAP (e.g. sdnc-dgbuilder, appc-dgbuilder & sdnc-portal) which requires npm packages. Those npm packages are collected from those containers from ~/.npm directory and copied into install server.

Those are then inserted into local nexus by following 3 steps:
# set registry to local nexus
$ npm config set registry https://nexus.student12/repository/npm-private/
# add user admin / admin123 to access nexus
$ npm adduser
# collect all tgz files and publish them into nexus
$ for i in `find . -name *.tgz`;do npm publish $i --access public;done

## 2.4. Other stuff

To be able to fully deploy ONAP several git repos, cloudify blueprints, binaries (helm, kubectl) and files are needed as well. Those are not changing that often and are statically stored inside our installation repo. They are deployed during kubernetes cluster deployment or later accessible via nginx simulated domains when they are requested from ONAP pods.

Additionally OOM was modified on several places to handle offline deployment:

**kubernetes/common/dgbuilder/templates/deployment.yaml**

```
38,45c38,39
<           command:
<           - /bin/bash
<           - -c
<           - >
<             UPDATE_HOSTS_FILE >> /etc/hosts;
<             UPDATE_NPM_REGISTRY;
<             cd /opt/onap/ccsdk/dgbuilder/;
<             ./start.sh sdnc1.0 && wait
---
>           command: ["/bin/bash"]
>           args: ["-c", "cd /opt/onap/ccsdk/dgbuilder/ && ./start.sh sdnc1.0 && wait"]
103d96
<
```

**kubernetes/dcaegen2/charts/dcae-bootstrap/templates/job.yaml**

```
74,81d73
<             - mountPath: /etc/pki/ca-trust/source/anchors
<               name: root-ca
<           command:
<             - /bin/sh
<           args:
<             - -c
<             - /scripts/bootstrap.sh; sleep infinity
<             - update-ca-trust extract; /scripts/bootstrap.sh; sleep infinity
106,108d97
<         - name: root-ca
<           hostPath:
<             path: /usr/local/share/ca-certificates/extra
```

**kubernetes/dcaegen2/charts/dcae-cloudify-manager/templates/deployment.yaml**

```
71,72d70
<           - mountPath: /etc/pki/ca-trust/source/anchors
<             name: root-ca
85,86d82
<                 echo -e '\nREQUESTS_CA_BUNDLE="/etc/ssl/certs/ca-bundle.crt"' >>
/etc/sysconfig/cloudify-restservice
<                 update-ca-trust extract
100,102d95
<         - name: root-ca
<           hostPath:
<             path: /usr/local/share/ca-certificates/extra
```

