RIFT.ware™ version 8.2.0.2.117270

Release Notes

October 2020

Notice

Table of Contents

RIFT.ware 8.2.0.2 Release Notes

This guide describes the RIFT.ware 8.2.0.2 release, including new features, fixed and known issues, with their workarounds.

New and Changed Features for 8.2.0.2

RIFT.ware version 8.2.0.2 introduces enhancements to improve management.

Feature	PFR and JIRAs
Amazon Elastic Kubernetes Service Support	PFR-653
ETSI Based Scaling for Lifecycle Management Operations for CNFs and VNFs	PFR-644
ETSI SOL Specification Enhancements	N/A
Kubernetes as a VIM Enhancements	PFR-662
Launchpad Performance Metrics and Threshold Notifications	PFR-667
Licensing Management Enhancements	PFR-319
Multi-Cloud Orchestration	PFR-660
RIFT.ware Installation and Upgrade Enhancements	N/A
RIFT.ware Instantiation Enhancements	PFR-667
RIFT.ware Observability Enhancements	N/A
Scalable Monitoring Parameters Support	N/A
Support for Enhanced Datacenter Accounts and Informational Display Updates	PFR-659 PFR-666

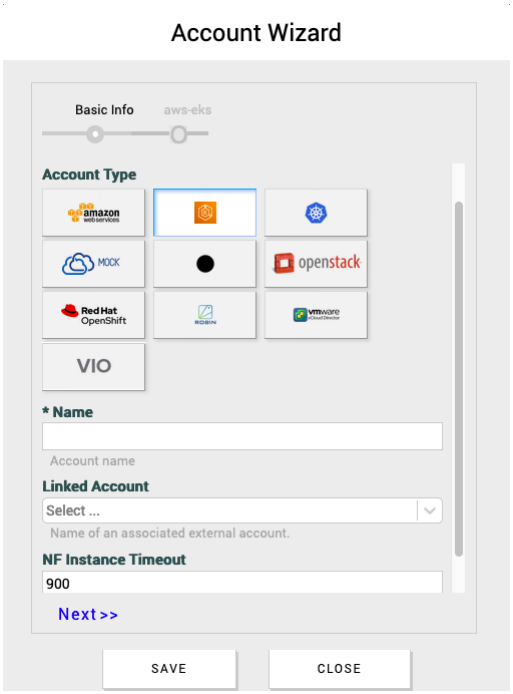
Amazon Elastic Kubernetes Service Support

RIFT.ware now supports orchestration of CNFs on Amazon Elastic Kubernetes Service (EKS) and EC2.

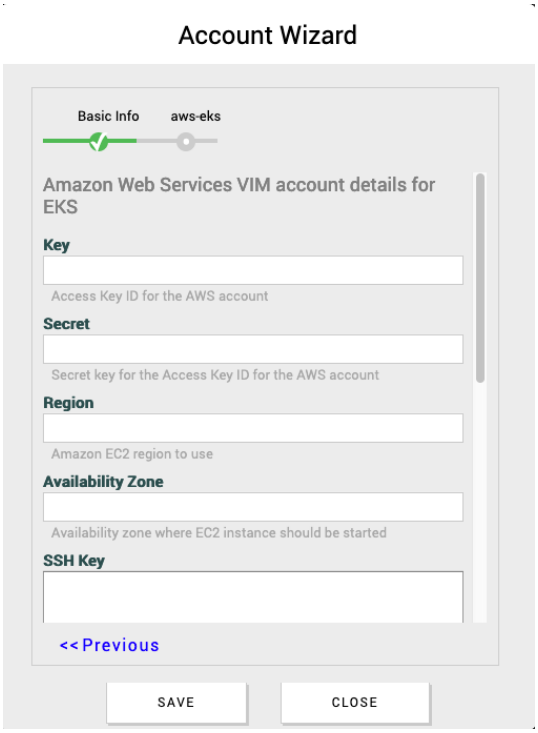
Add a AWS EKS Account

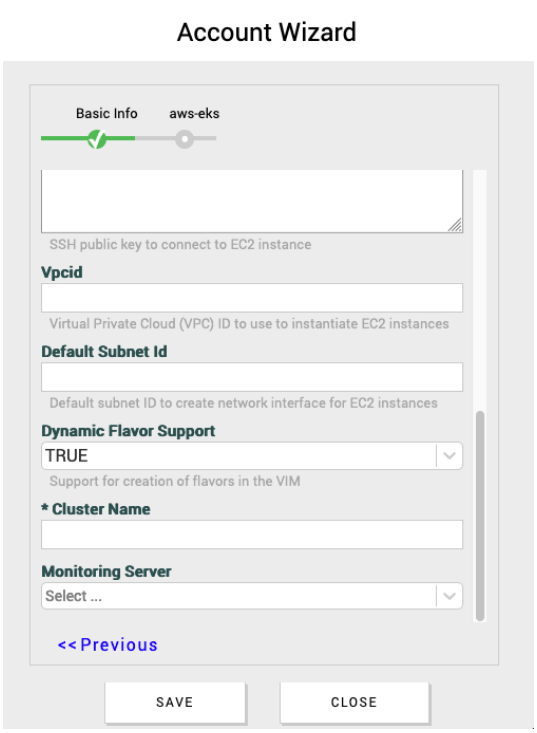
There is a new AWS EKS account type in the Datacenter section of the UI.

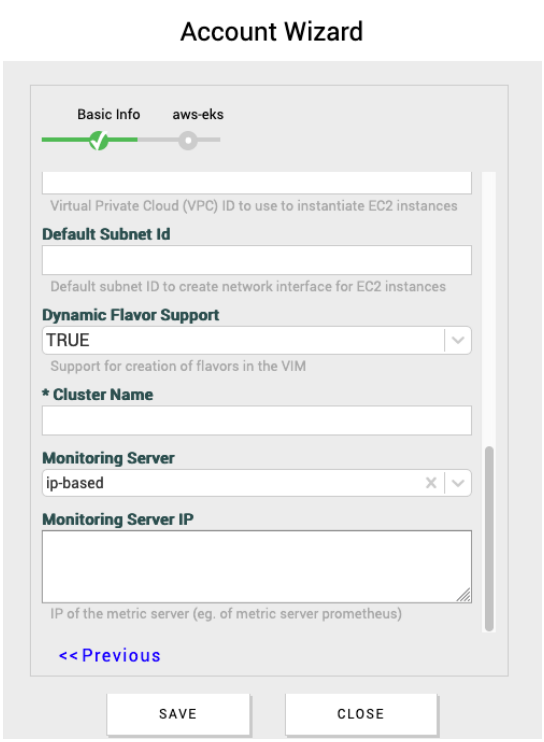
1. On the **Launchpad** menu, click **DATACENTERS**.
2. Click + to add a new datacenter. An account wizard appears.
3. Choose  in the **Account Type** section.
4. Add a unique account name in the ***Name** field.
5. Add a link to an associated external account in the **Linked Account** field.
6. Add the **Nf Instance Timeout** (SECONDS) information. This is the maximum time allocated for resource instantiation in the AWS EKS account. Default is 900.

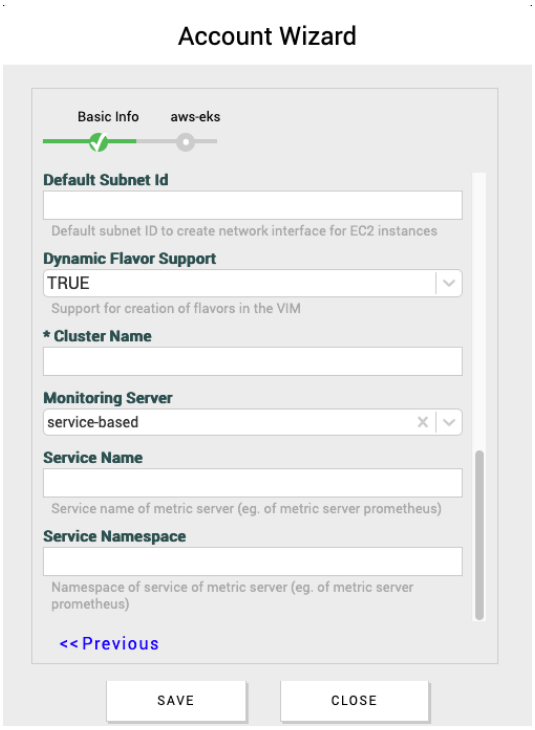


7. Click **Next >>** and provide the following Amazon Web Services VIM account details for EKS:
    - **Key**: Access Username for the AWS account.
    - **Secret**: Password for the AWS account.
    - **Region**: Name of the Amazon EC2 region.
    - **Availability Zone**: Availability zone where EC2 instance should be started.
    - **SSH Key**: SSH public key to connect to EC2 instance.
    - **Vpcid**: Virtual Private Cloud (VPC) ID to use to instantiate EC2 instances.
    - **Default Subnet Id**: Default subnet ID to create network interface for EC2 instances.
    - **Dynamic Flavor Support**: Support for creation of flavors in the VIM (True or False).
    - ***Cluster Name**: Name of the Cluster. This is a required field.
    - **Monitoring Server**: Select ip-based or server-based.
    - **ip-based**: If you choose an ip-based monitoring server, then add:
        - **Monitoring Server IP address**: IP of the metric server.
    - **server-based**: If you choose a server-based monitoring server, then add:
        - **Service Name**: Service name of the monitoring server.
        - **Service Namespace**: Namespace of service of metric server.

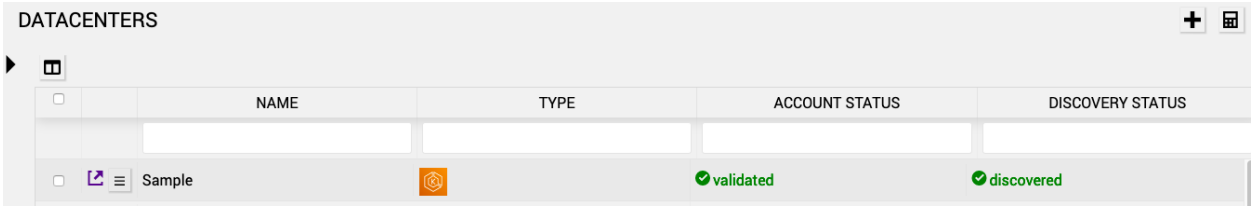








8. Carefully check your entries and click **SAVE**. After successfully creating the account, it will appear in the **DATACENTERS** list page as validated.



Before you can instantiate a service with CNFs on an AWK EKS account, you must add NSD and CNFD descriptions on the **Catalog** > **NSD/CNFD** List page.

AWS VPC Plugins

This feature also includes support for an Amazon specific CNI plugin for inter-pod connectivity. The AWS VPC plugin parameters are supported during VLD creation.

The following plugins are supported:

- **AWS VPC**: This plugin helps manage networking services for AWS resources. It includes concepts such as subnets, route tables and various kinds of gateways to enable one to connect to other VPCs and the internet.
- **ipvlan**: This plugin adds an ipvlan interface in a Kubernetes container. The Amazon CNI plugin can be integrated with IPvlan plugins.
- **host-device**: This plugin allows an operator to move a ‘host-device’ to a Kubernetes container in order to enable high speed data networking.

- **macvlan**: This plugin allows an operator to create a new MAC address and forwards all traffic to that container.

For more information on these plugins, see container networking plugins.

Model Changes

Added a new cloud account type in `rw-cloud:account` in the `rw-cloud:cloud` Data Model. This field includes the following variables:

Name	Type	Cardinality	Description
key	string	1	Access Key ID for the AWS account
secret	string	1	Secret key for the Access Key ID for the AWS account
region	string	1	Amazon EC2 region to use
availability-zone	string	1	Availability zone where EC2 instance should be started
ssh-key	string	1	SSH public key to connect to EC2 instance
vpcid	string	1	Virtual Private Cloud (VPC) ID to use to instantiate EC2 instances
default-subnet-id	string	1	Default subnet ID to create network interface for EC2 instances
dynamic-flavor-support	boolean	1	Support for creation of flavors in the VIM
cluster-name	string	1	
kube-config	string	1	Contents of Kube config file in YAML format
metric-server-ip	string	1	IP of the metric server
port	string	1	Port of the metric server
service-name	string	1	Service name of metric server
service-namespace	string	1	Namespace of service of metric server

ETSI Based Scaling for Lifecycle Management Operations for CNFs and VNFs

This feature introduces ETSI-based scale Lifecycle Management workflow support for both VNFs and CNFs. In previous releases, RIFT.ware added support for users to scale-in or scale-out a Network Service. In 8.2, we have expanded our support to allow an operator to scale-in or scale-out an individual CNF or VNF based on the capacity needed at the time.

The supported scale-out operations are retry, rollback, auto-rollback and heal. Scale-out is only possible when the Network Service is in the Running State. RIFT.ware supports retry for scale-in when the Network Service is in the Running or Failed-Temp state.

This feature introduces ETSI-based scale Lifecycle Management workflow support for both VNFs and CNFs. In previous releases, RIFT.ware added support for users to scale-in or scale-out a Network Service. In 8.2, we have expanded our support to allow an operator to scale-in or scale-out an individual CNF or VNF based on the capacity needed at the time. The supported scale-out operations are retry, rollback, auto-rollback and heal. Scale-out is only possible when the Network Service is in the Running State. RIFT.ware supports retry for scale-in when the Network Service is in the Running or Failed-Temp state.

This feature also introduces support for input variables in scaling CNFs. CNF and VNF scaling both support input parameters and pre, post config and service primitives. Lastly, this release includes CNFD, VNFD and VNFR model changes.

RIFT.ware CNF Scaling support

RIFT.ware now supports CNF scale-out of two use cases using a helm chart. RIFT.ware parses a helm chart and:

- supports scale-out and scale-in of Kubernetes deployment objects in a CNF.
- supports scaling operations using the Kubernetes horizontal pod autoscaler object.

The supported scale out operations include retry and heal. The supported scale-in operation is retry.

RIFT.ware VNF Scaling Support

RIFT.ware now supports VNF horizontal scale-out and scale-in on VDU objects in a VNF. A single operation can increase or decrease the number of VDUs in a VNF. The supported scale out operations include retry, rollback, auto-rollback and heal. The supported scale-in operation is retry.

Note: Release 8.2 introduces SOL003 VNF scaling support.

Scale Out

In this release, an operator can attempt scaling-out a CNF or a VNF when a Network Service is in the running state.

Note: Each scale operation is part of exec-scale-out and a transaction ID is generated for each operation.

CNF Scale Out

RIFT.ware CNF scale-out supports two use cases using a helm chart.

- Scale Out of Kubernetes Deployment Objects in a CNF
- Scaling Operations using the Kubernetes Horizontal Pod Autoscaler Object

Scale Out of Kubernetes Deployment Objects in a CNF

A scale-parameter must be defined in the CNFD input-parameters. The ‘is-scale-parameter’ must be True. While scaling out, an operator can pass the input parameters or input variables and the values file for the corresponding CNFR helm chart is updated. If the scale-out attempt fails, then the CNF and NSD is in the failed-temp state.

- **Retry**: The retry option attempts to bring up a failed resources and the service will not change. If retry is successful, then the CNF and NS is in the running state based on other resource states.
- **Heal**: The heal option is only possible on newly created Pods. If the heal attempt fails, then the CNF and NS is in the failed-temp state.

Scaling Operations using the Kubernetes Horizontal Pod Autoscaler Object

This scale operation occurs automatically from the Kubernetes cluster if horizontal pod autoscaler is configured. The corresponding scaled resources are updated in the CNFR.

VNF Scale Out

A scale group is defined in the VNFD that references a specific VDU object (connection point, volume and VM). If the scale-out attempt fails, then the VNF and NSD is in the failed-temp state. In this state, an operator can attempt to retry, rollback, auto-rollback or heal the VNF.

- **Retry**: The retry option attempts to bring up a failed resources and the service will not change. If retry is successful, then the VNF and NS is in the running state based on other resource states.
- **Rollback**: The rollback option removes the failed resources from the VNF. The VNF and NS moves to the running state based on other resource states.
- **Auto-Rollback**: The auto-rollback option sets while instantiating or scaling-out the VNF. This operation is similar to rollback.
- **Heal**: The heal option is possible on a newly created VDU. If the heal attempt fails, then the VNF and NS is in the failed-temp state.

If the scale-out attempt is successful, then config primitives run in the NS. Then, the VNFR in the configuration manager is updated with the latest VDU information.

Scale in

In this release, an operator can attempt scaling-in a CNF or a VNF when a Network Service is in the running or failed-temp states.

Note: Each scale operation is part of exec-scale-out and a transaction ID is generated for each operation.

CNF Scale In

RIFT.ware CNF scale-out supports two use cases using a helm chart.

- Scale in of Kubernetes Deployment Objects in a CNF
- Scaling Operations using the Kubernetes Horizontal Pod Autoscaler Object

Scale in of Kubernetes Deployment Objects in a CNF

A scale-parameter is defined in the CNFD input-parameters. The ‘is-scale-parameter’ must be True. While scaling in, an operator can pass the input parameters and the

values file for the corresponding CNFR helm chart is updated. If the scale-in attempt fails, then the CNF and NSD is in the failed-temp state.
Retry: The retry option attempts to bring up a failed resources and it will be idempotent. If retry is successful, then the CNF and NS is in the running state based on other resource states.

Scaling Operations using the Kubernetes Horizontal Pod Autoscaler Object

This scale operation occurs automatically from the Kubernetes cluster if horizontal pod autoscaler is configured. The corresponding scaled resources are updated in the CNFR.

VNF Scale In

A scale group is defined in the VNFD that references a specific VDU object (connection point, volume and VM). If the scale-out attempt fails, then the VNF and NSD is in the failed-temp state. In this state, an operator can attempt to retry, rollback, auto-rollback or heal the VNF.
Retry: The retry option attempts to bring up a failed resources and it will be idempotent. If retry is successful, then the VNF and NS is in the running state based on other resource states.
If the scale-in attempt is successful, then the VNFR in the configuration manager is updated with the latest VDU information.

RIFT.ware Primitive Support

In release 8.2., RIFT.ware supports the following vnf config primitives:

- Pre scale-out
- Pre scale-in
- Post scale-out
- Post scale-in

If the pre-config-primitives fail, then RIFT.ware ignores the failure. If the post-scale-out-primitive fails, then the VNFR config status fails. The heal primitives function the same as the config primitives.

