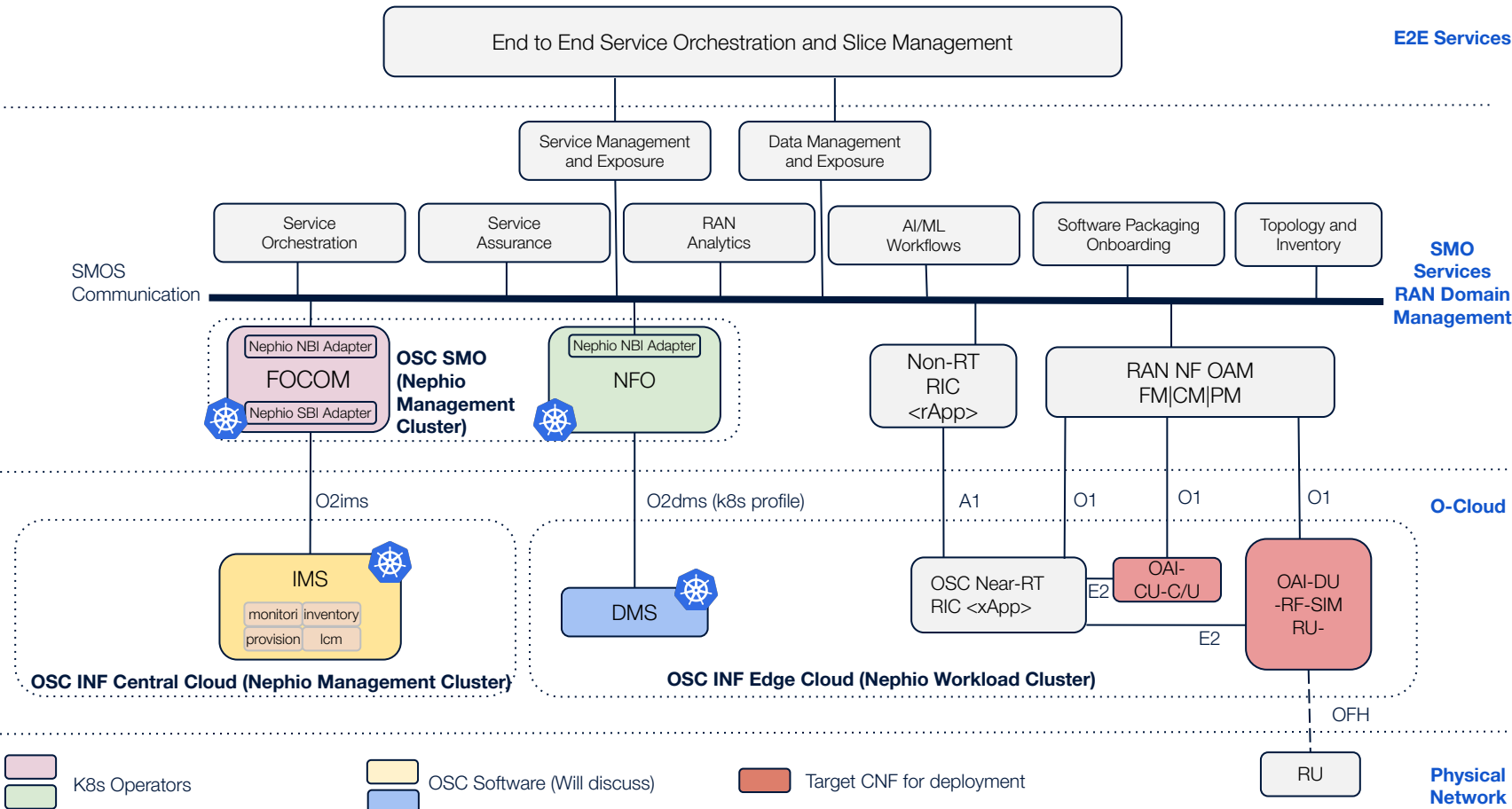


NF Orchestration Using SMO (User Story 2)

Contributors:

NF Orchestration Using SMO (Architecture)

