CNFO – Guilin CNF Improvements Overview

Seshu Kumar M (Huawei)
Lukasz Rajewski (Orange)
Konrad Bańka (Samsung)
Executive Summary - Provide CNF orchestration support through integration of K8s adapter in ONAP SO

› Support for provisioning CNFs using an external K8s Manager
› Support the Helm based orchestration
› leverage the existing functionality of Multi cloud in SO
› Bring in the advantages of the K8s orchestrator and
› Set stage for the Cloud Native scenarios

Owners: Lukasz Rajewski (Orange), Seshu Kumar M (Huawei), Srini Addepalli (Intel)

Business Impact - Enables operators and service providers to orchestrate CNFs based services along with the VNFs and PNFs

Business Markets - All operators and service providers that are intended to use the CNFs along with PNFs / VNFs

Funding/Financial Impacts - Reduction in the footprint of the ONAP for CNF support.

Organization Mgmt, Sales Strategies - There is no additional organizational management or sales strategies for this requirement outside of a service provider’s “normal” ONAP deployment and its attendant organizational resources from a service provider.
Guilin – CNF/Helm Day0/1 Flow
Instantiation of Helm Package with existing VNF model

Status and synchronization of instantiated k8s resources

✓ Helm Resource Artifact in SDC/SO
✓ Update of AAI Information by SO: vf-module

SO Orchestrates Helm Package -> not Heat Template

K8s Plugin as a standalone MS

✓ K8s Adapter in SO to interact directly with the K8s Plugin
✓ Enhance it to support the functions like the monitoring resources and status update (stretch)

Improvements in Helm customization/enrichment

Backward compatibility with CNF Macro Instantiation Workflow [Frankfurt] -> cvFW Example

Validation through flows cvFW Use Case
Improved Helm Distribution & Instantiation
ONAP - ETSI model Alignment

Integration of Native (K8s Adapter) with ETSi (SOL003 Adapter) paths in SO

Design/AAI CNF Model?
Virtual Developer & Testing Forum
June 22-25, 2020

CNF Model - Future

New model type into the SDC, AAI built over the helm charts as an input and would be distributed to the other ONAP components

Pros
✓ Onboard a design template to the SDC and create a new resource from that
✓ Requires a new model to be introduced
✓ Will be inline to the existing models of the heat and TOSCA based VNFs
✓ Can also be extended to other formats for CNF modeling

Cons
✓ Initial analysis for understanding the standard model
✓ Requires more effort and may span across multiple ONAP releases
✓ The grouping model currently used in ONAP may pose a one-one mapping to the other standard formats
Helm Chart on-boarded as an artifact type to the SDC and distributed to ONAP, AAI would persist it as existing VNF. Helm chart would be stored as flat file and add it to the CSAR package to be distributed.

**Pros**
- Easy to develop comparing with Approach 1
- Better reuse of the existing functional code

**Cons**
- Initial analysis for AAI persisting the CNF instance – Extend the VNF
- Very specific implementation and non extendable (non-helm)
{  
  "name": "virtualFirewall",
  "description": "",
  "data": [  
    {  
      "file": "base_template.yaml",
      "type": "HEAT",
      "isBase": "true",
      "data": [  
        {  
          "file": "base_template.env",
          "type": "HEAT_ENV"
        }  
      ]
    },
    {  
      "file": "base_template_cloudtech_k8s_charts.tgz",
      "type": "CLOUD_TECHNOLOGY_SPECIFIC_ARTIFACT"
    }
  ]
},

{  
  "name": "virtualFirewall",
  "description": "",
  "data": [  
    {  
      "file": "helm_base_template.tgz",
      "type": "HELM",
      "isBase": "true"
    }  
  ]
}

... from „small” change
SO – Native Orchestration of Helm

- Enabled Native CNF Day2 Operations
- CNF Status AAI Synchronization
- Closed-Loop Integration

... towards „big” things
Improved Helm Enrichment
Example of RB Profile’s manifest

---

version: v1

type:

values: override_values.yaml

configresource:

- filepath: resources/deployment.yaml

chartpath: templates/deployment.yaml

Example of profile’s structure

➢ K8s Plugin Requires profile to be archived as tar.gz file
➢ Complex Structure
➢ Profile contains Manifest + additional files
  ➢ Files: override.yaml + optional extra resources
  ➢ Extra resources: deployments.yaml, configmap.yaml etc.
  ➢ Extra resources replace existing helm templates or add new ones
  ➢ Values file gets merged with values file from helm package
Enrichment of Helm Package with CDS

We want to improve creation of profile and its upload to k8s plugin.
**Helm Package – Values**

```yaml
title: "My WordPress Site" # Sent to the WordPress template

global:
  app: MyWordPress

mysql:
  global:
    app: MyWordPress
  max_connections: 100 # Sent to MySQL
  password: "secret"

apache:
  global:
    app: MyWordPress
  port: 8080 # Passed to Apache
```