UI Enhancements

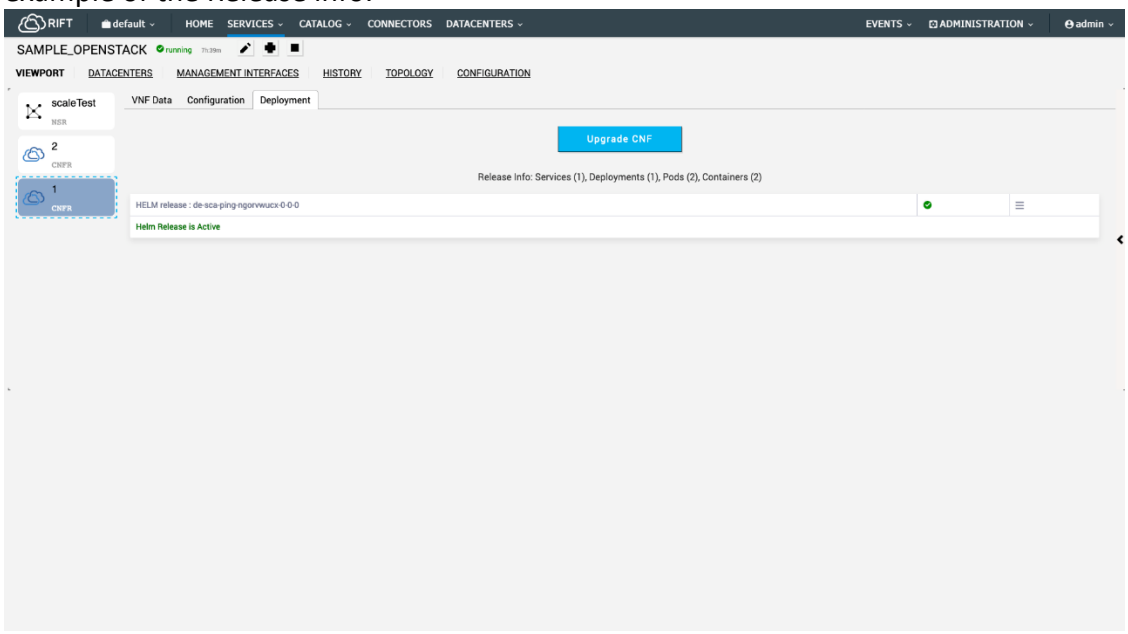
The **Deployment** tab in the **SERVICES → VIEWPORT** section of the UI displays the CNF and VNF level scaling information. The retry, rollback and heal for CNFs and VNFs is the same as other retry, rollback and heal operations.

Note: In release 8.2, there are no CNF or VNF scaling logs displayed in the Events tab of the UI.

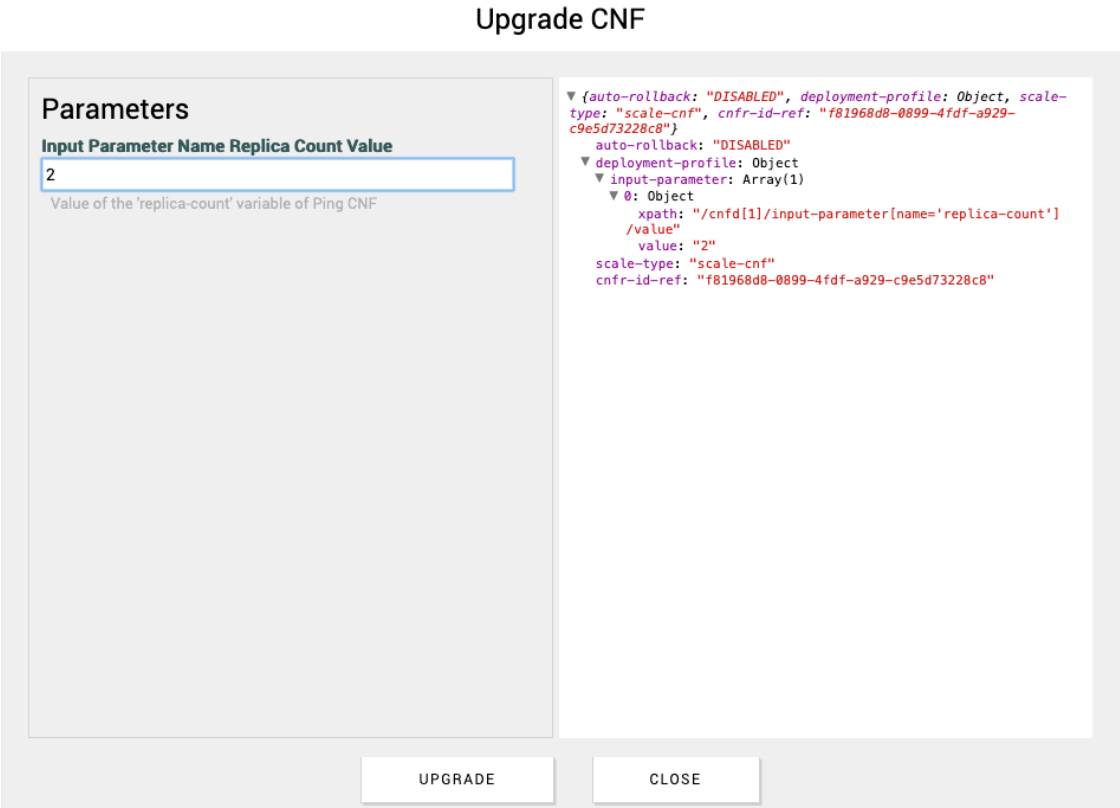
- Scaling CNFRs
- Scaling VNFRs

Scaling CNRFs

In release 8.2, there is a new **Upgrade CNF** button in the Deployment tab in the **SERVICES → VIEWPORT** section of the UI. This button triggers the CNF scale and it displays the number of services, deployments, pods, containers and release info. If any scale parameters are configured in the CNFD, then those will appear in this section of the UI. These values can change as part of the CNF scale. The screen shot below is an example of the Release info.

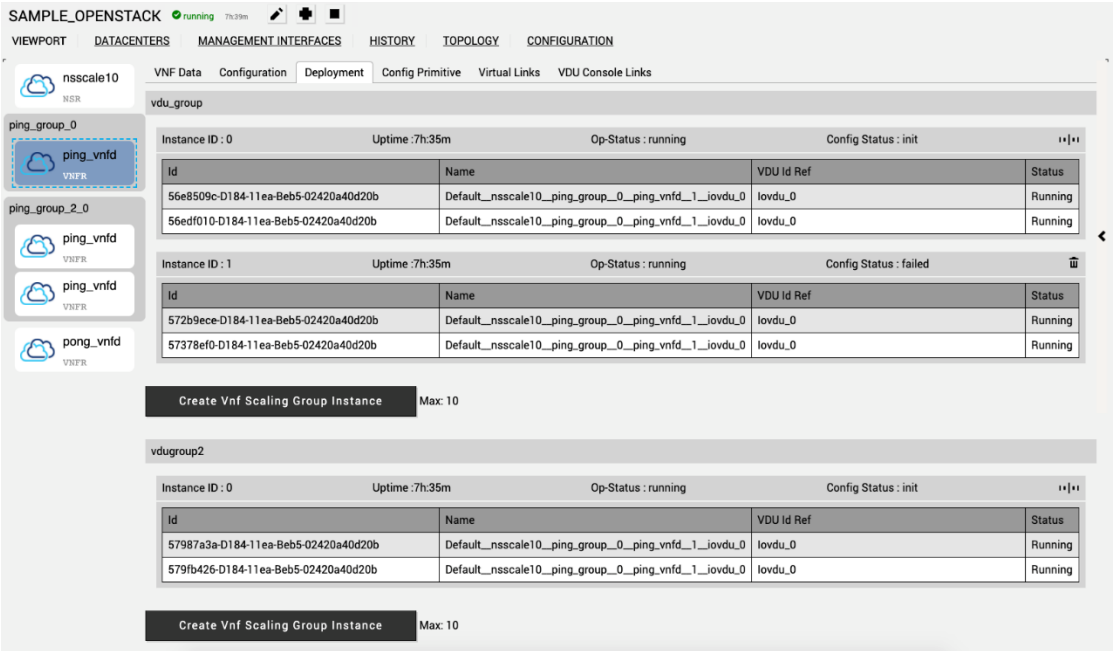


Click **Upgrade CNF** and the Upgrade CNF window appears. In this section, update the **Input Parameter Name Replica Count Value** and then click **UPGRADE**.



Scaling VNFRs

Scaling Groups, Instances and corresponding VDURs are now shown in the Deployment tab in the **SERVICES → VIEWPORT** section of the UI.



Model Changes

VNFD Changes

In this release, the VDU count in the VNFD is removed. There is a new scale-group field in the **VNFD Data Model (vnfd:vnfd)**.

Fields

Name	Type	Description
name	STRING	Name of this scaling group.
vdu-member	LIST	List of VDUs in this scaling group
min-instance-count	UINT32	Minimum instances of the scaling group which are allowed. These instances are created by default when the VNF is instantiated.
max-instance-count	UINT32	Maximum instances of this scaling group that are allowed in a single VNF. The VNF scaling will fail, when the number of service group instances exceed the max-instance-count specified.
scaling-config-action	LIST	List of scaling config actions

vdu-member

The vdu-member field contains a list of VDUs in the scaling group.

Fields

Name	Type	Description
vdu-ref	LEAFREF	member VDU index of this member VNF
count	UINT32	Count of this member VDU within this scaling group. The count allows to define the number of instances when a scaling action targets this scaling group

scaling-config-action

The scaling-config-action field contains a list of scaling config actions.

Fields

Name	Type	Description
config-primitive-type	ENUM	Scaling config primitive type
vnf-config-primitive-name-ref	LEAFREF	Reference to the VNF config primitive

VNFR Changes

In this release, there is a new scale-group field in **VNFR Data Model (vnfr:vnfr)**.

Fields

Name	Type	Description
config-primitive-type	ENUM	Scaling config primitive type
vnf-config-primitive-name-ref	LEAFREF	Reference to the VNF config primitive

CNFD changes

In this release, there is a new input-parameter field in input-parameter **CNFD Data Model (cnfd:cnfd)** model to specify whether the CNF is part of scaling.

RPC Changes

In release 8.2, the exec-scale-out/in RPC APIs are now being used to complete CNF and VNF scaling.

```
https://<launchpad_ip>:8008/api/operations/exec-scale-out

https://<launchpad_ip>:8008/api/operations/exec-scale-in
```


Scale-Network-Service-Input RPC Changes

```
grouping scale-network-service-input {
    uses rpc-common;

    leaf auto-rollback {
        description "Enable auto-rollback when NS scale-out fails";
        type auto-rollback-val;
    }
    leaf scale-type {
        description "Type of scale operation";
        type scale-type-val;
        default scale-ns;
    }
    choice scale-params{
      case scaling-ns{
        container scale-ns{
          leaf scaling-group-name-ref {
              description "Name of the scaling group";
              type string;
```

```
              mandatory true;
            }

            leaf instance-id {
              description "id of the scaling group";
              type uint16;
            }
            container deployment-profile {
              uses manotypes:input-variable-group;
              list vnf-input-parameter {
                description
                  "List of input parameters for Constituent VNFs that
can be specified when
                   instantiating a network service.";

                key "member-vnf-index-ref vnfd-id-ref";

                uses instantiate-vnf-input-profile;
              }

              list cnf-input-parameter {
                description
                  "List of input parameters for Constituent CNFs that
can be specified when
                   instantiating a network service.";

                key "member-id cnfd-id-ref";

                uses instantiate-cnf-input-profile;
              }
            }
          }
        }
        case scaling-vnf{
          container scale-vnf{
            leaf vnfr-id-ref {
              description
                "A reference to the VNFR which needs to be scaled";
              type yang:uuid;
              mandatory true;
            }
            leaf scaling-group-name-ref {
              description "name of the scaling group";
              type string;
              mandatory true;
            }

            leaf instance-id {
              description "id of the scaling group";
              type uint16;
            }
            container deployment-profile {
              uses manotypes:input-variable-group;
            }
          }
        }
```

```
               case scaling-cnf{
          container scale-cnf{
            leaf cnfr-id-ref {
              description
                "A reference to the CNFR which needs to be scaled";
              type yang:uuid;
              mandatory true;
            }
            container deployment-profile {
              uses manotypes:input-parameter-group;
            }
          }
        }
    }
  } /* end of scale-network-service-input */
```

RPC exec-scale-out

```
rpc exec-scale-out {
    description "Executes scale out request";

    input {
      uses scale-network-service-input;
    }

    output {
      leaf instance-id {
        description "id of the scaling group";
        type uint16;
      }

      uses rpc-transaction-id;

      leaf transaction-status {
        description "Enumerated status of NS scale out";
        type async-txn:async-transaction-status;
      }

      leaf transaction-status-details {
        description "The error message in case of a failed NS scale
out";
        type string;
      }

      uses manotypes:nats-rpc-output;
    }
  }
```

RPC exec-scale-in

```
rpc exec-scale-in {
    description "Executes scale in request";
    input {
      uses rpc-common;
      leaf scale-type {
        description "Type of scale operation";
        type scale-type-val;
```

```
        default scale-ns;
      }
      choice scale-params{
        case scaling-ns{
          container scale-ns{
            leaf scaling-group-name-ref {
              description "name of the scaling group";
              type string;
              mandatory true;
            }
            leaf instance-id {
              description "id of the scaling group";
              type uint16;
              mandatory true;
            }
          }
        }
        case scaling-vnf{
          container scale-vnf{
            leaf vnfr-id-ref {
              description
                "A reference to the VNFR which needs to be scaled";
              type yang:uuid;
              mandatory true;
            }
            leaf scaling-group-name-ref {
              description "name of the scaling group";
              type string;
              mandatory true;
            }

            leaf instance-id {
              description "id of the scaling group";
              type uint16;
              mandatory true;
            }
          }
        }

              case scaling-cnf{
          container scale-cnf{
            leaf cnfr-id-ref {
              description
                "A reference to the CNFR which needs to be scaled";
              type yang:uuid;
              mandatory true;
            }
            container deployment-profile {
              uses manotypes:input-parameter-group;
            }
          }
        }
      }
    }

    output {
      leaf instance-id {
```

```
        description "id of the scaling group";
        type uint16;
      }

      uses rpc-transaction-id;

      leaf transaction-status {
        description "Enumerated status of NS scale in";
        type async-txn:async-transaction-status;
      }

      leaf transaction-status-details {
        description "The error message in case of a failed NS scale
in";
        type string;
      }

      uses manotypes:nats-rpc-output;
    }
  }
```

Scale Out CNF Example

```
{
    "input": {
        "nsr-id-ref": "f08c9b59-8976-4267-a130-94524159cfeb",
        "project-name": "default",
        "scale-type": "scale-cnf",
        "scale-cnf": {
            "cnfr-id-ref": "47bb309a-963a-4f63-ac06-af45f5d19f86",
            "deployment-profile": {
                "input-parameter": [
                    {
                        "xpath": "/cnfd[1]/input-
parameter[name='replica-count']/value",
                        "value": "4"
                    }
                ]
            }
        }
    }
}
```

Scale Out VNF Example

```
{
    "input": {
        "nsr-id-ref": "5dc8dda8-1ca4-4be4-8bbb-910ac7fb048f",
        "project-name": "default",
        "scale-type": "scale-vnf",
        "auto-rollback": "ENABLED",
        "scale-vnf": {
            "vnfr-id-ref": "4a9918b3-e1e7-499b-a60a-416577764bc7",
            "scaling-group-name-ref": "vdu_group",
            "instance-id": 51
        }
    }
}
```

Scale In CNF Example

```
{
    "input": {
        "nsr-id-ref": "f08c9b59-8976-4267-a130-94524159cfeb",
        "project-name": "default",
        "scale-type": "scale-cnf",
        "scale-cnf": {
            "cnfr-id-ref": "47bb309a-963a-4f63-ac06-af45f5d19f86",
            "deployment-profile": {
                "input-parameter": [
                    {
                        "xpath": "/cnfd[1]/input-
parameter[name='replica-count']/value",
                        "value": "4"
                    }
                ]
            }
        }
    }
}
```

Scale In VNF Example

```
{
    "input": {
        "nsr-id-ref": "f50d0a4a-56fd-496c-abf3-6ac7c839afed",
        "project-name": "default",
        "scale-type": "scale-vnf",
        "scale-vnf": {
            "vnfr-id-ref": "0721ff46-8ed5-49c4-91e4-0a6c03770d7f",
            "scaling-group-name-ref": "vdu_group",
            "instance-id": 52
        }
    }
}
```

input-variable-group

In this release, the `input-variable-group` field in the below `exec-scale-out/scale-cnf/deployment-profile` RPC APIs is now being used to pass input variables with scaling out or in a VNF and CNF.

```
https://<launchpad_ip>:8008/api/operations/exec-scale-out

https://<launchpad_ip>:8008/api/operations/exec-scale-in
```

Note: input-parameter and input-variables are both supported in the exe-scale in/out RPC APIs.

ETSI SOL Specification Enhancements

In release 8.2, we have updated RIFT.ware to support the latest ETSI SOL 001, 003, 004, 005 and 007 specifications. Refer to the below documentation for additional information.

- ETSI GS NFV-SOL 001 V3.3.1 (2020-09)
- ETSI GS NFV-SOL 004 V2.8.1 (2020-08)
- ETSI GS NFV-SOL 007 V2.8.1 (2020-08)
- ETSI GS NFV-SOL 003 V3.3.1 (2020-08)
- ETSI GS NFV-SOL 005 V2.7.1 (2020-01)

Note: There are no deviations in any of these APIs for RIFT.ware besides the notes indicated below.

ETSI GS NFV-SOL 001, ETSI GS NFV-SOL 004 and ETSI GS NFV-SOL 007 Specification Enhancements

This release includes changes to the following specifications:

- NFV descriptors based on TOSCA specification (ETSI GS NFV-SOL 001)
- VNF Package and PNFD Archive specification (ETSI GS NFV-SOL 004)
- Network Service Descriptor File Structure Specification (ETSI GS NFV-SOL 007)

SOL001 VNFD Updates

The below SOL 001 VNFD enhancements are now supported in RIFT.ware in release 8.2.

- BootData for customizing a virtualized compute resource at boot time.

ETSI GS NFV-SOL 003 Specification Enhancements

This release includes changes to the RESTful protocols specification for the Or-Vnfm Reference Point (ETSI GS NFV-SOL 003).

VNF Package Management Updates

This release introduces a new set of VNF Package Management APIs with `vnfdId` as the identifier.

- `/vnfpkgm/v2/onboarded_vnf_packages`
- `/vnfpkgm/v2/onboarded_vnf_packages/{vnfdId}`
- `/vnfpkgm/v2/onboarded_vnf_packages/{vnfdId}/vnfd`

- `/vnfpkgm/v2/onboarded_vnf_packages/{vnfdId}/manifest`
- `/vnfpkgm/v2/onboarded_vnf_packages/{vnfdId}/package_content`
- `/vnfpkgm/v2/onboarded_vnf_packages/{vnfdId}/artifacts`
- `/vnfpkgm/v2/onboarded_vnf_packages/{vnfdId}/artifacts/{artifactPath}`

The release also introduces two additional APIs:

- `/vnfpkgm/v2/vnf_packages/{vnfPkgId}/manifest` - /manifest API to retrieve manifest file
- `[/vnfpkgm/v2/vnf_packages/{vnfPkgId}/artifacts]` - /artifacts API for bulk fetch of artifacts with specific filters for mano/non-mano.

ETSI GS NVF-SOL 005 Specification Enhancements

This release includes changes to the RESTful protocols specification for the Os-Ma-nfvo Reference Point (ETSI GS NFV-SOL 005).

VNF Package Management Updates

This release introduces the following new APIs:

- `[vnfpkgm/v2/vnf_packages/{vnfPkgId}/manifest]` - /manifest API to get manifest file.
- `[/vnfpkgm/v2/vnf_packages/{vnfPkgId}/ext_artifacts_access]` - /ext_artifacts_access API to set and read access config for external artifacts.