O-RAN Network Function Deployment Instantiation Using SMO



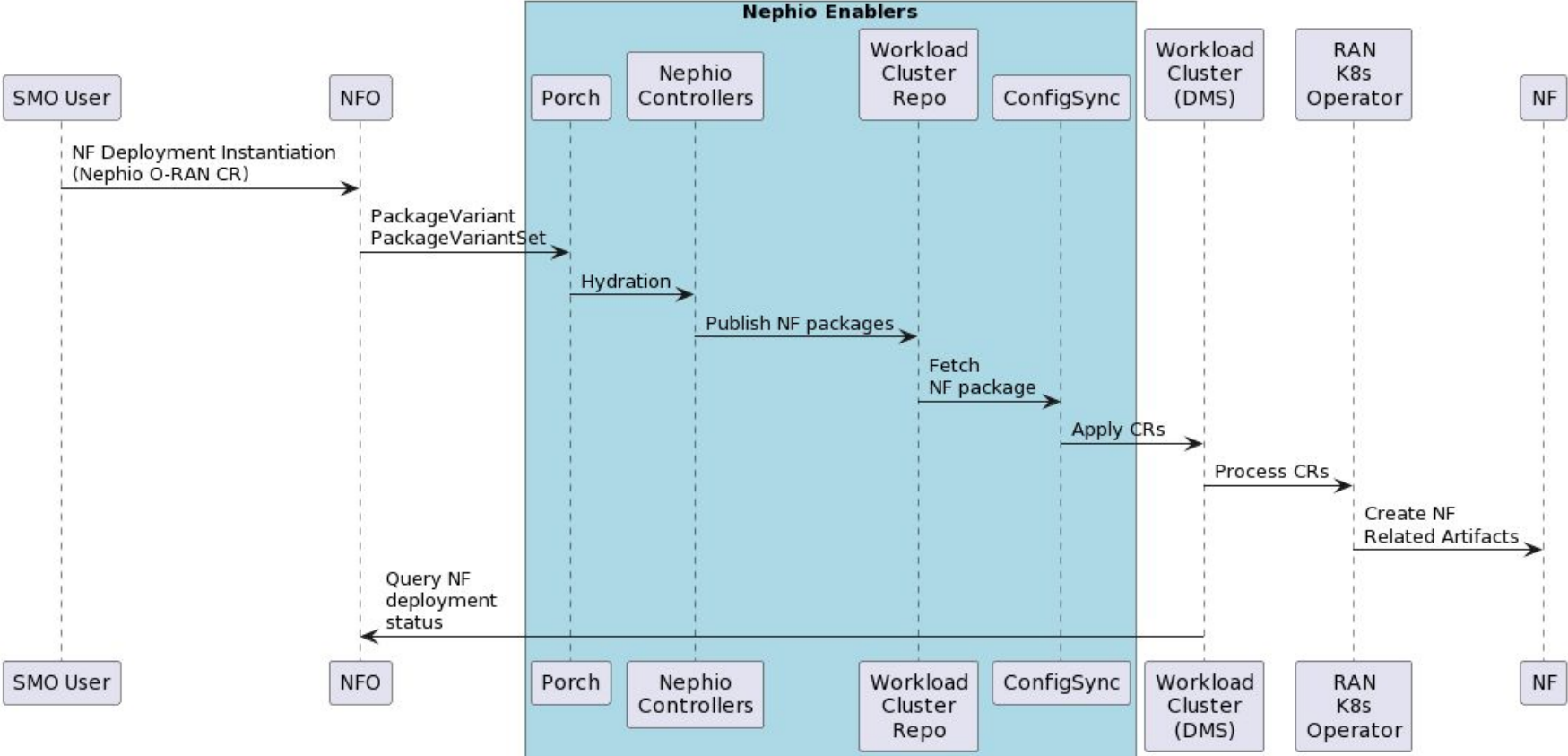
Scope and Priority

1. Deployment of OAI-DU, OAI-CU-CP/UP
 - a. (Current OSC mechanism) Via Helm Charts
 - b. (Extension of Nephio R2) Via KPT packages. Prerequisite: We will require Nephio RAN K8s operator to be deployed by FOCOM. Re-use of R2 RAN packages.
2. **Stretch:** Delete RAN NF Deployment
3. Update and modify of NF Deployment is not in scope
4. Timeline: Targeted for O-RAN J release and Nephio R3