- Complex Structure of parameters
- Keys like `.mysql.global.app`
- Arrays in key names
- Too complex for simple key-value input from SO to k8s Plugin
- Good for Profile Templating in CDS
Helm Package - Structure

|-Chart.yaml
|-templates
| |-network_attachment_definition.yaml
| |-onap-private-net.yaml
| |-protected-private-net.yaml
| |-unprotected-private-net.yaml
|-values.yaml

- Yaml syntax
- Complex Structure
  - Chart descriptor
  - Templates
  - Override values file
  - [Nested Helm charts]
- Override values
  - Full Yaml structure possible
  - Flat key-value map is only one option
  - Nested values.yaml

We need flexibility to adapt Helm Charts provided by Helm application provider
Here the k8s-rb* inputs will be taken from resource-assignment-map

Each profile source must be listed as artifact-k8sprofile-content artifact

If profile source is a folder it needs to have associated a mapping file
RB Profile Native Upload – Profile Manifest

---

version: v1

type: values: override_values.yaml

configresource:

- filepath: resources/deployment.yaml
  chartpath: templates/deployment.yaml

- filepath: resources/configmap.yaml
  chartpath: templates/deployment.yaml

CBA profiling content

Manifest for profile1 source

k8s-rb-profile-source: complete profile with tar.gz or .tgz extension or folder with content

Profile will contain only files mentioned in the manifest

Templated file. Final file name without .vtl extension

Definitions of values used for Velocity (vtl) templating
K8splugin’s Enhancements
K8splugin’s v1 Enhancements

• New endpoint – Status API – with live information of content of instantiated resources
• Support for release-name provisioning at instantiation time
• Enhanced support for labeling of nested resource template
K8splugin’s v1 Status API - overview

Instance API (Get)
- Provide details of instantiation request (e.g. override parameters)
  - Provide information about vf-module’s instance release-name
  - Provide information about resources namespace in k8s cluster
  - Lists all resources assigned in k8s cluster provided in Helm Chart (specifying their name and GVK)

Status API
- Provide details of instantiation request (e.g. override parameters)
  - Provides total number of vf-module’s resources created in cluster
  - Provides all details of resources defined in Helm Chart as well as Pods created for given vf-module instance (similar to `kubectl describe X Y` command).
K8splugin's v1 Status API - overview

Instance API (Get):
```json
"id": "goofy_merkle",
"request": {
    "rb-name": "vfw",
    "rb-version": "plugin_test",
    "profile-name": "test_profile",
    "release-name": "",
    "cloud-region": "kud",
    "labels": {...
    },
    "override-values": {...
    },
    "namespace": "plugin-tests-namespace",
    "release-name": "test-release",
    "resources": [
        {
            "GVK": {
                "Group": "",
                "Version": "v1",
                "Kind": "ConfigMap"
            },
            "Name": "sink-configmap"
        },
        {
            "GVK": {
                "Group": "",
                "Version": "v1",
                "Kind": "ConfigMap"
            },
            "Name": "sink-configured"
        }
    ],
    "status": {
        "apiVersion": "v1",
        "data": {
            "protected_net_gw": "192.168.20.100",
            "protected_private_net_cidr": "192.168.10.0/24"
        },
        "kind": "ConfigMap",
        "metadata": {
            "creationTimestamp": "2020-10-05T13:45:43Z",
            "labels": {
                ""k8splugin.io/rb-instance-id": "goofy_merkle"
            }
        }
    }
}
```
Currently only Pods are listed in Status API resources apart from these defined directly in Helm Chart

Status API result can be used i.e. to determine

- Image names used in Deployment/Pods
- Status of Deployment/Pods
- Content of ConfigMaps/Secrets
- Assigned dynamic NodePort of Service
- Metadata/Annotations of Pods (e.g. to see custom, runtime assigned CNI network addresses)
K8splugin's Status API - demo

- Image names used in Deployment/Pods
- Status of Deployment/Pods
- Content of ConfigMaps/Secrets
- Assigned dynamic NodePort of Service
- Metadata/Annotations of Pods (e.g. to see custom, runtime assigned CNI network addresses)
REQ-341 ONAP Honolulu
Guilin - K8s Adapter (Helm) Flow Day 0/1

Focus on Native Day2 Operations

- AAI model changes
- SO AAI Data Update
- SO CNF Status
- SDC Distribution
- Helm Validation
- K8s Plugin – v2 APIs
- Native Day2 for CNF in CDS
Helm Package Day 0/1 + Day 2

1. Onboarded Helm Package
2. Designer
3. Override parameters
4. Day 0/1 Profile
5. Service-specific Helm Package
6. K8s Cluster Resources
7. Day 2 Configuration Template
8. Day 2 Configuration Parameters
9. Day 2 Configuration Instance

IP addresses from IPAM (Netbox)
Unique name generated from Naming Policy
Values retrieved from AAI
User Input Parameters

External Data Sources (REST/SQL)
Day 2 CNF Health Flow - ONAP

On-demand Health Check

1. On-demand Health Check
2. Background Health Check

Day 0/1 Synch by SO CNF Adapter

1. SO
2. AAI
3. K8s Plugin
4. K8s Res

DCAE

K8s Cluster

Prometheus
Thank You!