  Note: This API is not supported by RIFT.ware.

- `[/vnfpkgm/v2/vnf_packages/{vnfPkgId}/artifacts]` - /artifacts API for bulk download of artifacts. This API supports specific query parameters for controlling mano/non-mano.

NSD Package Management Updates

This release introduces the following new APIs:

- `[/nsd/v2/ns_descriptors/{nsdInfoId}/nsd]` - /nsd API to get content of NSD in an NS descriptor archive.
- `[/nsd/v2/ns_descriptors/{nsdInfoId}/nsd_archive_content]` - /nsd_archive_content API to fetch and upload NS descriptor archive.
- `[/nsd/v2/ns_descriptors/{nsdInfoId}/manifest]` - /manifest API to get manifest file of an NSD archive.

Kubernetes as a VIM Enhancements

Release 8.2 includes changes to the way that RIFT.ware handles Kubernetes accounts. These types of accounts are now treated as a cloud accounts and they are managed by the cloud account API. The process of creating an account is now straightforward in the RIFT.ware UI. This feature also includes discovery support for Kubernetes accounts. In addition, this release introduces a new input parameter framework where an operator can onboard a helm chart, create a CNFD in the RIFT.ware catalog and then identify the input variables for that specific CNFD.

Kubernetes Account Changes

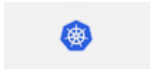
In release 8.2, Kubernetes accounts have been removed from the list of connectors and merged with cloud accounts. Kubernetes accounts are now treated as cloud accounts and are managed by the cloud account API interfaces. The Kubernetes accounts are also now part of the Datacenters section in the UI.

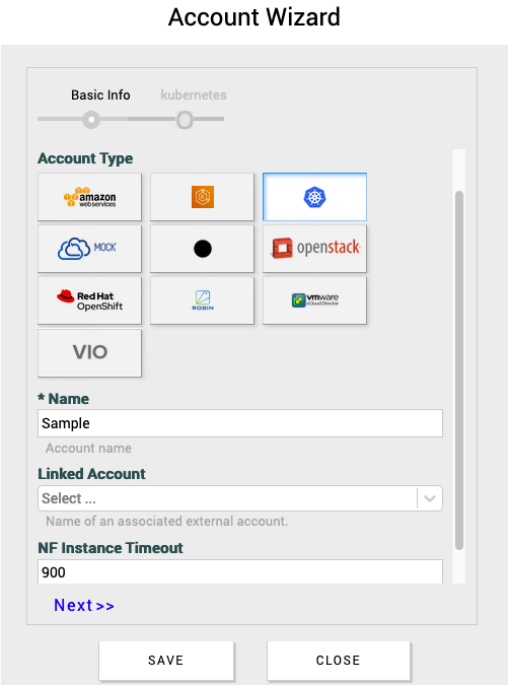
Kubernetes Account Enhancements in the UI

Kubernetes accounts are now configured in the cloud account configuration section. Prior to release 8.2, creating a cloud account was a yang model choice differentiated by the account-type. Kubernetes accounts were a yang model choice with Kubernetes-account-type differentiating the support account configurations. Now, all cloud and Kubernetes accounts are merged as cloud-account-types.

Configuring a Kubernetes Account

Previously, a user created a Kubernetes account in the **CONNECTORS** section on the UI. A user now adds a Kubernetes account in the **DATACENTERS** section.

1. On the **Launchpad** menu, click **DATACENTERS**.
2. Click + to add a new datacenter. An account wizard appears. Fill in the appropriate details for the specific datacenter.
3. Choose  in the **Account Type** section.
4. Add a unique account name in the ***Name** field.
5. Add a link to an associated external account in the **Linked Account** field.
6. Add the **Nf Instance Timeout** (SECONDS) information. This is the maximum time allocated for resource instantiation in the Kubernetes account. Default is 900.

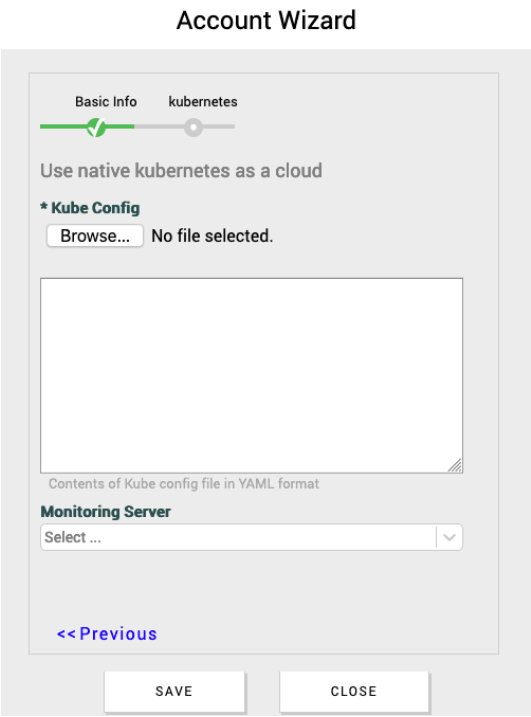


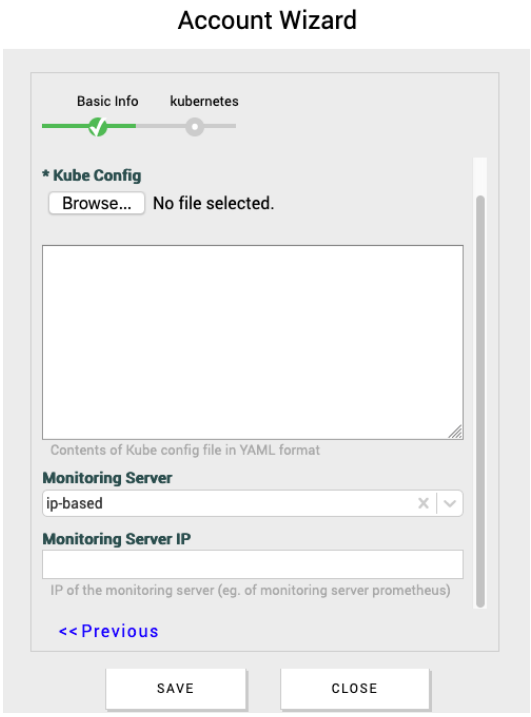
7. Click **Next >>** and provide the following Kubernetes account details:
   - **Kube Config***: Click Browse to provide the Kubernetes configuration file in YAML format or type the Kube Config in the Kube Config field.

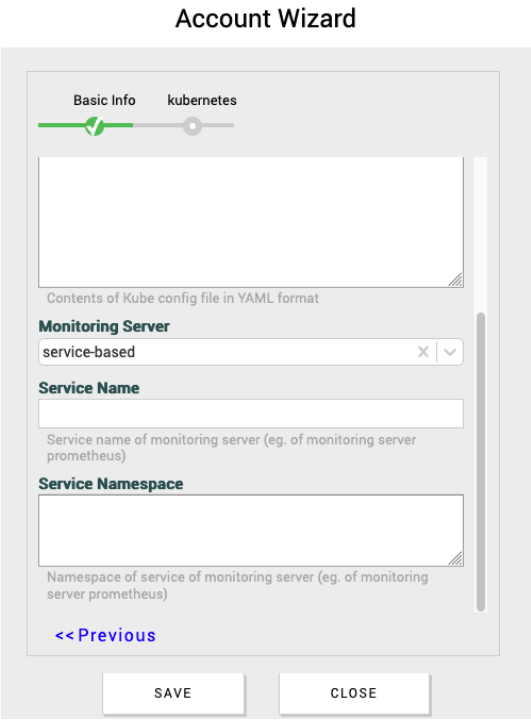
     Organizing Cluster Access using Kubeconfig Files

Note: This configuration file must be provided by the operator.

   - **Monitoring Server**: Select ip-based or server-based.
     - **ip-based**: If you choose ip-based monitoring server, then add the Monitoring Server IP address.
     - **server-based**: If you choose server-based monitoring server, then add:
       - **Service Name**: Service name of monitoring server
       - **Service Namespace**: Namespace of service of monitoring server







8. Carefully check your entries and click **SAVE**.

Before you can instantiate a service with CNFs on an Kubernetes account, you must add NSD and CNFD descriptions on the **Catalog** > **NSD** and **CNFD** List pages.

Discover Resources in a Kubernetes Account

This release also includes discovery support for Kubernetes accounts. Not all of the resources listed in the discovery section are applicable for Kubernetes Accounts.

The following resources:

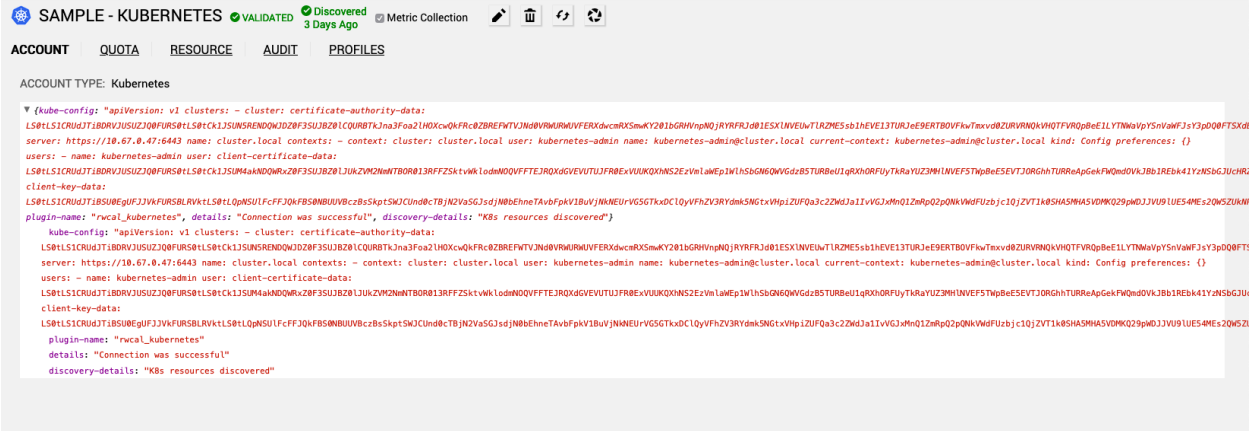
- **are supported** for Kubernetes Account discovery: server, network, volume, public-ip, cnfr.
- **are applicable** for Kubernetes accounts but they are currently not supported as part of the Kubernetes Account discovery: role, security-group (Kubernetes: network-policy), placement-group-vim-availability
- **are not applicable** for Kubernetes Account discovery: flavor, interface, image.

Some of the resource names are different for a Kubernetes account. Refer to the bulleted list below to identify the meaning in a Kubernetes Account.

- Server = Pod
    - Id = Pod ui (id and name both show the interface name because there is no uuid for an intf.)
    - Name = Pod name (id and name both show the interface name because there is no uuid for an intf.)
    - Volume = persistent-volume-claim attached to the pod.

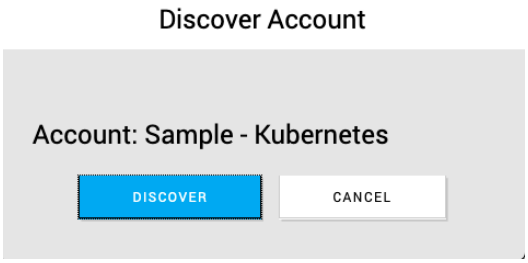
  The Network associated with the pod is fetched form *metadata: annotations: k8s.v1.cni.cncf.io/networks*. It will also show the default-k8s-nw. The interface is fetched from *metadata: annotations:* k8s.v1.cni.cncf.io/networks-status.
- Network: net-attach-def (network definitions)
    - List subnet = This is applicable if the configuration carries ipam dict. This shows the default nw ‘default-k8s-network’ and the attached id, subnet, etc.
- Volume = persistent-volume
- Public-ip = load-balancer’s ingress ip

Note: Audit reports are not supported for Kubernetes accounts in release 8.2.



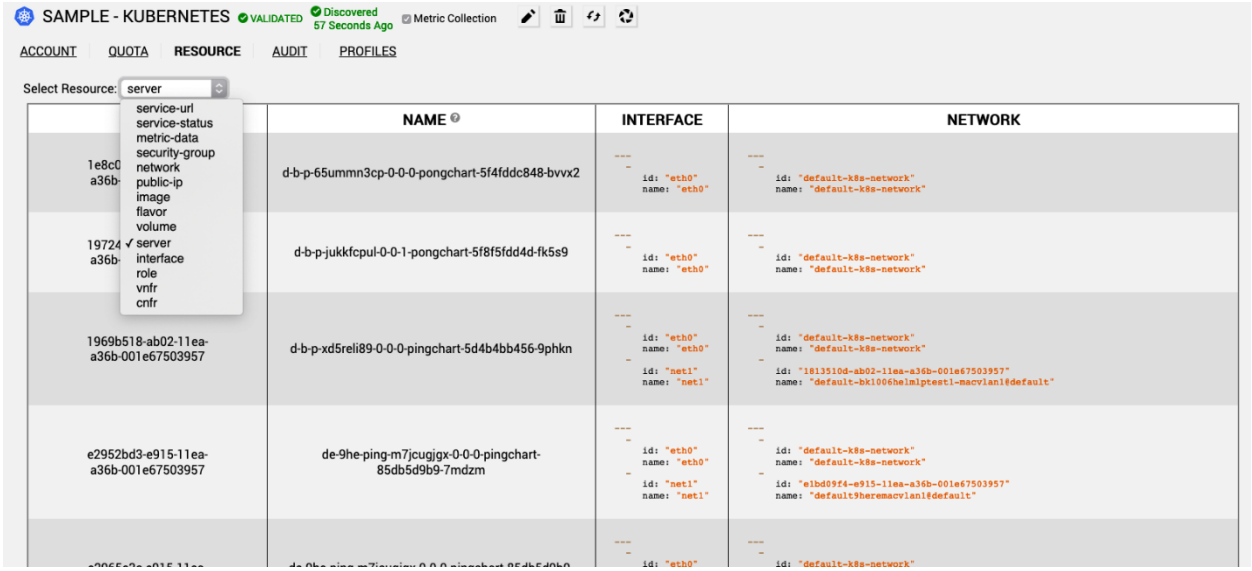
To discover resources in a Kubernetes account:

1. Click  in the **DATACENTERS** > **Account** page. A pop-up screen appears.



2. Click **DISCOVER** and Discovery is initiated.

3. Once Discovery is complete, the status appears at the top of the screen. Go to the **RESOURCE** tab to see all of the discovered resources for the account.



API Enhancements

This release introduces the following model changes:

- The Kubernetes account was previously available in the config path `/rw-project:project[rw-project:name='project']/rw-Kubernetes-account:Kubernetes/rw-Kubernetes-account:account`. In release 8.2, the whole config path is now moved to `/rw-project:project[rw-project:name='project']/rw-cloud:cloud/rw-cloud:account`. The configuration can be accessed using: `show project cloud account`.
- Previously, cloud accounts included the `vdu-instance-timeout` field and Kubernetes accounts included the `cnf-instance-timeout` field. These two fields are now renamed to a single field in cloud account config model. The new field name is `nf-instance-timeout`.
- `Kubernetes-account-types` is renamed to `cloud-account-type`.

New Input Parameter Framework

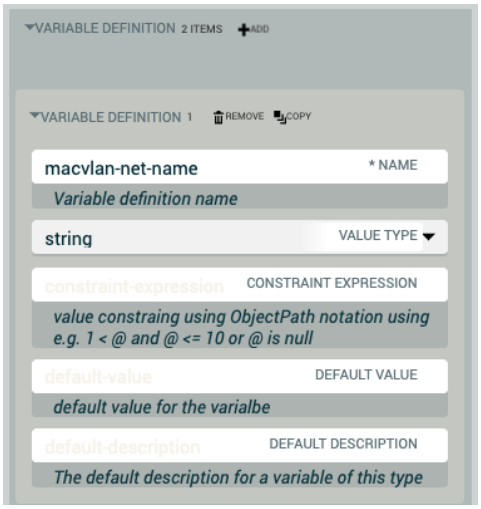
This release introduces a new input parameter framework that an operator can apply to an onboarded helm chart that created a CNFD in the RIFT catalog. Once an operator defines a values file for that CNFD and adds it as an asset in the RIFT.ware CNFD, a list of input parameters (variables-definition and input-parameter) can be specified in the CNFD which can be modified during the Life cycle operation.

Define CNFD Parameters and Instantiate with CNFs

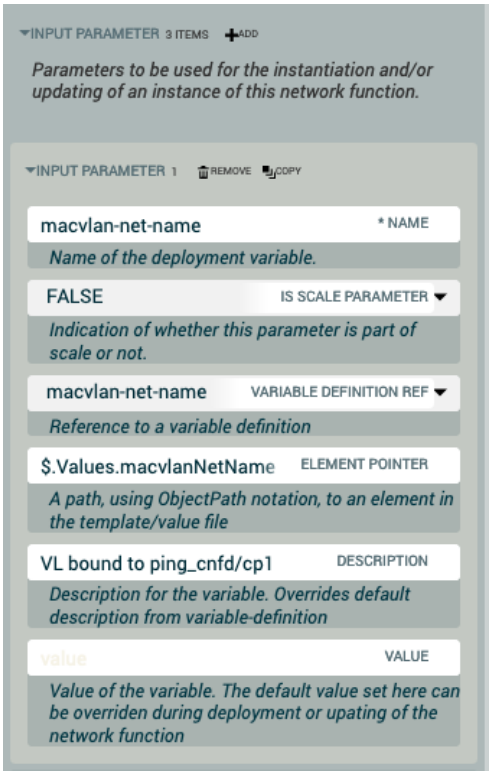
An operator must define the CNFD input parameters. This is a two-step process. First, define a variable-definition entry. Next, create an input-parameter. The variable definitions can be used in more than one input-parameter.

1. On the **Launchpad** menu, click **CATALOG > CNFD LIST** and select the appropriate CNFD descriptor from the catalog. Double-click the descriptor.
2. The **PING_CNFD** or **PONG_CNFD** screen opens.

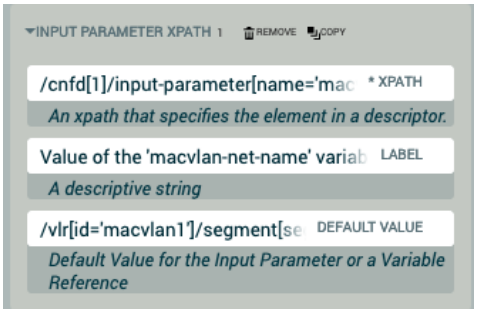
   After onboarding a helm chart, you can define the input-variable required for the CNFD.
3. Navigate to the Variable Definition section, provide the following information:
   - **NAME**: Name of the Variable definition name. Define macvlan-net-name.
   - **VALUE TYPE**: This field indicate whether this parameter is part of scale or not (True or False).
   - **CONSTRAINT EXPRESSION**: This field is a reference to a variable definition.
   - **DEFAULT VALUE**: This field is the default value for the variable.
   - **DEFAULT DESCRIPTION**: This field is the default description of the variable.