**kubernetes/onap/values.yaml**

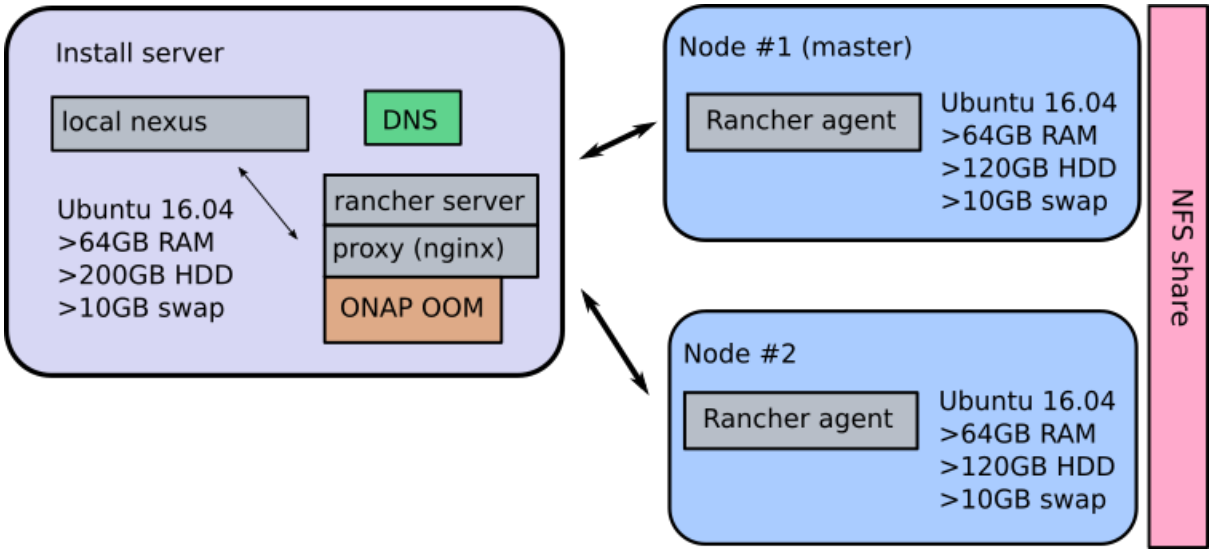
```
42,43c42
<   #pullPolicy: Always
<   pullPolicy: IfNotPresent
---
>   pullPolicy: Always
101d99
<     openStackUserName: "vnf_user"
103,105c101
<     openStackKeyStoneUrl: "http://1.2.3.4:5000"
<     openStackServiceTenantName: "service"
<     openStackEncryptedPasswordHere: "c124921a3a0efbe579782cde8227681e"
---
>     openStackVNFTenantId: "1234"
```

**kubernetes/sdnc/charts/sdnc-ansible-server/templates/deployment.yaml**

```
50,60c50,51
<           command:
<           - bash
<           - "-c"
<           - |
<             pip install /root/ansible_pkg/*.whl
<             dpkg -i /root/ansible_pkg/*.deb
<             cp /etc/ansible/ansible.cfg /etc/ansible/ansible.cfg.orig
<             cat /etc/ansible/ansible.cfg.orig | sed -e 's/#host_key_checking/host_key_checking/'
> /etc/ansible/ansible.cfg
<             touch /tmp/.ansible-server-installed
<             cd /opt/onap/sdnc
<             ./startAnsibleServer.sh
---
>           command: ["/bin/bash"]
>           args: ["-c", "cd /opt/onap/sdnc && ./startAnsibleServer.sh"]
86,87d76
<           - mountPath: /root/ansible_pkg
<             name: ansible-pkg
106,108d94
<         - name: ansible-pkg
<           hostPath:
<             path: /root/ansible_pkg
110,111c96
<       - name: "{{ include "common.namespace" . }}-docker-registry-key"
<
---
>       - name: "{{ include "common.namespace" . }}-docker-registry-key"
\ No newline at end of file
kubernetes/sdnc/charts/sdnc-portal/templates/deployment.yaml
52,58c52,53
<           command:
<           - /bin/bash
<           - -c
<           - >
<             UPDATE_HOSTS_FILE >> /etc/hosts
<             UPDATE_NPM_REGISTRY;
<             cd /opt/onap/sdnc/admportal/shell && ./start_portal.sh
---
>           command: ["/bin/bash"]
>           args: ["-c", "cd /opt/onap/sdnc/admportal/shell && ./start_portal.sh"]
```

# 3. Installation process

Before real ONAP offline deployment is started, self-contained archive is generated from our repo. This archive is encapsulating all data needed for ONAP deployment and it usually consumes ~60G of disc space. As of now we're supporting just one type of deployment consists of 3 nodes (install server + 2 kubernetes nodes)




We have bash scripting, which will perform complete deployment of kubernetes cluster and subsequently ONAP pods.

There are multiple configuration entries (either asked during deployment or provided prior deployment in local_repo.conf)

e.g.

```
LOCAL_IP=10.2.2.6
NEXUS_FQDN=nexus.oom-beijing-preRC2-master
NODE_MAIN_IP=10.2.2.12
NODE_SLAVE1_IP=10.2.2.11
CERT_ORGANIZATION=Samsung
CERT_COUNTRY=PL
CERT_STATE=Poland
CERT_LOCALITY=Krakow
```

## 3.1 Deploy infrastructure

In this part infrastructure will be deployed. More specifically local nexus, dns, docker, rancher & docker will be deployed on install server. Kubernetes nodes will get rancher agent running and form kubernetes cluster.