Needed Software Development

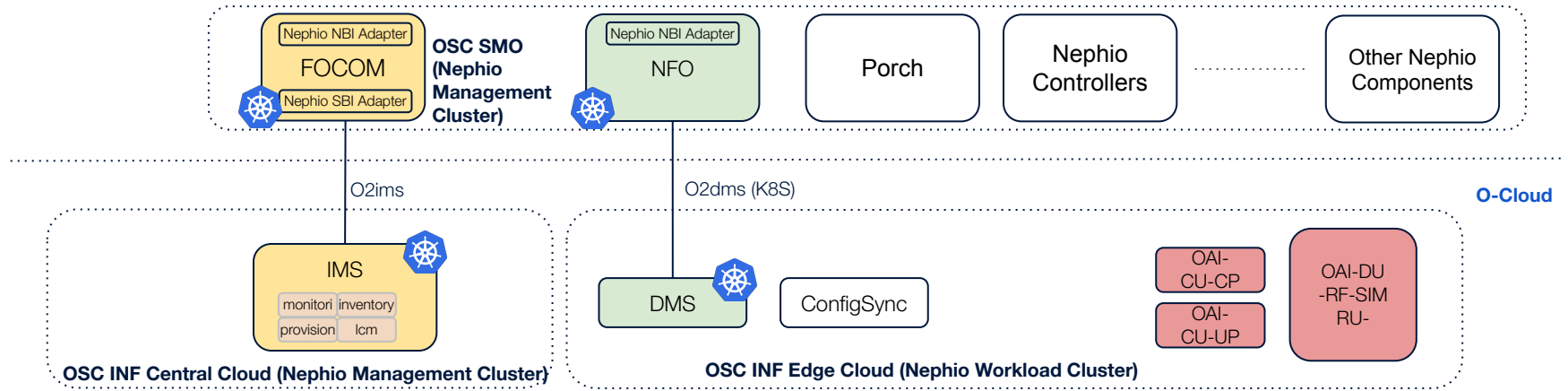
1. Development of NFO
 - a. Development of NFO service logic adapted to deployment of OSC K8S profile (NFs packaged as Helm Charts) **NOTE:** No Nephio enablers will be used at this step
 - b. Adapting the NFO service logic to deploy RAN NFs via Nephio Enablers (Nephio O-RAN CR).
 - c. NFO source code will be hosted in OSC Gerrit
2. Development and re-use of DMS, OSC Components (SMO, INF(IMS & DMS), OAM, RIC, Integration)
3. Re-use of helm-charts of OAI (DU, CU-C/U) - OSC
4. Re-Use of the Nephio RAN K8s Operator and KPT Packages

Sequence Flow Diagram (with Nephio Enablers)