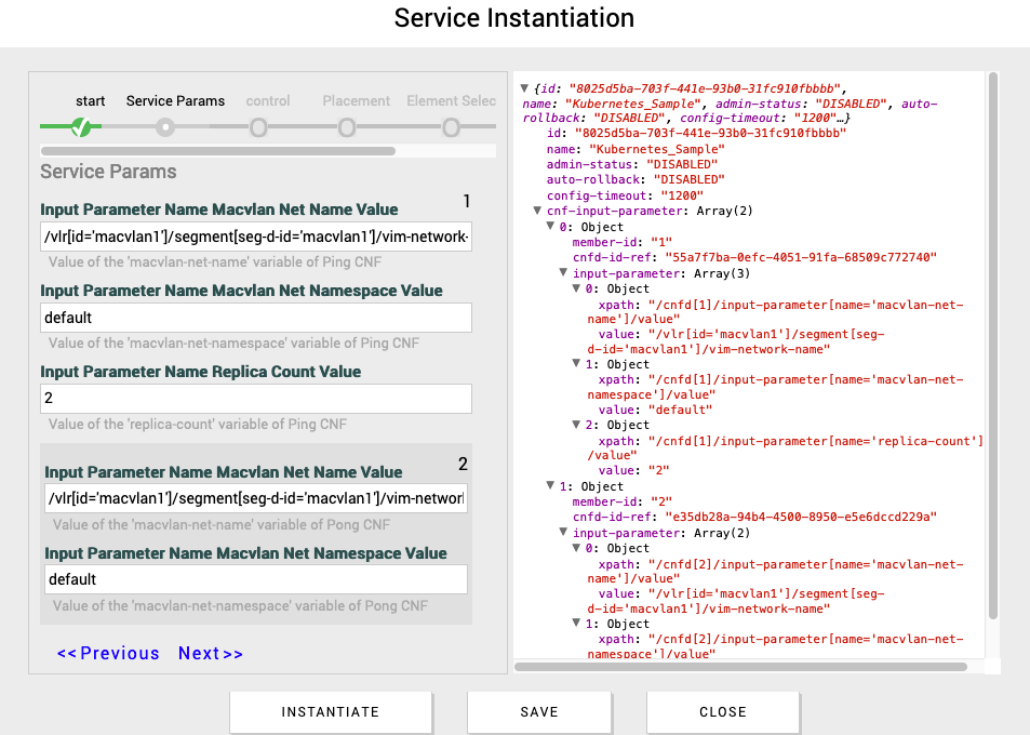
4. Navigate to the input parameter section, provide the following information:
   - **NAME**: Name of the deployment variable.
   - **IS SCALE PARAMETER**: This field indicate whether this parameter is part of scale or not (True or False).
   - **VARIABLE DEFINITION REF**: This field is a reference to a variable definition.
   - **ELEMENT POINTER**: This field is the path, using ObjectPath notation, to an element in the template/value file.
   - **DESCRIPTION**: This field is the description for the variable. This overrides the default description from the variable-definition.
   - **VALUE**: This field is the value of the variable. This default value can be overridden during deployment or when updating the Network Function.



5. Click Update.
6. On the **Launchpad** menu, click **CATALOG > NSD LIST** and select the appropriate NSD descriptor from the catalog. Double-click the descriptor.
7. The **PING_PONG NSD** screen appears. Search for input-parameter-xpath in the navigation pane. In this release, new paths have been added in this section to add the runtime values. These values will appear during NS instantiation or any other LCM operation.
8. In the Input Parameter section, provide the following information:
   - **XPATH**: An xpath that specifies an input-parameter value to be set in a CNFD. For example, /cnfd[ping]/input-parameter[name='macvlan-net-name']/value.
   - **LABEL**: A descriptive string.
   - **DEFAULT VALUE**: Default value for the input parameter or a variable reference.



9. Click Update.
10. Next, click to Instantiate the Service. The Service Params section of the Instantiation Wizard will include the input-parameter-xpath entries that you added when building a service.



11. Click **INSTANTAITE** or **SAVE**.

API Enhancements

This release introduces the following CNFD model changes:

- `input-parameter`: This variable references a variable definition.
- `variable-definition`: This variable is a list of variable definitions.

Launchpad Performance Metrics and Threshold Notifications

This release includes new metrics and SNMP traps related to performance threshold and capacity usage. Launchpad now sends alerts or provides configuration in the UI and the CLI for setting alerts whenever a Launchpad container CPU, Launchpad container memory, RIFT_VAR_ROOT disk or RW.REST request rates exceeds or goes below a particular threshold. RIFT.ware also provides an alert in the CLI whenever active authentication sessions exceed or go below a particular threshold. A user can assign or change a threshold assigned to a parameter that is monitored by Launchpad. This feature also introduces a new model to create these alerts and publish them in RIFT.ware.

UI Enhancements

There is a new REST Metrics widget on the Home page of the UI. A user can view, track and change alerts in the following widgets.

- System CPU
- System Memory
- System Disk Utilization
- Rest Metrics

Note: Manually refresh the widgets to see the most current information.

Each widget has a new CHANGE button on the top right corner to configure the alerts. Only operators with super-admin or platform-admin roles can change the alert metrics. The RIFT.ware Launchpad provides alerts for Memory, RVR, REST and Active authentication sessions by default. When a threshold is reached, an alert is sent. Refer to the Events → System Alarms section of the UI for details about SNMP Traps. From more details about viewing alerts, refer to the Grafana Dashboard (See Grafana Dashboard). Grafana displays the past 15 days of statistics.

The Grafana Dashboard displays the following metrics for this feature.

- **Launchpad Alerts**: The Launchpad (LP) Alert panel shows all the alerts that have been triggered by RIFT.ware. These can include system resource usage, up/down targets, MANO policy invoked or NS auto scaling alerts. An operator can customize this panel to view specific alerts and statistics. For example, a user can view:

  - how long an alert is in the FIRING state.
  - how many times a particular alert is triggered.
  - how much time is takes for an alert to get to the resolved state.
- **Launchpad Core Metrics**: The Launchpad (LP) Core Metris panel has many panels that can be used for LP system resource monitoring. It displays the usage trend for CPU, Memory, Disk, File Descriptors, Threads, etc at the system and process level. This panel also shows some Node level resource usage like CPU and tracked Memory.
- **MANO Metrics**: The MANO Metrics panel tracks many system variables. Some of these variables include the:
  - number of deployed VMs and Pods. This includes both the running and cumulative count.
  - number of NSes that are managed by the orchestration layer. This includes both the running and cumulative count.
  - state of the Lifecycle Management workers in the backend. This panel indicates if the service is online or offline for processing requests.
  - backend queue status and the processing rates.
- **RW.Auth Metrics**: The RW.Auth Metrics panel displays the authentication statistics which is a historical view of the openidc provider stats.
- **RW.REST Metrics**: The RW.Rest Metrics panel displays the request rate, latency, processing rate, request count per verb etc. of HTTP(S) requested received by the RW.REST workers. This panel also displays the detailed statistics per RW.REST worker.

SNMP Alerts

If an operator configures a threshold in the UI, a trap is sent out when the threshold is reached. RIFT.ware sends out an additional clear snmp trap once the threshold comes down. The SNMP table below lists the possible traps for Launchpad Alerts, Launchpad Core Metrics, MANO Metrics and RW.REST Metrics.

SNMP Traps

Trap name	OID	Severity	NOC Action	Additional Fields	Description and action to be taken
riftLPCPU Usage AlertFiring	1.3.6.1.4.1. 53030.1. 2.27	Critical	Contact Operations	riftAlertGroup riftSeverity riftMessage	CPU usage for the Launchpad container exceeded configured threshold.

Trap name	OID	Severity	NOC Action	Additional Fields	Description and action to be taken
					Call Orchestration operations or RIFT Support at 800-914-7438 if the issue persists.
riftLPCPU Usage AlertResolved	1.3.6.1.4.1.53030.1.2.28	Info	Clear	riftAlertGroup riftSeverity riftMessage	CPU usage has come down below the threshold after going up. No action required.
riftLPMemory UsageAlert Firing	1.3.6.1.4.1.53030.1.2.29	Critical	Contact Operations	riftAlertGroup riftSeverity riftMessage	The memory usage of the LP container has crossed the configured threshold. Call Orchestration operations or RIFT Support at 800-914-7438 if the issue persists.
riftLPMemory UsageAlert Resolved	1.3.6.1.4.1.53030.1.2.30	Info	Clear	riftAlertGroup riftSeverity riftMessage	Memory usage of the LP container has now come down below the configured threshold. No action required.

Trap name	OID	Severity	NOC Action	Additional Fields	Description and action to be taken
riftLPDisk UsageAlert Firing	1.3.6.1.4.1. 53030.1.2. 31	Critical	Contact Operations	riftAlertGroup riftSeverity riftMessage	The disk volume used for RIFT artifacts has crossed a threshold value or percentage. Call Orchestration operations or RIFT Support at 800-914-7438 if the issue persists.
riftLPDisk UsageAlert Resolved	1.3.6.1.4.1. 53030.1.2. 32	Info	Clear	riftAlertGroup riftSeverity riftMessage	The disk usage for the RIFT artifacts is now below the configured value. No action required.
riftLPRestRPS AlertFiring	1.3.6.1.4.1. 53030.1.2. 33	Major	Contact Operations	riftAlertGroup riftSeverity riftMessage	The API request rate has exceeded the configured threshold value. Call Orchestration operations or RIFT Support at 800-914-7438 if the issue persists.
riftLPRestRPS AlertResolved	1.3.6.1.4.1. 53030.1.2.	Info	Clear	riftAlertGroup riftSeverity	The API request rate

Trap name	OID	Severity	NOC Action	Additional Fields	Description and action to be taken
	34			riftMessage	has come down below the threshold value. No action required.
riftLPLicense AlertFiring	1.3.6.1.4.1. 53030.1.2. 35	Critical	Contact Operations	riftAlertGroup riftSeverity riftMessage	License firing alert raised because LP VMs or Pods crossed the threshold limit. Call Orchestration operations or RIFT Support at 800-914-7438 if the issue persists.
riftLPLicense AlertResolved	1.3.6.1.4.1. 53030.1.2. 36	Info	Clear	riftAlertGroup riftSeverity riftMessage	Instances of VMs/Pods managed by RIFT is now below the configured threshold value. No action required.
riftAuthActive SessFiring	1.3.6.1.4.1. 53030.1.2. 37	Critical	Contact Operations	riftAlertGroup riftSeverity riftMessage	The number of active authenticated user sessions has exceeded the configured threshold value.

Trap name	OID	Severity	NOC Action	Additional Fields	Description and action to be taken
					Call Orchestration operations or RIFT Support at 800-914-7438 if the issue persists.
riftAuthActiveSessResolved	1.3.6.1.4.1.53030.1.2.38	Info	Clear	riftAlertGroup riftSeverity riftMessage	The authenticated user sessions has come down below the threshold value. No action required.

SNMP Variables Sent with Traps

Variable Name	OID	Type	Description
riftAlertGroupString	1.3.6.1.4.1.53030.1.1.9	String	Alert group as defined by the RIFT.ware Alert Manager. These variables are only applicable for alerts generated by the RIFT.ware Alert Manager. Possible values: • LICENSE • LP-CORE • RW-AUTH • RW-REST • LCM • LCM-POLICY • METRIC-EXPORTER

RW.REST Metrics

SNMP Trap Name	RIFT.ware Notification
riftLPRestRPSAlertFiring	rw-alert-log:rw-rest-rps-alert-firing
riftLPRestRPSAlertResolved	rw-alert-log:rw-rest-rps-alert-resolved

RPC Changes

An operator can configure the below Remote Procedure Call (RPC) commands to add or remove alerts.

```
rpc register-alert {

    description "RPC for alert registration. Applications can register
for alerts based on the
      metric being added to Prometheus.";

    input {
      leaf group-name {
        description "The group under which the particular class of
alert belongs to.";
        type rw-alert:group-type;
        mandatory true;
      }

      leaf alert-rule-name {
        description "The alert rule name. Should signify what kind of
alert is being evaluated.";
        type string;
        mandatory true;
      }

      leaf alert-expression {
        description "The alert expression that needs to be evaluated";
        type string;
        mandatory true;
      }

      leaf hysteresis-time {
        description "Time to wait for the alert expression to be true
over the time period.";
        type string {
          pattern '[0-9]?[m|s]';
        }
        mandatory true;
      }

      leaf annotation-summary {
        description "Message which will come along with the alert.
          Can container template variables to get alert specific
message.";
```

```
        type string;
        mandatory true;
      }
    }

    output {
      leaf status {
        description "Either success or failure. Tells the status of
 the alert registration.";
        type string;
      }

      leaf error {
        description "Additional error details if the status was
 failure.";
        type string;
      }
    }
  }
```

Model Changes

The new model is created from the above RPC. This model sets alert configurations.

```
typedef group-type {
    description "Supported groups under which alerts can be created";

    type enumeration {
      enum LCM;
      enum LP-CORE;
      enum METRIC-EXPORTER;
    }
  }

  typedef alert-type {
    description "An alert is in either FIRING state or RESOLVED state
      when it reached RIFT.ware.



    type enumeration {
      enum FIRING;
      enum RESOLVED;
    }
  }

  list alert-group-config {
    description "Alert rules are grouped under a unique group name.
      Multiple rules can thus be logically coupled for a single
component.";

    key "group-name";

    leaf group-name {
      description "The component for which the alert rules are set";
      type group-type;
    }
```

```
    list alert-rule {
      description "The alerts conditions or rules that needs to be
set";
      key "rule-name";

      leaf rule-name {
        description "Rule identification name. Should reflect the kind
of alert it triggers.";
        type string;
      }

      leaf rule-expression {
        description "PromQL expression for evaluating the alert
trigger";
        type string;
      }

      leaf hysteresis-time {
        description "Time to wait for the alert expression to be true
over the time period.";
        type string {
          pattern '[0-9]?[m|s]';
        }
      }

      container annotation {
        leaf summary {
          description "Message which will come along with the alert.";
          type string;
        }
      }
    } // END alert-rule
  } // END alert-group-config
```

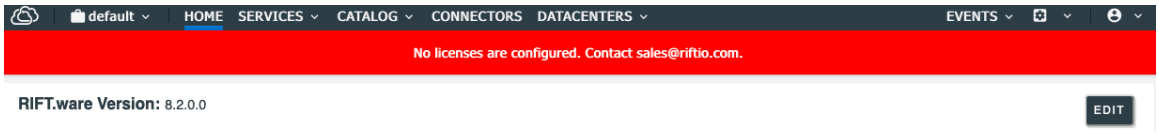
Licensing Management Enhancements

Release 8.2 introduces an updated method for configuring and managing RIFT.ware licenses. An operator can install and partially configure RIFT.ware without a license key. However, a valid license key is now required to perform any Lifecycle management (LCM) functions such as onboarding and instantiating a service.

- RIFT.ware Licensing in Release 8.2
- RIFT.ware Licensing Widget

RIFT.ware Licensing in Release 8.2

If Launchpad is not registered, then a red warning banner appears at the top of the RIFT.ware UI. The job tracker will also show that an action fails if there is an attempt to perform any LCM functions without a valid license.



Note: If a license key expires, then the existing Network Services will continue to run. However, all Lifecycle Management operators are disabled until there is a valid license key associated with the RIFT.ware instance.

There must be one license key per each instance of RIFT.ware. A RIFT.ware instance is defined as either a single Virtual Machine (VM) or a cluster of VMs (if this option is supported) that forms a single RIFT.ware instance. Operators obtain a valid license key from RIFT support and then enter the license using the Launchpad Dashboard. For license information, see Services & Support or contact sales@riftio.com.

If there is no valid license key for a RIFT.ware instance, then it is not possible to enable functionality for Network Slicing and/or Geographic Redundancy. To enable Geographic Redundancy in release 8.2, RIFT.ware Launchpad is installed at two sites as separate instances and then the sites are configured as backups of each other. Each version of RIFT.ware Launchpad is considered a separate instance of Launchpad and must be licensed separately. You must add both licenses on each Launchpad.

This feature also allows users to monitor license capacities (such as host, server and VMs/Pods) by setting a threshold limit as a percentage. Once the percentage is set, RIFT.ware raises one of the following alerts if a threshold limit is reached.

SNMP Trap Name	RIFT.ware Notification
riftLPLicenseAlertFiring	license-usage-alert-firing
riftLPLicenseAlertResolved	license-usage-alert-resolved

Note: An operator must have the platform-admin role (`rw-rbac-platform:super-admin`) to configure a RIFT.ware license.

RIFT.ware Licensing Widget

There is a new **RIFT.ware Licensing** widget on the Home page of the UI that allows customers and RIFT support to easily access licensing information. If you have any questions about entering a license provided by RIFT support, then see License Management for more information.

This widget includes the following data:

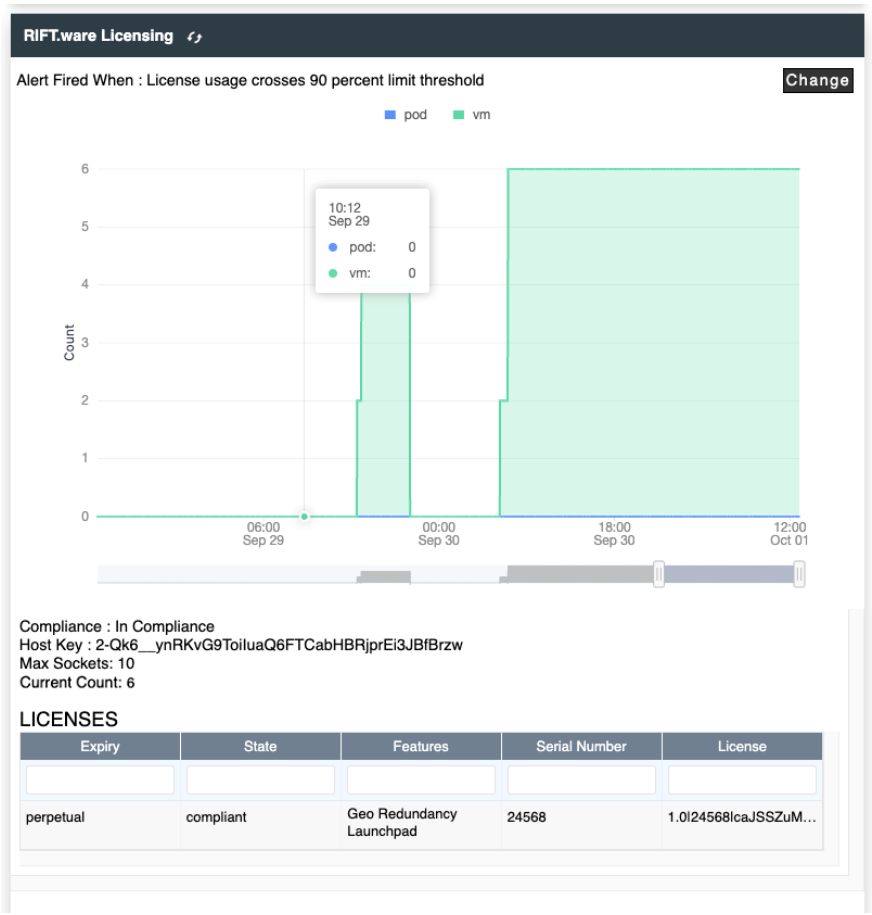
- **Recorded values collected periodically**: A dynamic chart displays this data. You can scroll to expand or reduce the timeframe.
- **Compliance**: The compliance status of the license (unlicensed or in-compliance).
- **Host key**: Customer provides a host key to RIFT when requesting a license. The host key uniquely identifies an instance.
- **Max Sockets or Max Servers**: If you generate a socket or server license, then the UI displays the maximum number of sockets or servers under management.
- **Current Count**: The current number of VMs/Pods, sockets or servers under management.