**Note:** During the execution of provided install script several questions will be asked, if you don't want to fulfil required information during script execution, there is an option to preset config file skipping this step.

- To execute a script simply run (from installation directory):
  - **`/path_to_script/selfinstall_onap_beijing.sh`**
    **Note:** it is not necessary to have the script in installation directory (e.g. onap).
- Answer questions asked by the script
  **Note:** questions will be asked only when script won't be able to find config file in the current folder, otherwise script will use existing config file.
- Wait until script finish execution.

**Check-box:**
One can verify, that 4.1 was successful in following way:

# kubectl get cs should display healthy etcd-0 component

```
root@oom-beijing-xenial-offline-master:~/install# kubectl get cs
NAME                 STATUS      MESSAGE                                                                                     ERROR
scheduler            Unhealthy   Get http://127.0.0.1:10251/healthz: dial tcp 127.0.0.1:10251: getsockopt: connection refused
controller-manager   Unhealthy   Get http://127.0.0.1:10252/healthz: dial tcp 127.0.0.1:10252: getsockopt: connection refused
etcd-0               Healthy     {"health": "true"}
```

**#** kubectl get nodes should display 2 kubernetes nodes in "Ready" state.

```
root@oom-beijing-xenial-offline-master:~/install# kubectl get nodes
NAME                                  STATUS    ROLES     AGE       VERSION
oom-beijing-xenial-offline-compute1   Ready     <none>    32m       v1.8.10-rancher1
oom-beijing-xenial-offline-compute2   Ready     <none>    27m       v1.8.10-rancher1
```

## 3.2 ONAP deployment

Before ONAP is deployed ./oom/kubernetes/onap/values.yaml in OOM should be configured accordingly to cover correct VIM (Openstack) credentials. Also number of deployed ONAP components might be modified in there.

ONAP installation is afterwards triggered via ./deploy_onap.sh

# 4. Planned activities and improvements

In July/2018 we are working on following topics to further improve our current offline solution.

1) vFWCL demo - Up to now we verified stability and vitality of offline installed environment by running robot healthchecks and we have reached all green 43/43 result. However being able to run vFWCL demo seems to be for us another important indication that it's working fine.

2) Adding RHEL support - important for some future ONAP customers

3) Automating / simplifying of preparation for installation medium - currently we need to make an online reference deployment in order to collect all artifacts for offline deployment. By having static list of versions we would like to automate that preparation part and remove necessity of running online installation deployment for obtaining installation medium.

(* this will fly just for static/verified deployments of ONAP - assuming no new dependency pops-up)

Afterwards we would like to further improve installation scripting and replace bash based core scripts by ansible solution, which should give us more benefits in a sense of flexibility of deployments, easier and faster maintainability of the overall solution and much more.

# 5. Questions / Concerns to be addressed

Our current understanding is that solution described in this document can't be used "out of the box" for community offline deployment approach. Instead of simulating currently used domains, we should probably extend current support in OOM to specify own nexus for deployments and eliminate areas where data are downloaded directly from internet (Adding them into current docker images ?)

If we want to have offline deployment possible, we need to prohibit changes across all ONAP projects, which can bring new dependency in upstream. How to arrange that ?

It looks like having local nexus is the only viable option as of now. Probably the only reasonable option would be like what we're doing, means prepare some way how to easy collect all dependencies and populate own nexus based on actual requirements, any other options ?

Apart of having offline nexus, there must be some installation logic utilizing modified OOM project in order to deploy ONAP offline way. As of now we have bash scripting for that and we plan to rewrite it to ansible solution, which is in our opinion more flexible and suitable for various ONAP deployments. Is ansible solution acceptable for that ? To which repo such implementation should be proposed ?

Do we need to run online deployment for getting all artifacts ?