Component Architecture

NFO in Nephio Management Cluster



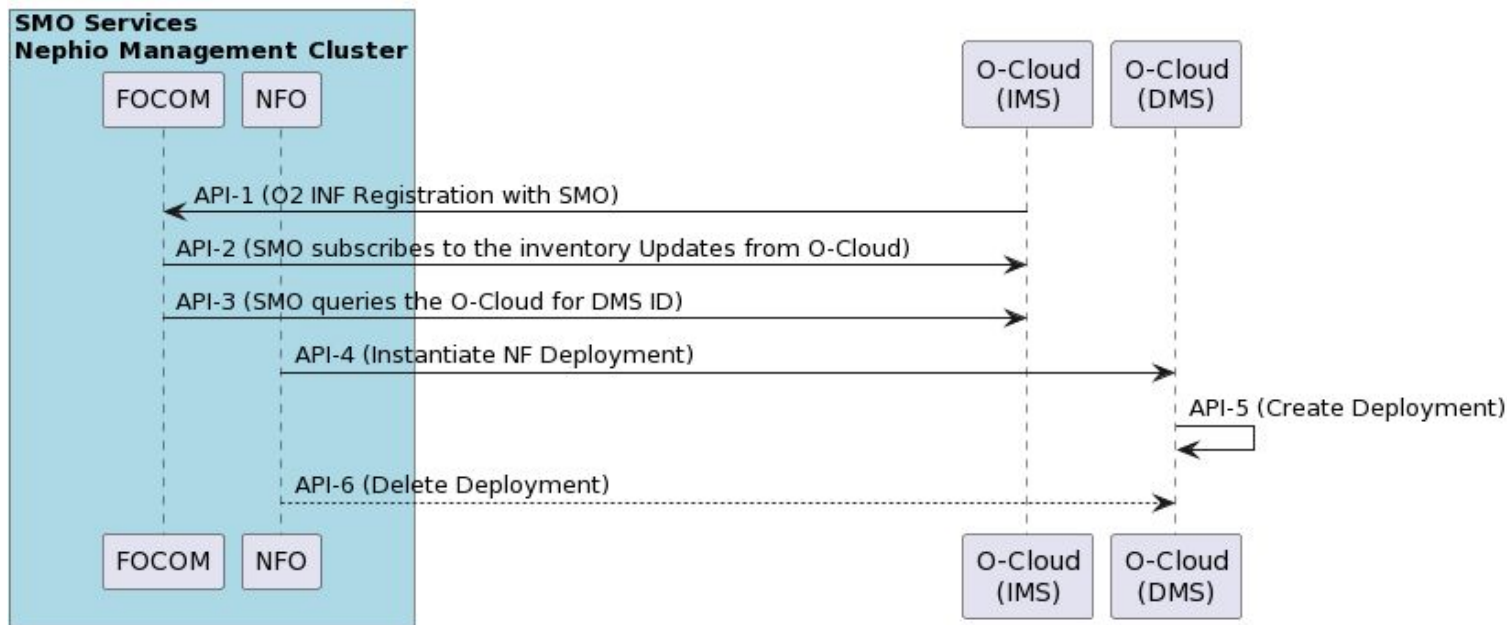
 K8s Operators

 Target CNF for deployment

Appendix

Sequence Flow Diagram (APIs in the Appendix Section)

Dependencies and O-RAN Deployment Flow



E2E test environment (Nephio)

1. Prerequisite
 - a. SMO (FOCOM, NFO) and OSC components are deployed on the management cluster.
 - b. FOCOM creates the K8s edge cloud (workload cluster) and DMS(k8S api) is brought up in this process
 - c. FOCOM installs the necessary CRDs and Operators required to handle the NF Deployment in the desired K8S Cluster
 - d. Cluster registration (DMS will register its information in IMS DB)
 - e. Nf deployment Kpt Packages/Helm-charts or other dependent artifacts should be available and accessible via NFO
2. Deployment:
 - a. NFO should fetch DMS information
 - b. NFO will receive the request for CU-CP deployment and will bring up CU-CP
 - c. NFO will receive the request for CU-UP and DU deployment and will bring up CU-UP and DU
 - d. We will use the KPT packages based deployment if and when ready. (Nephio)
3. This will be first implemented in OSC and then ported to the Nephio.
4. Integration test
 - a. The trigger to deploy would be initiated as a **Nephio O-RAN CR**/REST call to NFO (for the time being).
 - b. Stretch: Deploy NR-UE (Emulated package used in Nephio R2 and can be deployed via NFO because they are expressed as helm-charts) and make an end to end call

API Details/Design

API 1: O2 Registration, Provisioning INF platform with SMO endpoint configuration

- Configure INF platform with SMO endpoint address. This provisioning of INF O2 service will make a request from INF O2 service to SMO, that make SMO know the O2 service is working.