  Note: The RIFT.ware Licensing graph only displays VMs/Pods for the current count statistics.

- **LICENSES**: The UI displays the features associated with each license and any limits imposed on that license. Prior to release 8.2, the list of installed licenses were displayed on the About page in the UI.
  - **Expiry**: The date when the license expires.
  - **State**: The state of the license (licensed or compliant).
  - **Features**: The features supported with each license.
  - **Serial Number**: The serial number for the license.
  - **License**: The license key.

Note: If your license includes a capacity limit, then it will appear on the RIFT.ware Licensing Dashboard.

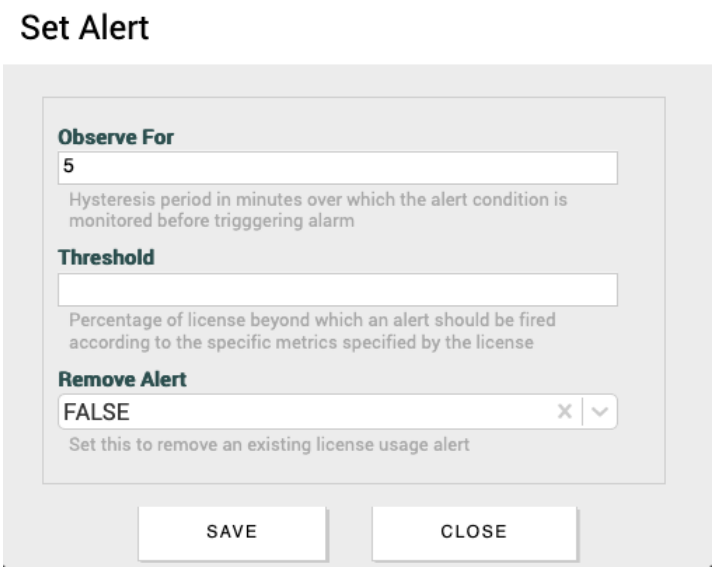
If a valid license is entered correctly, then the product will be licensed. Verify the licensing status in the **HOME** > **RIFT.ware Licensing** > **Compliance** section. If you encounter an error when attempting to add a license, that error condition will be identified under State in the **RIFT.ware Licensing** widget and in the **Events** > **License** section of the UI. Contact RIFT if you require assistance.



There is also a change button in the top right corner of the **RIFT.ware Licensing** widget. Click **CHANGE** to configure the thresholds and provide the **Set Alert** details in the pop-up screen.

Note: An operator must have the platform-admin or platform-oper role (`rw-rbac-platform:platform-admin` or `rw-bac-platform:platform-oper-role`) to configure licensing thresholds.

- **Observe For**: The hysteresis period in minutes over which the alert condition is monitored before triggering an alarm.
- **Threshold**: The percentage of licenses beyond which an alert should be fired according to the specific metrics specified by the license.
- **Remove alert**: This alert allows a user to remove an existing license usage alert (False or True).



Check your entries and click **SAVE**.

Multi-Cloud Orchestration

In release 8.2., RIFT.ware now supports hybrid Network Service functionality which allows an operator to deploy a Network Service in a number of ways. Our architecture now supports multiple interconnected Network Functions (NFs) that are with a mix of CNFs and VNFs and they can be in a private or a public cloud.

Multi-cloud Orchestration Workflow

RIFT.ware's multi-cloud orchestration enables end-to-end deployment of hybrid cloud and multi-cloud Network Services. These enhancements provide designers numerous options when building a service. An operator can now launch a service with:

- CNFs in a Mixture of Kubernetes Clusters
- VNFs on Multiple VIMs with two Different Tenants
- VNFs on VIMs of Different Types of Accounts
- CNFs and VNFs in a Combination of VIMs and Kubernetes Clusters

Cloud Account Profiles

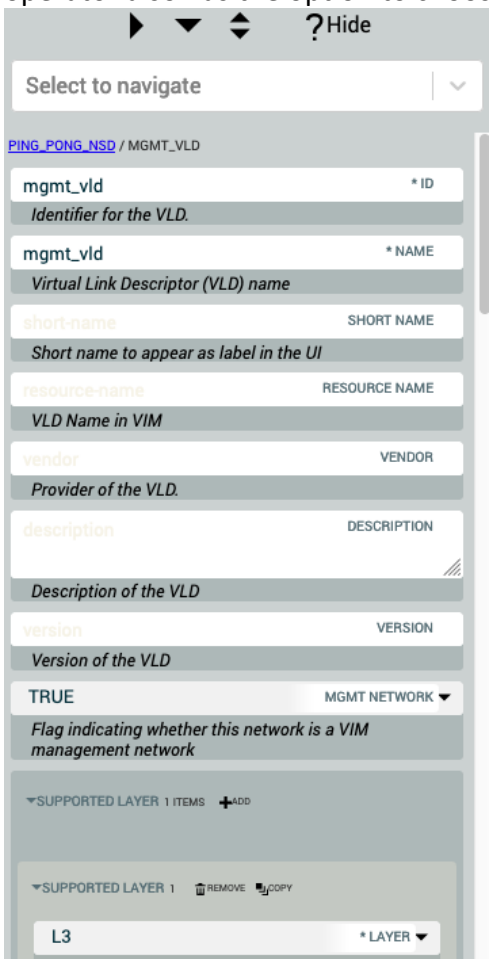
In release 8.2, an operator must define cloud account profiles by adding segment and IP profiles for each cloud account. Previously, an operator provided the vim-network-name as part of the VLD. It is now part of the segment in the datacenter and the VLD information is populated with the segment reference.

The VLD must contain at least one segment with a segment profile defined. Each VL in the NS must be associated with a segment profile in the VLD, During instantiation, a new segment profile can be picked. However, all Datacenters used in the NS must share the same segment profile for a given segment.

See New Profiles section in the DATACENTER tab in the RIFT.ware Instantiation Enhancements section. After adding the required ip profiles and segment profiles, then it is possible to instantiate the service.

Instantiation Workflow Updates

In this release, you'll find changes to the VLD section when building your service. An operator also has the option to choose the Datacenter to launch a service.



VLD Changes

- **ID***: Identifier for the VLD
- **Name***: Virtual Link Descriptor (VLD) name
- **SHORT NAME**: Short name to appear as a label in the UI.
- **RESOURCE NAME**: VLD Name in the VIM.
- **VENDOR**: Provider of the VLD.
- **DESCRIPTION**: Description of the VLD.
- **VERSION**: Version of the VLD.
- **MGMT NETWORK**: Indicates whether this network is a VIM management network.

L3 CONNECTIVITY

- **ID**: Identifier for the L3 network.
- **DESCRIPTION**: Name of the L3 network.
- **FLOW TYPE**: Flow pattern in this layer.

IP PROFILE PARAMS

SUBNET 1

- **NAME***: Name of the subnet
- **IP VERSION**
- **SUBNET TYPE**: Either define the subnet or refer to an already defined subnet
- **GATEWAY ADDRESS**: IP address of the default gateway associated with the IP Profile.

DHCP PARAMS

- **ENABLED**: TRUE or FALSE
- **START ADDRESS**: Start IP Address of the IP Address range associated with the DHCP domain
- **END ADDRESS**: End IP address of the IP Address range associated with the DHCP domain
- **COUNT**: Size of the DHCP pool associated with the DHCP domain

SEGMENT

- SEGMENT 1
- **ID***: Identifier for the L2 segment.
- **DESCRIPTION**: Description of L2 segment.
- **FLOW TYPE**: Flow pattern in this layer.

L2 ATTRIBUTES

- **ROOT BANDWIDTH**: Aggregate Bandwidth
- **LEAF BANDWIDTH**: Bandwidth of branches

SEGMENT PROFILE REF: Segment profile ref in the datacenter

- **VIRTUAL CONNECTION POINTS**: A list of virtual-connection points associated with Virtual Links. These connection points are not directly associated with any CNFs or VNFs
- **CNFD or VNFD CONNECTION POINTS**: A list of references to the CNFD or VNFD connection points.

CNFD and VNFD CONNECTION POINT REF

- **MEMBER VNF INDEX REF**: Reference to the member cnf or vnf within the constituent-cnfds/vnfds
- **CNFD/VNFD ID REF**: A reference to the CNFD or VNFD.
- **CNFD/VNFD CONNECTION POINT REF**: A reference to a connection point name.
- **SEGMENT REF**: The segment reference attached to the L2 segment.

Note: The supported layer field does not require any information in release 8.2.

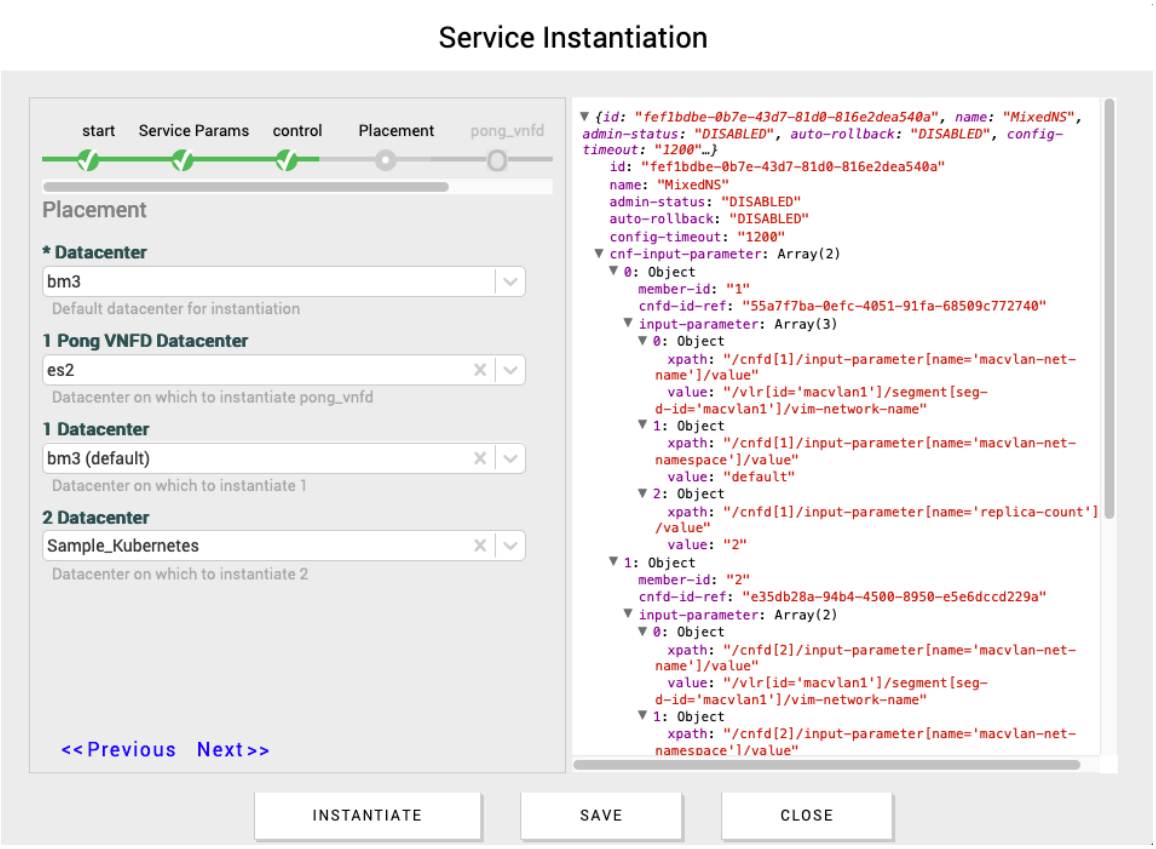
In order to instantiate a service in this release, one segment is mandatory. The datacenter must be provisioned with the segment profile that corresponds to the segment profile referenced in the NSD. Then, the segment can connect to the CNFD or VNFD connection points. The Service Instantiation allows users to select the datacenter to host the NFs. The UI will try to define your default datacenter at the NS level and attempt to assign them to a CNFD or VNFD. The UI also allows you to modify this section. Next, the UI attempts to automatically pick the virtual link segment profiles for each of the segments but you can edit this section as well.

UI Enhancements

Release 8.2 introduces new requirements to instantiate your service. In the **DATACENTERS** page, you can add ip-profile details to describe all of the IP characteristics for the Virtual-Link. Next, you must add one segment profile for the NF. See New Profiles section in the DATACENTER tab for steps on adding ip-profiles and segment-profiles. This release also includes changes to the steps to map pre-existing VLDs. Previously, the UI allowed a user to map pre-existing VLDs in the Instantiation Wizard. This step is deprecated in release 8.2. The UI now includes a datacenter definitions for ip-profiles and segment-profiles in the Instantiation Wizard. See VLDs Section Changes . The Wizard also allows an operator to choose or change a datacenter while instantiating, scaling or updating a service.

Datacenter

After adding your ip-profile and segment-profile, you can pick the datacenter where you want to launch your service.



An operator can now choose how to launch a service in the UI.

- CNFs in Multiple Kubernetes Clusters
- VNFs on Multiple VIMs with two Different Datacenters
- VNFs on VIMs of Different Types of Accounts
- CNFs and VNFs in a Combination of VIMs and Kubernetes Clusters

CNFs in Multiple Kubernetes Clusters
An operator can create multiple Kubernetes cluster accounts and then instantiate each CNF in the same or different clusters.

VNFs on Multiple VIMs with two Different Datacenters
An operator can create multiple Openstack tenants and instantiate each VNF on different tenant.

VNFs on VIMs of Different Types of Accounts
An operator can create different types of VIM accounts (i.e.: an OpenStack account and an AWS EKS account) and the instantiate a service.

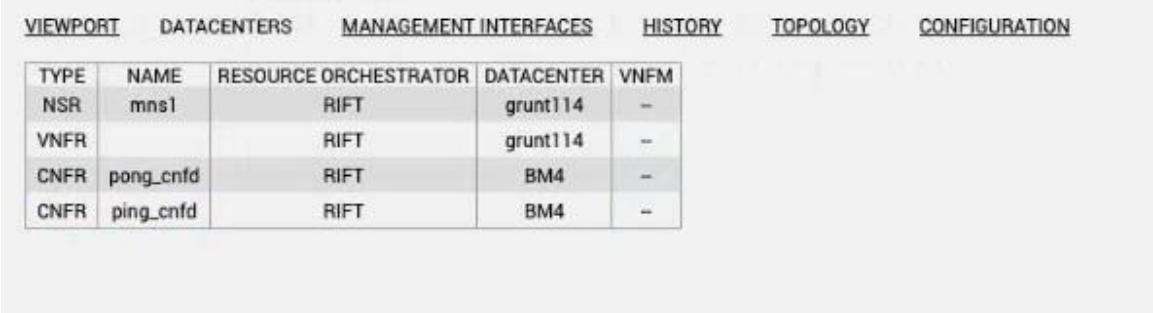
CNFs and VNFs in a Combination of VIMs and Kubernetes Clusters
An operator can create a VIM account and a Kubernetes account and instantiate the VNF on the VIM account and the CNF on the Kubernetes account.

This feature also introduces the ability to scale out or scale in a service using any of the above workflows.

After NS Instantiation

Once a service is running, then the **VIEWPORT → DATACENTERS** page lists all of the datacenters that you have assigned in your service. The **VIEWPORT→ Virtual links** tab now has a list of all the associated segments.

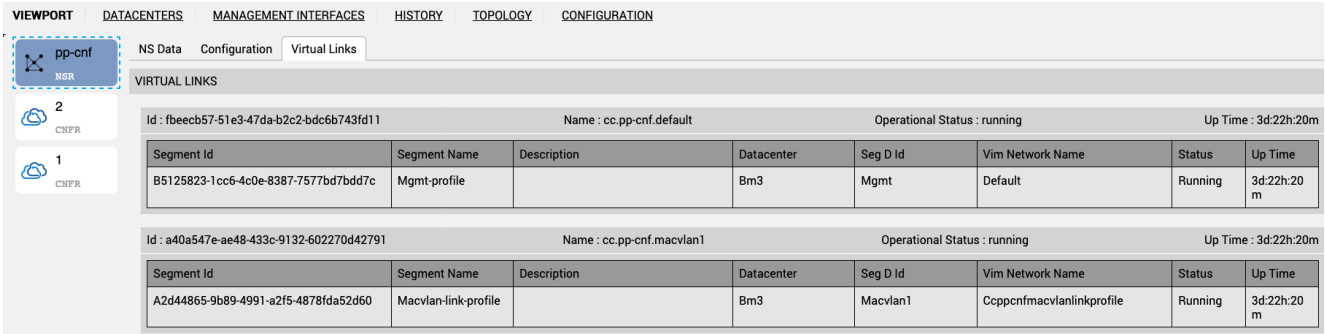
DATACENTERS



TYPE	NAME	RESOURCE ORCHESTRATOR	DATACENTER	VNFM
NSR	mns1	RIFT	grunt114	–
VNFR		RIFT	grunt114	–
CNFR	pong_cnfd	RIFT	BM4	–
CNFR	ping_cnfd	RIFT	BM4	–

Virtual Links

The updated virtual links page now shows a list of segments for each virtual link.



API Enhancements

- NSD Model changes
- VLR Model Changes
- Cloud Account Model Changes

NSD Model Changes

Added the following new fields to the nsd:nsd-catalog in the nsd:vld Data Model.

- nsd:supported-layer: Layers supported by the VLD
- nsd:l3-connectivity: This field contains the ip-profile-params field to configure the ip profiles in the datacenter.
    - nsd:ip-profile-params
        - nsd:subnet
            - nsd:dhcp-params
            - nsd:dns-server

- nsd:segment: This field allows an operator to configure the segment profile in the datacenter.
    - nsd:l2-attributes
    - nsd:virtual-connection-points: A list of virtual-connection points associated with Virtual Link. These connection points are not directly associated with any VNFs
        - nsd:associated-cps: A List of connection points associated with virtual connection point
- nsd:vnfd-connection-point-ref: A list of references to VNFD connection points.
- nsd:cnfd-connection-point-ref: A list of references to CNFD connection points.

VLR Model Changes

Added the following new fields to the vlr:vlr-catalog in the vlr:vlr Data Model.

- vlr:supported-layer: Layers supported by the VLD
- vlr:segment: This field allows an operator to configure the segment profile in the datacenter.
    - vlr:ip-profile-params
        - vlr:subnet
            - vlr:dhcp-params
            - vlr:dns-server
    - vlr:provider-network: Container for the provider network.
    - vlr:subnets
        - vlr:dns-servers
        - vlr:ip-pools
    - rw-vlr:virtual-connection-points
        - rw-vlr:associated-cps
- vlr:l3-connectivity: This field contains the ip-profile-params field to configure the ip profiles in the datacenter.
    - vlr:subnets
        - vlr:dns-servers
        - vlr:ip-pools
- vlr:ip-profile-params
    - vlr:subnet
        - vlr:dhcp-params
        - vlr:dns-server

Cloud Account Model Changes

Added the following new fields in the rw-project:project in the rw-cloud:cloud Data Model.

- rw-cloud:ip-profiles: This is the list of ip profiles. The ip profiles describe the IP characteristics for the Virtual-Link.
    - rw-cloud:ip-profile-params
        - rw-cloud:subnet
            - rw-cloud:dhcp-params

- rw-cloud:dns-server
- rw-cloud:segment-profile: The list of network segment profiles in the cloud.
  - rw-cloud:provider-network: Container for the provider network.

RIFT.ware Installation and Upgrade Enhancements

This release introduces changes to the RIFT.ware installation and upgrade processes. Prior to release 8.2, RIFT supported Launchpad (LP) deployment both as a VM qcow2 image and in Kubernetes using a Helm Chart. Starting in release 8.2, LP is deployed in a Kubernetes based Cluster k8s or Kubernetes k3s in a VM platform.

Install Launchpad in a Kubernetes Based Cluster k8s

Prepare for Installing LP in a Kubernetes Based Cluster k8s

RIFT.ware requires the following prerequisites to install Launchpad in a Kubernetes based Cluster k8s.

- **Helm Chart**: Build a helm chart for RIFT.ware Launchpad. Enter the CLI from a remote machine using SSH or on the platform itself. See the Helm Chart section below for steps to install the chart. For details on how to create a chart see The Chart Template Developer's Guide.
- **Values File**: The values file is a template that an operator must customize. The values can be extracted from the helm chart (values.yaml). For details on how to create and edit a values file, see the helm Values Files documentation.

> **Note:** The values file template is embedded directly into the RIFT code.

- **Docker Access**: For docker access, contact RIFT Services and Support at support@riftio.com to obtain access to a docker registry with docker images.
- **Working Cluster and Kubeconfig File**: A working cluster and kubeconfig file for the target Kubernetes cluster. For additional details, contact your Cluster Admin.

- **Cluster Details**: An operator must understand all possible configurations of the cluster prior to installing RIFT.ware.
  See a couple questions below that an operator must ask prior to installation.
  - Does your cluster use ceph for storage? For ceph, you can make this change in the values file: storageClassName: ocs-storagecluster-cephfs
  - Does it have an HA proxy, NodePort and/or MetaLB?

Helm Chart
It is necessary to deploy a help chart prior to installing RIFT.ware.

Note: An operator must have a Kubernetes Cluster Admin or Operator role with the proper permissions to install helm charts. For additional details, contact your Cluster Admin.

rift-shell is a command line user interface used to set up environmental variables, such as `PATH` and `RIFT_ROOT`, necessary to run and develop RIFT.ware software. The environmental variables are inferred from the code base and are written to:

```
${RIFT_ROOT}/.build/.rift-env
```

You do not interact with RIFT.ware from rift-shell. You interact with RIFT.ware through the RIFT UI or the RIFT CLI (RW.CLI).

1. To enter the RIFT.ware CLI (rwcli) in a new terminal, run the following series of commands:

   ```
   $ cd /usr/rift
   $ sudo -H ./rift-shell -r -i /usr/rift -a /usr/rift/.artifacts
   $ rwcli
   ```

   The NETCONF username is `admin`. Contact RIFT Services and Support at support@riftio.com to obtain the proper password.

   Note: You can safely ignore any errors output to the terminal, as long as you see the rift# prompt.

2. Run the following commands to install the helm chart:

   ```
   $ helm3 install <release-name> <path-to-helm-package> <custom-
   values>

   eg> helm install my-lp ./launchpad-8.2.0.tgz --set
   global.namespace.name=myns-lp --set
   launchpad.service.type=LoadBalancer
   ```

```
eg> helm install my-lp https://artifactory.riftio.com/helm-
local/launchpad/launchpad-8.2.0.tgz -f custom_values.yaml
```

This command will install the Launchpad as a Pod in the Kubernetes cluster. Persistent volume is required for RIFT_VAR_ROOT and by default it is mounted using Network Functions (NFs). The NFs server and mount path is specified in the <package>/values.yaml file. See the Values Required to Install LP in a Kubernetes Based Cluster k8s table below.

Note: Values can be customized using '-set' override or a custom values.yaml file (-f option) when executing the chart install.

Prior to installing the helm chart, an operator can test the chart to check for errors in the package using the following command:

```
$ helm3 install --dry-run <release-name> <path-to-helm-package>
```

Cluster Details

The recommended configuration for deploying RIFT.ware LP in a Kubernetes Based Cluster k8 is:

- 16 VCPUs
- 32 GB RAM
- 150 GB Disk

These resources need to be on only 1 or 2 nodes. If there are 2 nodes, then the requirements are:

Node 1

- 12 VCPUs
- 24 GB RAM
- 100 GB Disk

Node 2

- 4 VCPUs
- 8 GB RAM
- 50 GB Disk

Install LP in a Kubernetes Based Cluster k8s

1. Customize the values file as required for a specific installation. The details are embed as comments in the sample values file inside the helm chart. The values file template is constantly changing but there are some values that must be customized since they are required to install RIFT.ware. The mandatory values are listed in the table below.

Values Required to Install LP in a Kubernetes Based Cluster k8s	
Value	**Setting**
launchpad.service.type	This value sets the Kubernetes service type for the Launchpad. **NodePort**: When installed in NodePort mode, external facing RW.UI/RW.REST services are assigned a port >30000. These ports should be used for accessing RW.UI/RW.REST from outside the Kubernetes cluster. See NodePort. **LoadBalancer**: When installed in LoadBalancer mode, the external IP used for access will point to an external LoadBalancer IP. The usual ports can be used for this option. See LoadBalancer. **Default Value**: NodePort
externalAddress	This is a hostname or IP address of Launchpad that is externally accessible. In a Kubernetes cluster, any node in a cluster can be set to this value. Once this value is set, you must use the same address to access RW.UI. RW.REST. Other services continue to be accessed using other node names. **Note:** If you install Launchpad in LoadBalancer mode, then this value is not required.
global.namespace.name	This is the Namespace where the Launchpad is installed. **Default Namespace**: rift-lp To share multiple Launchpad installations in the same Kubernetes cluster, change the namespace. This will also require a different nfs mount.
launchpad.image. repository	The docker image for Launchpad. The version is specified in the Chart.yaml:appVersion. Contact RIFT Services and Support at support@riftio.com for the docker image.
storage.nfs.mountPath	This is the path to mount the NFs. The NFs server is specified using the storage.nfs.server parameter. To disable nfs and use the Kubernetes cluster's default storage class, set storage.nfs to null To run multiple Launchpads on the same Kubernetes cluster, specify a mount path not used by other Launchpad services.

2. Run the following command to find and fix any error prior to installing LP.
```
helm install LP launchpad.tgz -f values.yaml --dry-run'''
```

3. Next, run the below command to install Launchpad.
```
helm install LP launchpad.tgz -f values.yaml '''
```

4. Verify access to Launchpad. See NodePort and LoadBalancer for the specific commands to check your access.

During this installation process, the RIFT.ware UI and other Launchpad components are installed during the LP Helm chart install.

NodePort

By default, Launchpad is installed as a service using the Kubernetes NodePort option. If you choose this option, then Launchpad services can only be accessed externally using the NodePort ports (ports >30000 assigned by k8s) and not the default ports. You can pre-select port numbers in the values file. If you do not configure port numbers in the values file, then RIFT.ware will automatically assign you port numbers. Port numbers are necessary so that LP can create correctly formatted HTTP redirects.

To get the service ports execute the following command.

```
$ kubectl get service -n rift-lp
NAME           TYPE       CLUSTER-IP     EXTERNAL-
IP   PORT(S)                                        AGE
lp-
launchpad   NodePort   10.233.28.79  <none>        8009:31317/TCP,8008:3
1943/TCP,8443:32515/TCP   3m1s
```

In this mode, the externalAddress in the values.yaml is mandatory. This externalAddress can be any one of the IPs/FQDN of the Kubernetes cluster nodes in which the Launchpad is deployed. In the above installation example, the UI can be accessed using the port 32515, REST with port 31943 and OAuth2 service using port 31317. The RIFT.ware UI can be accessed using the url https://<externalAddress-in-values.yaml>:32515.

LoadBalancer

When Launchpad is installed in LoadBalancer mode, the Kubernetes controllers allocate a Load Balance with external access that routes messages only to the Launchpad service. Launchpad service can be accessed using the IP address allocated for the Load Balancer. If there is a LoadBalancer acting as an ingress controller, then you can configure port numbers in the values file. In this scenario, the externalAddress points to the LoadBalancer. Port numbers are necessary so that LP can create correctly formatted HTTP redirects.

This Load Balancer external IP can be obtained by using the below command. The external IP column specifics the address assigned to the Load Balancer. Launched can be accessed using the external-ip address or an FQDN pointing to it.

```
$ kubectl get service lp-a-launchpad -n site-a
NAME              TYPE           CLUSTER-IP     EXTERNAL-
IP   PORT(S)                                        AGE
lp-a-
launchpad   LoadBalancer   10.233.16.37  10.67.1.220  8009:32660/TCP,800
8:32161/TCP,8443:32240/TCP   17h
```

In this mode the externalAddress in the values.yaml file is optional. If this address is not specified, then the helm chart automatically takes the EXTERNAL-IP for the LP service. In the above installation example, use URL https://10.67.1.220:8443 to access the LP UI and use URL https://10.67.1.220:8008 to use REST.

Uninstall LP in a Kubernetes Based Cluster k8s

To uninstall LP in a Kubernetes Based Cluster k8s, run the following command.

```
helm uninstall my-LP
```

Upgrade LP in a Kubernetes Based Cluster k8s

- Prepare for Upgrading LP in a Kubernetes Based Cluster k8s
- Upgrade LP in a Kubernetes Based Cluster k8s

Prepare for Upgrading LP in a Kubernetes Based Cluster k8s

RIFT.ware requires the following prerequisites to upgrade Launchpad in a Kubernetes Based Cluster k8s.

- **Helm Chart**: Build a new and/or updated helm chart for RIFT.ware Launchpad. Enter the CLI from a remote machine using SSH or on the platform itself. See the Helm Chart section below for steps to install the chart. For details on how to create a chart see The Chart Template Developer's Guide.
- **Values File**: Customize the launchpad.upgradeForm value. If you have more new or updated fields, then make additional updates to the values file template. The values can be extracted from the helm chart (values.yaml). For details on how to create and edit a values file, see the helm Values Files documentation.

  Note: The values file is embedded directly into the RIFT code.

- **Docker Access**: For docker access, contact RIFT Services and Support at support@riftio.com to obtain access to a docker registry with docker images.
- **Kubeconfig File**: This is the same kubeconfig file that you used to initially install Launchpad.

Upgrade LP in a Kubernetes Based Cluster k8s

An operator can upgrade from one version of RIFT.ware to another version. First, customize the launchpad.upgradeForm field in the values file and then run the following command to complete the helm LP upgrade.

```
helm upgrade <release-name> <chart-version> <flags>
```

Values Required to Upgrade LP in a Kubernetes Based Cluster k8s	
Value	**Setting**
launchpad.upgradeFrom	The deployed launchpad image version. The image version is on the RIFT.ware login page or you can find it by running the helm history <release-name> command. This value is mandatory to upgrade LP.

Run the following commands to verify the upgrade. Confirm the status of the upgrade from the output.

```
helm history <release-name>
```

Install Launchpad in a Kubernetes k3s in a VM platform

This process is similar to the installation process in previous releases.
Starting in release 8.2, you must run the required commands to install RIFT.ware inside a VM platform. RIFT provides a QCOW2 image that contains a fully-populated version of the RIFT.ware Launchpad. RIFT.ware is ready to use as soon as you instantiate the image on virsh or OpenStack Horizon.

In order to install LP using this method, you must have an SSK Key and you must set the `RIFT_EXTERNAL_ADDRESS`. The `RIFT_EXTERNAL_ADDRESS` is the hostname or the IP address of Launchpad that is externally accessible to the operator. Use the following command to update the `RIFT_EXTERNAL_ADDRESS`.

```
/usr/rift/var/rift/env.d/RIFT_EXTERNAL_ADDRESS
```

Upgrade Launchpad in a Kubernetes k3s in a VM platform

An operator can upgrade from one version of RIFT.ware to another version. Run the following commands to complete the upgrade. This must be done from inside the VM.

```
helm upgrade <release-name> <chart-version> <flags>
```

Run the following commands to verify the upgrade. Confirm the status of the upgrade from the output.

```
helm history <release-name>
```

RIFT.ware Instantiation Enhancements

This feature includes several changes to the instantiation process in release 8.2

New Profiles section in the DATACENTER tab

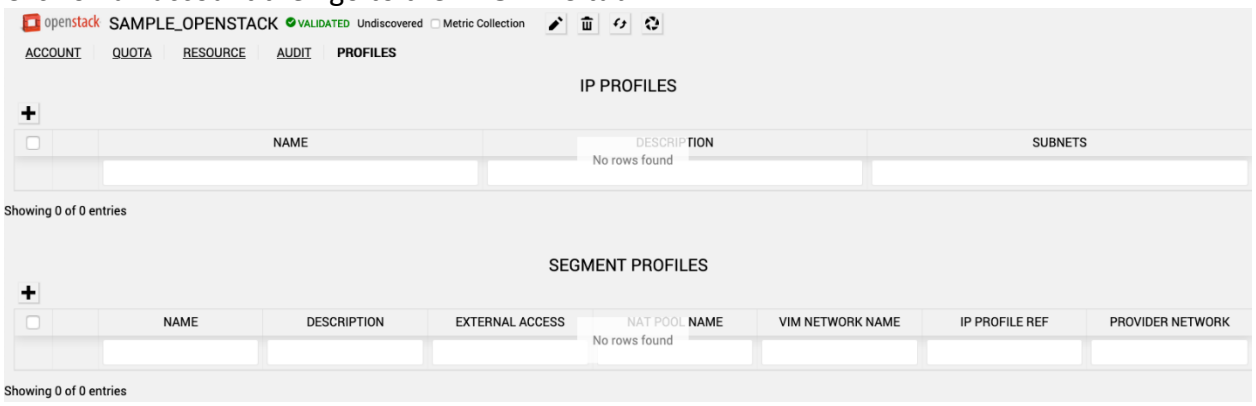
The **DATACENTERS** page now includes a **PROFILES** section to add **IP PROFILES** and **SEGMENT PROFILES** for each account. This new profile configuration simplifies the service instantiation VLD configuration step.

The **IP PROFILES** section describes all the ip characteristics for the virtual link. Depending on how you configure your segment profile, you may need to add an ip profile.

The **SEGMENT PROFILES** section allows an operator to define a network configuration that will be available on the datacenter. The segment profile may reference a previously created ip profile. It is necessary to add a segment profile for each data center account.

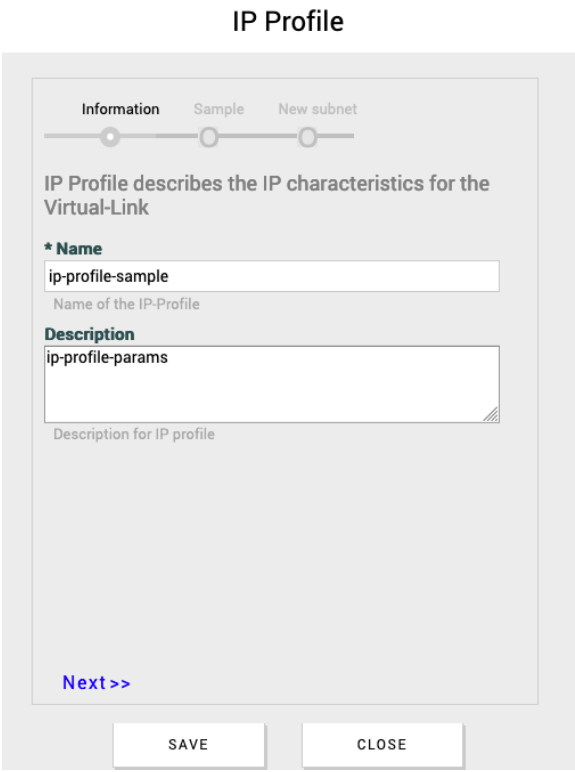
Note: The segment profiles and ip profiles can be set up at any point. This step does not need to be done when initially setting up an account.

Click on an account then go to the **PROFILES** tab.

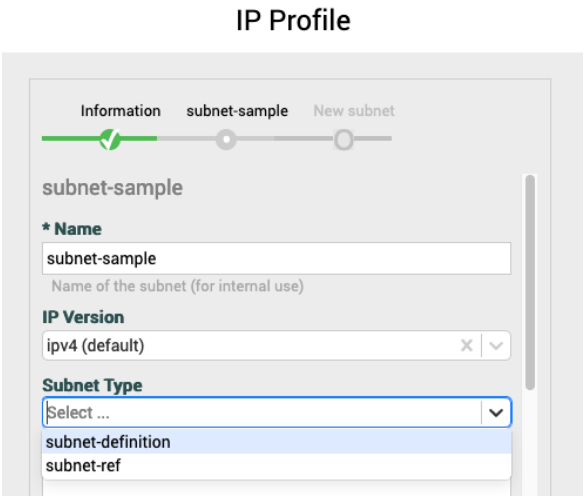


IP PROFILES

1. Click + to add a new IP Profile section. The IP Profile describes the IP characteristics for the Virtual-Link.



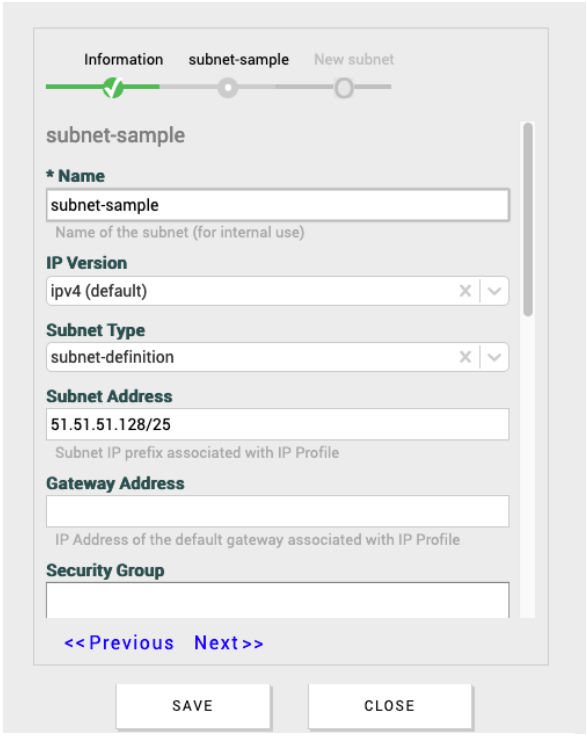
2. Provide the following information (fields with * are required):
   - ***Name**: Name of the IP-Profile.
   - **Description**: Description for IP profile.
3. Click **Next >>** to provide a New Subnet.
   - ***Name**: Name of the subnet (for internal use).
   - **IP Version**: ipv4 (default) or ipv6
   - **Subnet Type**: subnet-definition or subnet-ref



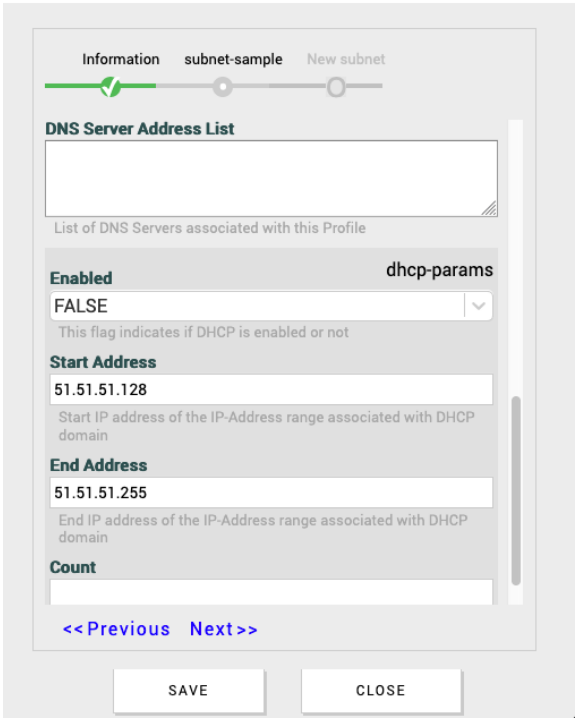
   a. If you choose **subnet-definition** then provide the following information:
      - **Subnet Address**: Subnet IP prefix associated with the IP Profile.
      - **Gateway Address**: IP Address of the default gateway associated with the IP Profile.