It needs SMO to have an API like “*http(s)://SMO_HOST:SMO_PORT/registration*”, which can accept JSON format data.

```
curl -X 'POST' \
  'http://'${IP}':30205/provision/v1/smo-endpoint' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "endpoint": "http://<SMO_HOST>:<SMO_PORT>/registration"
}'
```

API Details/Design

API-2 : O2 Inventory Subscription, create subscription in the INF O2 IMS

```
curl -X 'POST' \
  "http://${IP}:30205/o2ims_infrastructureInventory/v1/subscriptions" \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "callback": "http://SMO/address/to/callback",
    "consumerSubscriptionId": "<ConsumerIdHelpSmoToIdentify>",
    "filter": "<ResourceTypeNameSplitByComma,EmptyToGetAll>"
  }'
```

API Details/Design

API 3: Orchestrate CNF in helm chart , get the DMS Id in the INF O2 service

```
curl --location --request GET
"http://${IP}:30205/o2ims_infrastructureInventory/v1/deploymentManagers "

export dmsId=`curl --location --request GET
"http://${OAM_IP}:30205/o2ims_infrastructureInventory/v1/deploymentManagers"
2>/dev/null | jq      .[].deploymentManagerId | xargs echo

echo ${dmsId}
```

API Details/Design

API 4: Create NfDeploymentDescriptor on the INF O2 DMS

```
curl --location --request POST "http://${IP}:30205/o2dms/${dmsId}/O2dms_DeploymentLifecycle/NfDeploymentDescriptor" \
--header 'Content-Type: application/json' \
--data-raw '{
  "name": "cfwdesc1",
  "description": "demo nf deployment descriptor",
  "artifactRepoUrl": "http://'${NODE_IP}':30330",
  "artifactName": "firewall-host-netdevice",
  "inputParams":
    "{\n  \"image\": {\n    \"repository\": \"ubuntu\",\n    \"tag\": 18.04,\n    \"pullPolicy\": \"IfNotPresent\"\n  },\n  \"resources\": {\n    \"cpu\": 2,\n    \"memory\": \"2Gi\",\n    \"hugepage\": \"0Mi\",\n    \"unprotectedNetPortVpg\": \"veth11\",\n    \"unprotectedNetPortVfw\": \"veth12\",\n    \"unprotectedNetCidr\": \"10.10.1.0/24\",\n    \"unprotectedNetGwIp\": \"10.10.1.1\",\n    \"protectedNetPortVfw\": \"veth21\",\n    \"protectedNetPortVsn\": \"veth22\",\n    \"protectedNetCidr\": \"10.10.2.0/24\",\n    \"protectedNetGwIp\": \"10.10.2.1\",\n    \"vfwPrivateIp0\": \"10.10.1.1\",\n    \"vfwPrivateIp1\": \"10.10.2.1\",\n    \"vpgPrivateIp0\": \"10.10.1.2\",\n    \"vsnPrivateIp0\": \"10.10.2.2\"\n  }\n}",
  "outputParams": "{ \"output1\": 100 }"
}'
```

API Details/Design

```
curl --location --request GET
"http://${OAM_IP}:30205/o2dms/${dmsId}/O2dms_DeploymentLifecycle/NfDeploymentDescriptor"

export descId=` curl -X 'GET'
"http://${OAM_IP}:30205/o2dms/${dmsId}/O2dms_DeploymentLifecycle/NfDeploymentDescriptor"  -H
'accept: application/json'  -H 'X-Fields: id' 2>/dev/null | jq .[].id | xargs echo`

echo ${descId} (are these part of API-4?)
```

API-5: Create NfDeployment on the INF O2 DMS: This will trigger an event inside of the IMS/DMS, and use the K8S API to create a real pod

```
POST "http://${OAM_IP}:30205/o2dms/${dmsId}/O2dms_DeploymentLifecycle/NfDeployment" \
--header 'Content-Type: application/json' \
--data-raw '{
  "name": "cfw100",
  "description": "demo nf deployment",
  "descriptorId": "'${descId}'",
  "parentDeploymentId": ""
}'
```

API Details/Design

API-6: Delete the deployment (Stretch)

```
export NfDeploymentId=`curl --location --request GET
"http://${OAM_IP}:30205/o2dms/${dmsId}/O2dms_DeploymentLifecycle/NfDeployment" 2>/dev/null
|
jq .[].id | xargs echo`
echo ${NfDeploymentId} # Check the exported deployment id
curl --location --request DELETE
"http://${OAM_IP}:30205/o2dms/${dmsId}/O2dms_DeploymentLifecycle/NfDeployment/${NfDeploymentId}"
```