- **Security Group**: Name of the security group.
- **DNS Server Address List**: List of the DNS Servers associated with this Profile.

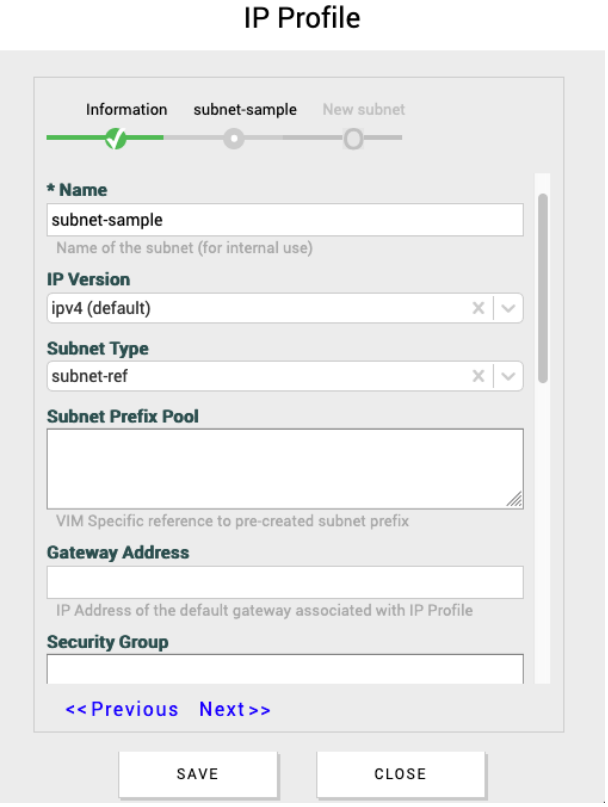
IP Profile



IP Profile



b. If you choose **subnet-ref**, then provide the following information:
   - **Subnet Prefix Pool**: VIM specific reference to the pre-created subnet prefix.
   - **Gateway Address**: IP Address of the default gateway associated with the IP Profile.
   - **Security Group**: Name of the security group.
   - **DNS Server Address List**: List of the DNS Servers associated with this Profile.

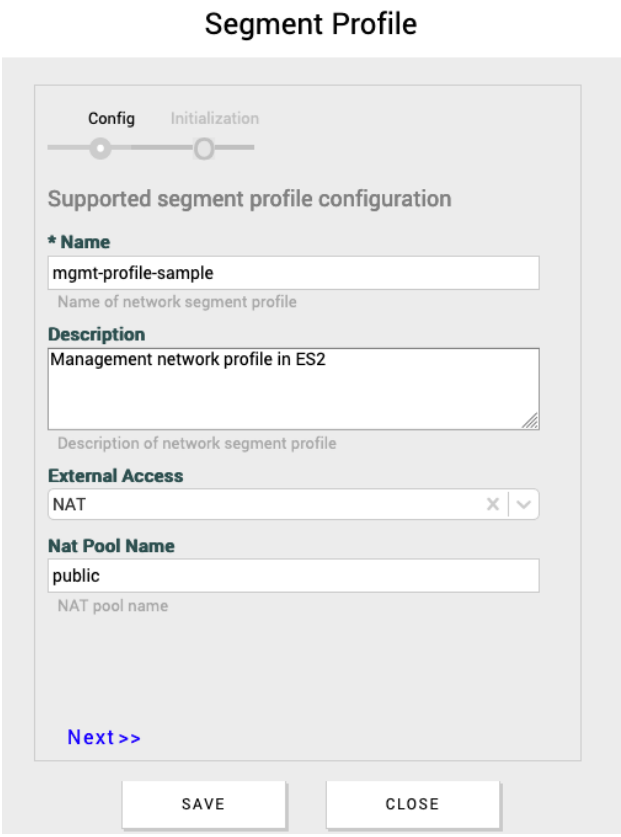


dhcp-params
- **Enabled**: This flag indicates if DHCP is enabled or not (True or False)
- **Start Address**: Start IP address of the IP-Address range associated with DHCP domain
- **End Address**: End IP address of the IP-Address range associated with DHCP domain
- **Count**: Size of the DHCP pool associated with DHCP domain.

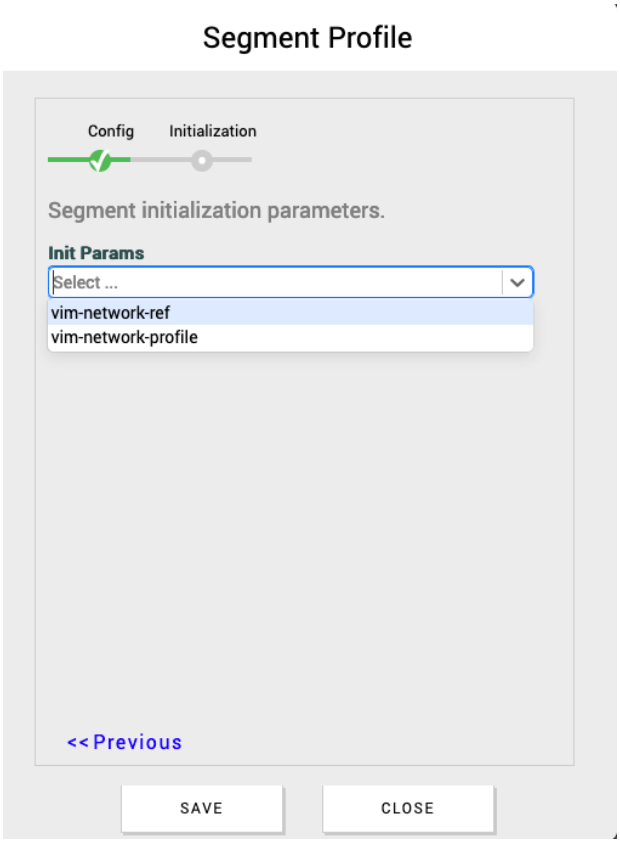
4. Click **Next >>** to provide another New Subnet.
5. Click **SAVE**.

SEGMENT PROFILES

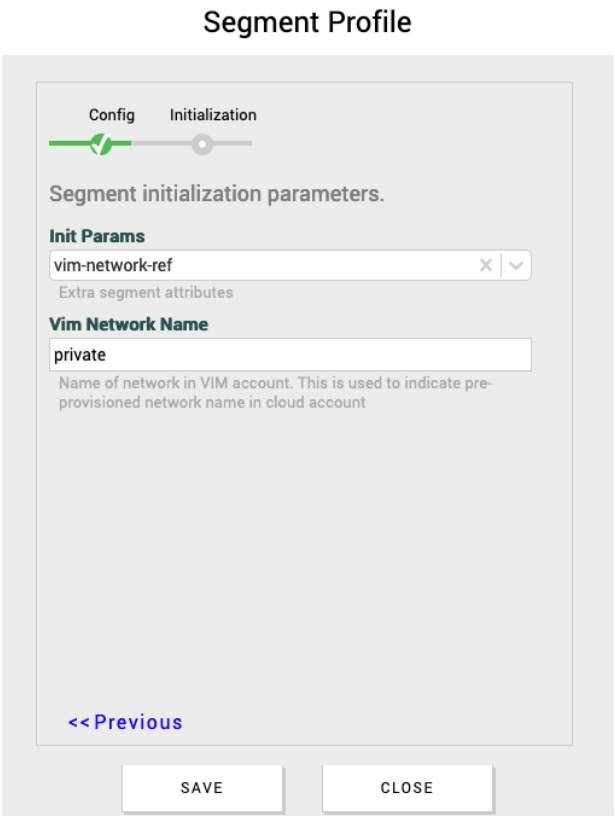
1. Click + to add a new Segment Profile section. The segment profile allows an operator to define a network configuration available on the datacenter. It may reference a previously created ip profile.



2. Provide the following information (fields with * are required):
   - ***Name**: Name of the network segment profile.
   - **Description**: Description of the network segment profile.
   - **External Access**:
     - NONE (default)
     - DIRECT
     - NAT
     - LOAD-BALANCER
   - **Nat Pool Name**: NAT pool name
3. Click **Next >>** to provide the segment initialization parameters.
   - **Init Params**: These are extra segment attributes. Choose **vim-network-ref** or **vim-network-profile**.



a. If you choose **vim-network-ref** then provide the following information:
   - **Vim Network Name**: Name of the network in the VIM account. This is used to indicate pre-provisioned network name in cloud account. Add the **Vim Network Name** and then click **SAVE**.



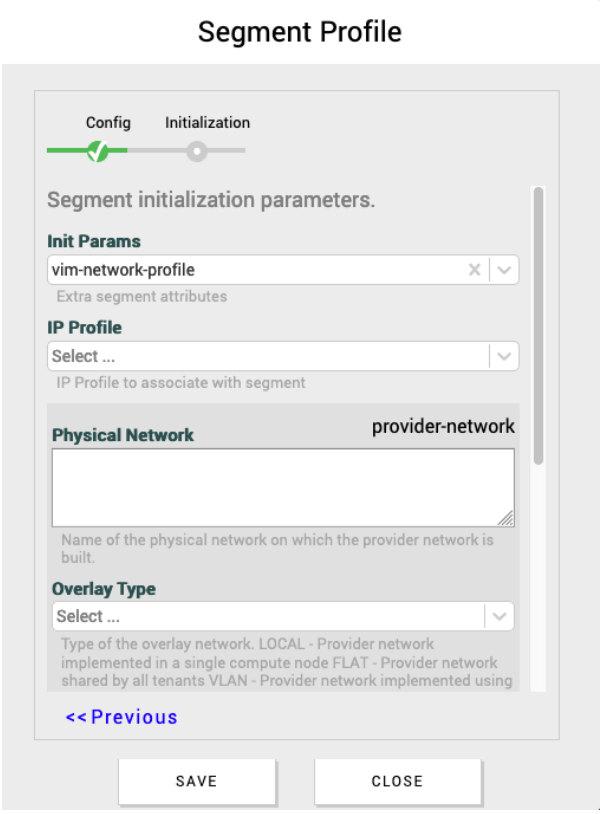
b. If you choose **vim-network-profile**, then provide the following information:
   - **IP Profile**: VIM specific reference to the pre-created subnet prefix.

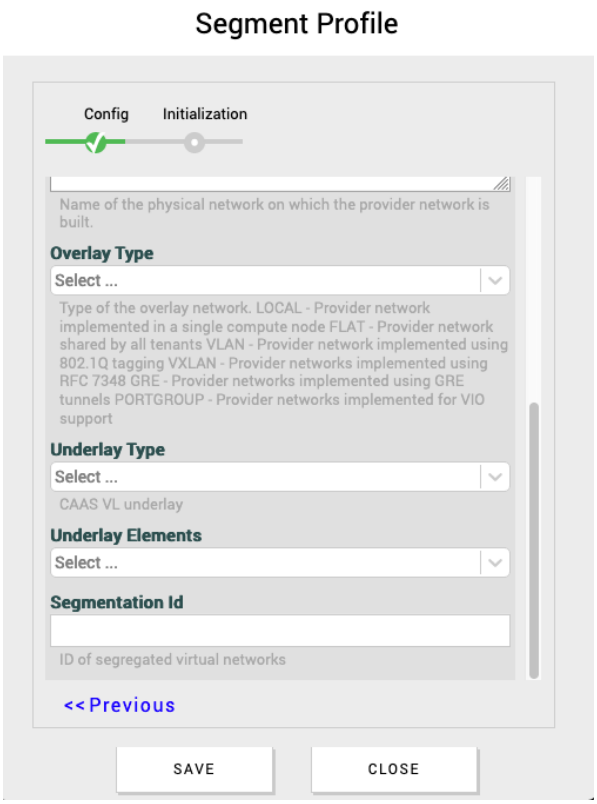
provider-network

- **Physical Network**: Name of the physical network on which the provider network is built.
- **Overlay Type**: Type of the overlay network.
  - LOCAL: Provider network implemented in a single compute node.
  - FLAT: Provider network shared by all tenants.
  - VLAN: Provider network implemented using 802.1Q tagging.
  - VXLAN: Provider networks implemented using RFC 7348.
  - GRE: Provider networks implemented using GRE tunnels.
  - PORTGROUP: Provider networks implemented for VIO support.
- **Underlay Type**: CAAS VL underlay.
  - MACVLAN
  - SRIOV
  - AWS-CNI
  - IPVLAN
- **Underlay Elements**:
  - macvlan
  - sriov
  - aws-cni

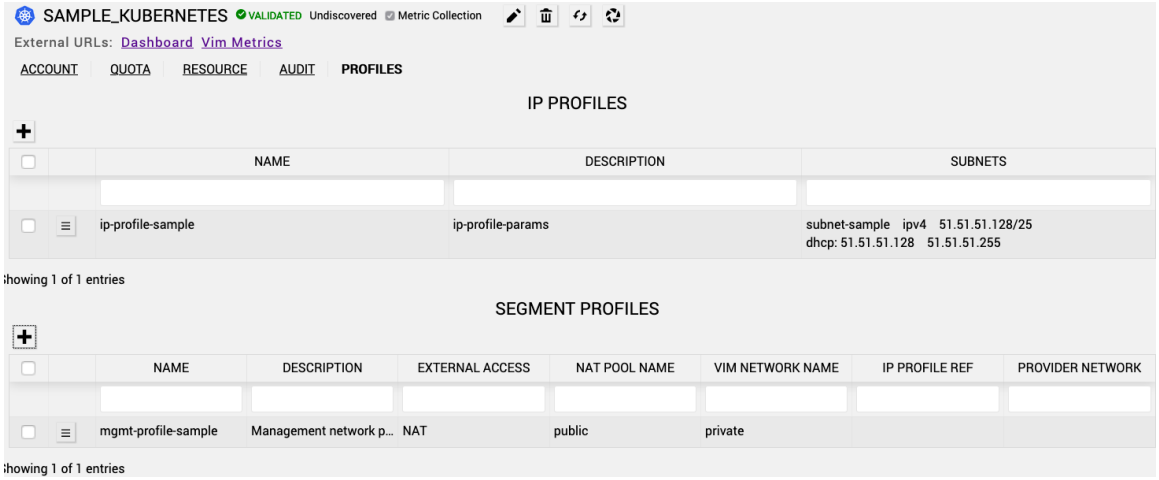
    - ipvlan
    - host-device
- **Segmentation ID**: The ID of segregated virtual networks.

4. Click **SAVE**.





After saving your configurations, the VLD and segment reference information appears on the **IP PROFILES** and **SEGEMENT PROFILES** screen.

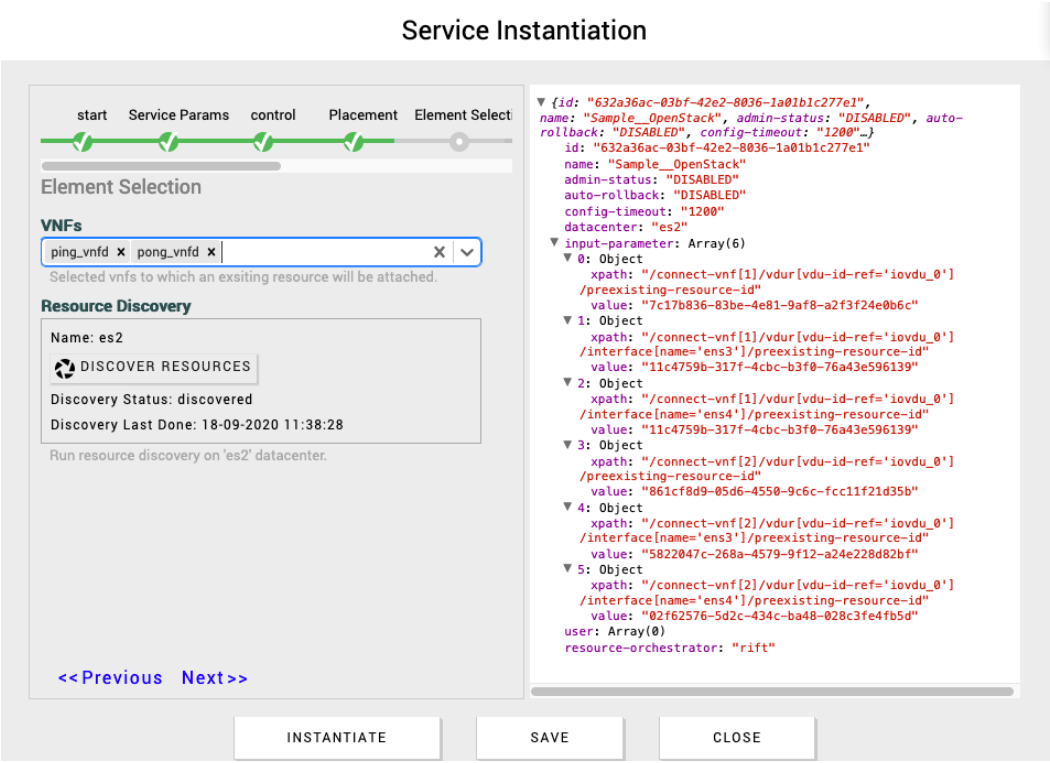


Instantiation Wizard Enhancements

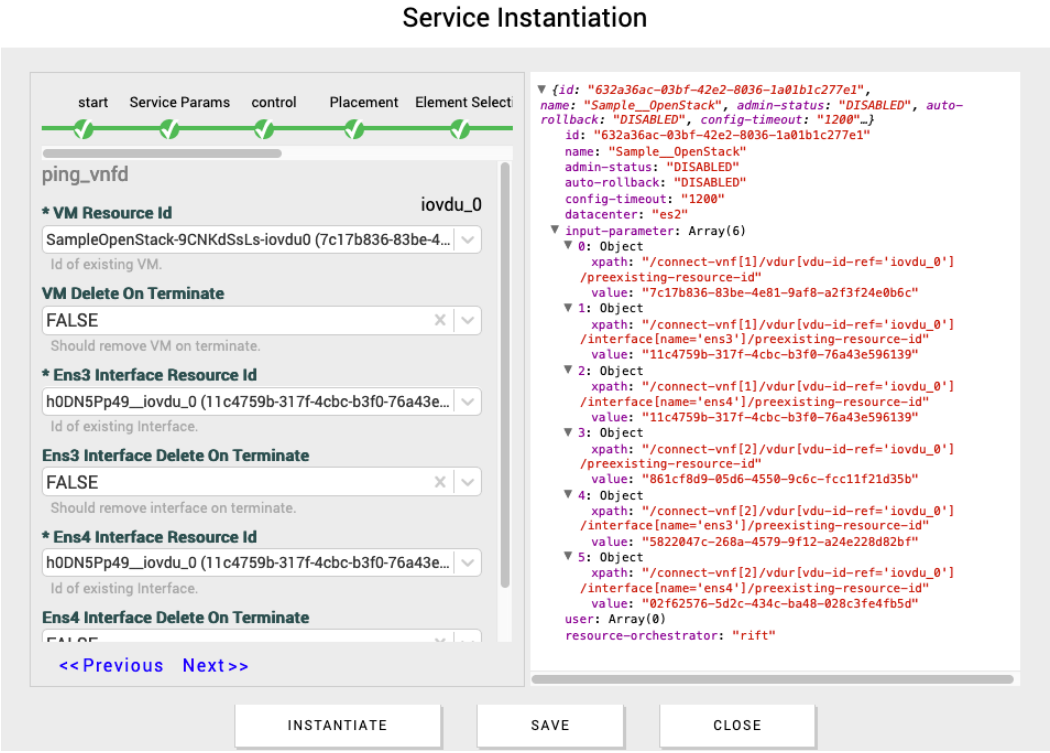
Mapping Existing Resource Improvements

This release includes improvements to the way an operator can map existing resources during instantiation. A user can now specify the existing resources to map from a drop-down menu in the **Element Selection** section. The drop-down menu will only show available resources if the operator runs discovery on the datacenter. Prior to 8.2, a user had to return to the **Datacenters** page to run discovery on an account. Now, there is a new DISCOVER RESOURCES button in the Instantiation wizard to find resources.

Click **DISCOVER RESOURCES** and the Discovery Status changes to discovering. The discover process can take several minutes. An operator can continue to perform other functions such as save, close, etc. while the discovery process takes place. Once discovery is complete the user continues with the steps to Instantiate.



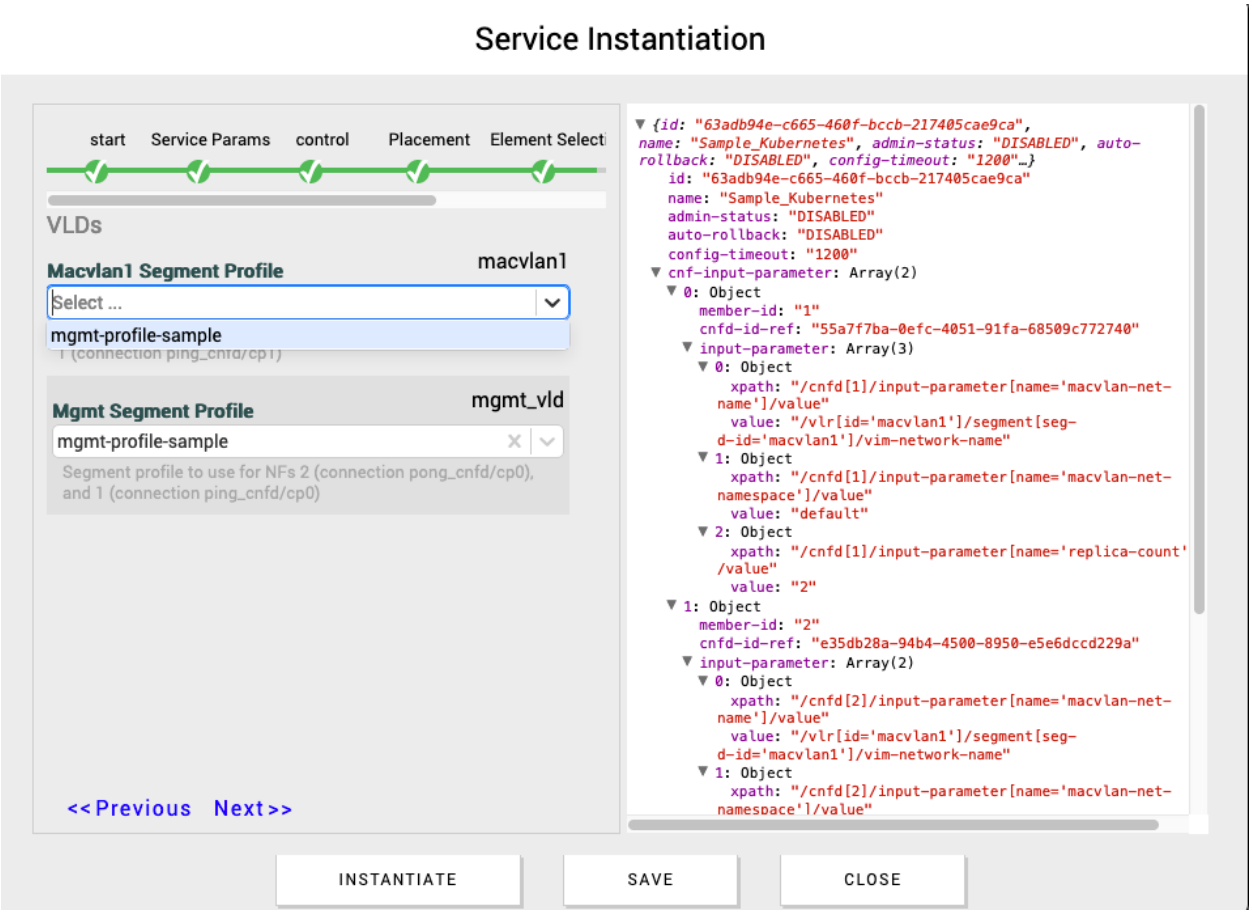
The Instantiation Wizard also includes a section to the select the VNFD (or CNFD) to be associated with a discovered set of resources.



Note: Prior to release 8.2, an operator could select VLDs to map as part of the mapping existing resources step. There was also a section to select pre-existing VLDs. This is now deprecated and will no longer appear in 8.2.. The VLDs section is now simpler with the user of the new Datacenter Profiles. See VLDs Section Changes.

VLDs Section Changes

In previous release, an operator could configure ip profiles and specify a network name, a NAT pool or an ip profiles in the NSD. These configuration steps are now deprecated. In release 8.2, the service instantiation page VLDs section allows for the selection of segment profiles.



See New Profiles section in the DATACENTER tab for steps to add ip-profiles and segment-profiles. An operator can find the segment profile values from a drop-down menu and choose a specific segment profile to user for the Network Functions.

RIFT.ware Observability Enhancements

In release 8.2, it is now possible to use Grafana Dashboards to view additional RIFT.ware statistics. Grafana is an open source analytical and visualization tool that consists of multiple individual panels arranged in a grid. In release 8.2, it is automatically installed along with Launchpad but it is a separate POD. Grafana login is managed through Oauth with Launchpad as the authentication server and it displays the past 15 days of statistics.

Installing Grafana

The Launchpad Helm chart automatically installs a link to Grafana in a Kubernetes cluster. It can be used in both a NodePort or LoadBalancer service type mode.
NodePort: When installing Grafana in NodePort mode, you must pass the external address for Grafana. This is similar to Launchpad.

Note: It must be possible to ping the external address FQDN from the respective containers.

```
helm install lp ./launchpad-8.2.0.0.tgz --set
global.namespace.name=rift-lp --set launchpad.image.tag=8.2.0.0.115661
--set externalAddress=grunt13.maas --set
grafana.externalAddress=grunt13.maas --set
grafana.service.type=NodePort
```

LoadBalancer: When installing Grafana in LoadBalaner mode, either use the IP assigned by the LoadBalancer directly or use it view a FQDN which maps it to the LoadBalancer IP address. You can determine the IP address and the port to use from the service.

```
ubuntu@grunt13:~$ kubectl -n arn-prom get svc
NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)
AGE
name-prom-alertmgr      ClusterIP   10.233.30.187   <none>
9093/TCP
25h
```

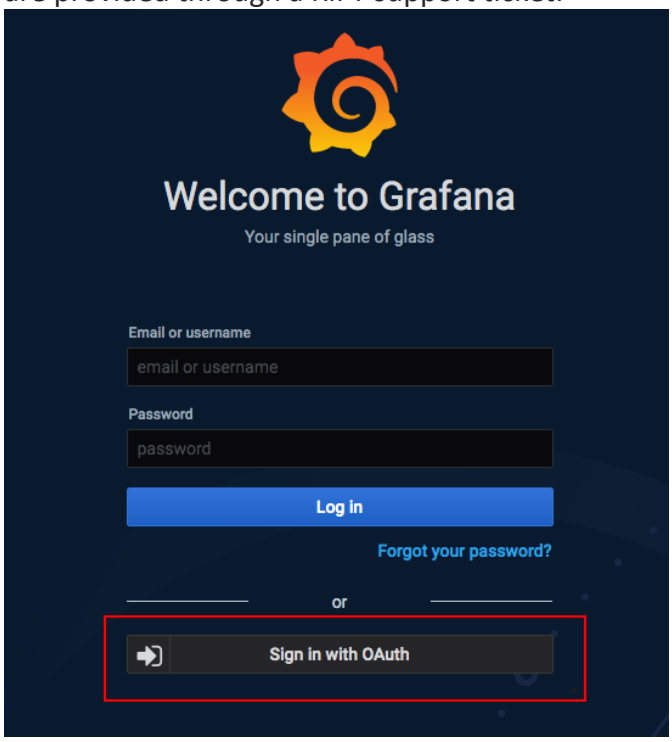
```
name-prom-grafana       NodePort    10.233.54.138   <none>
3000:30563/TCP
25h
name-prom-launchpad     NodePort    10.233.61.77    <none>
8009:32531/TCP,8006:32091/TCP,8014:30859/TCP,8008:30103/TCP,8443:32527
/TCP   25h
name-prom-loki          NodePort    10.233.10.77    <none>
3100:30510/TCP
25h
name-prom-prometheus    ClusterIP   10.233.16.48    <none>
9090/TCP
25h
```

Logging in to the Grafana Dashboard

Grafana login is processed with OAuth2.0. The Grafana Login page is redirect to the RIFT.ware Launchpad page.

Note: An operator must be a super user with super-admin or platform-admin roles to access the Grafana Dashboard.

There is a non-RIFT.ware administrator user configured with Grafana. The credentials are provided through a RIFT support ticket.



Configuring Grafana as an Administrator

When you login as a RIFT.ware user, you cannot modify dashboards or configure additional data sources. If you need to configure data, then you must login as a grafana admin “rw_admin”. In this case, you must fetch the password from the grafana config file.

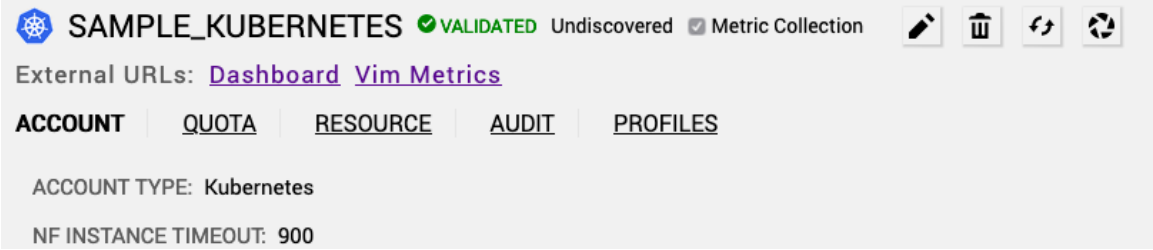
Access the grafana config by logging into a grafana container:

```
ubuntu@grunt13:~$ kubectl -n name-prom exec -it name-prom-grafana-
7c658df7df-bsvpt -- sh
# cat /grafana/config/grafana.ini | grep "password"
# You can configure the database connection by specifying type, host,
name, user and password
# If the password contains # or ; you have to wrap it with triple
quotes. Ex """#password;"""
;password =
# default admin password, can be changed before first start of
grafana,  or in profile settings
admin_password = ehhtuxxh
;password_hint = password
# If the password contains # or ; you have to wrap it with triple
quotes. Ex """#password;"""
;password =
; basic_auth_password =
;password =
#
```

The password is ‘admin_password’. In the above example, the password is ‘ehhtuxxh’.

RIFT.ware UI External Link

After installing RIFT.ware and creating a datacenter account, a Vim Metrics external link appears on the **DATACENTERS** page. Click this link to open the Grafana Dashboard.



Grafana Dashboard Statistics

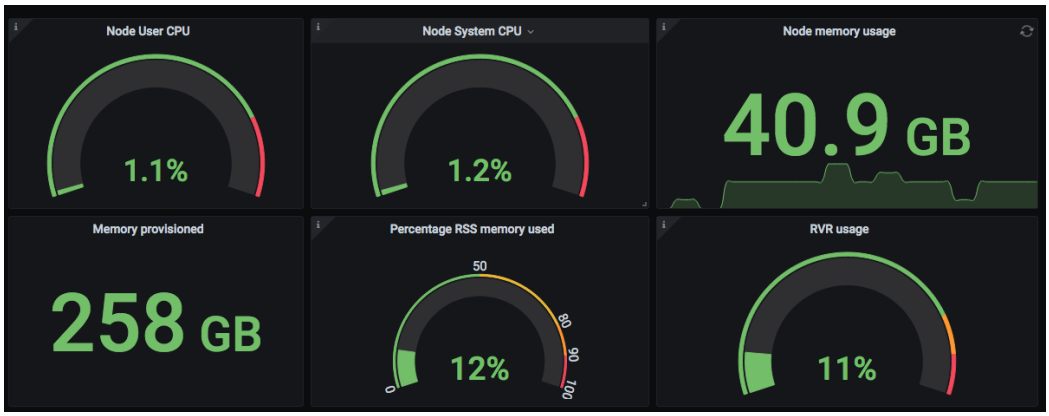
Grafana displays the following dashboard statistics.

- LP Core Metrics
- RW.Rest Metrics
- MANO Metrics
- Caas Metrics
- LP Alerts
- RW.Auth Metrics
- Metric Exporter

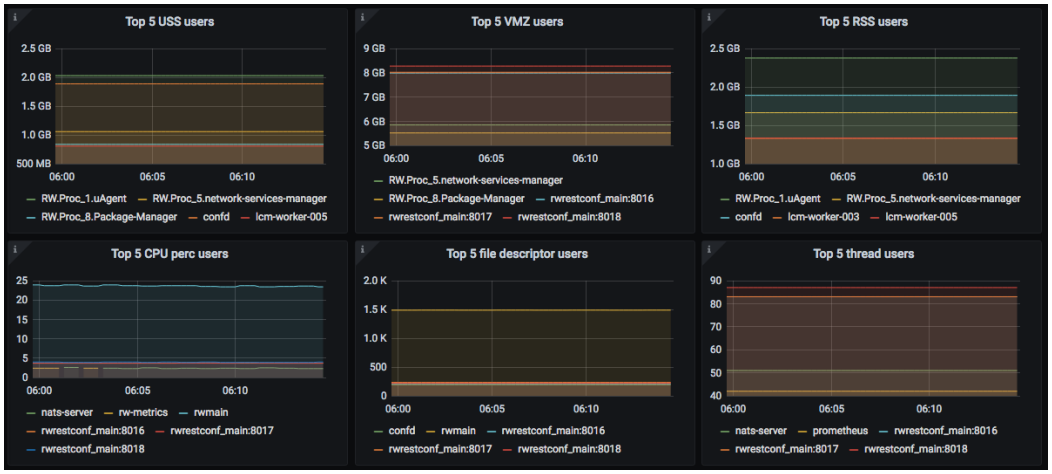
LP Core Metrics

This dashboard has many panels which can be used for LP system resource monitoring. It shows the usage trend of CPU, memory, disk, file descriptors, threads etc at the system and process levels.

Node CPU/memory, container memory and RVR disk usage example



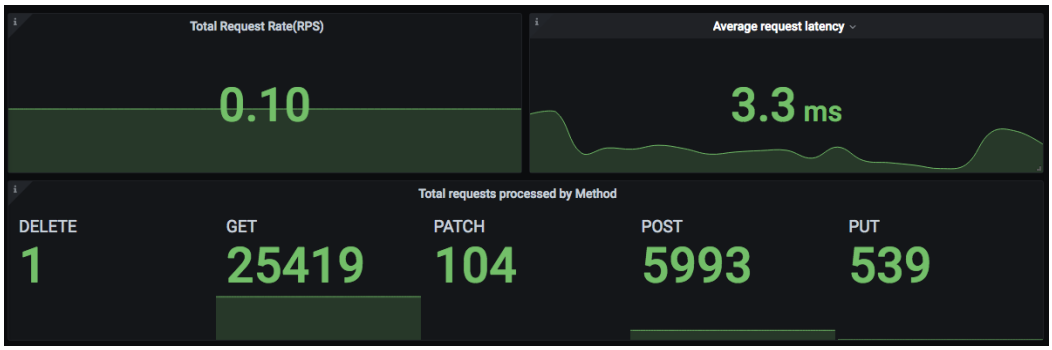
Top five resource consuming processes example

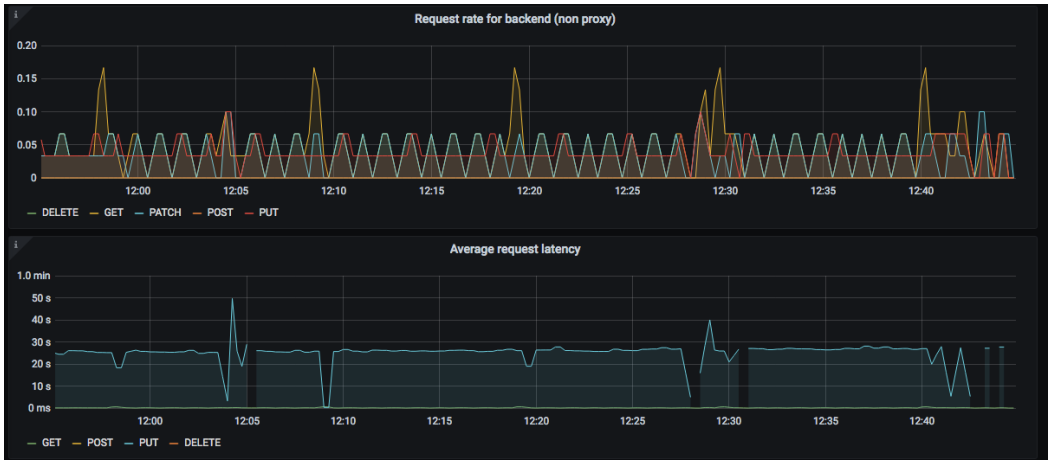


RW.Rest Metrics

This dashboard shows the request rate, latency, processing rate, request count per verb etc of HTTP(S) requests received by the RW.Rest workers.

Request rate, latency and total requests processed examples



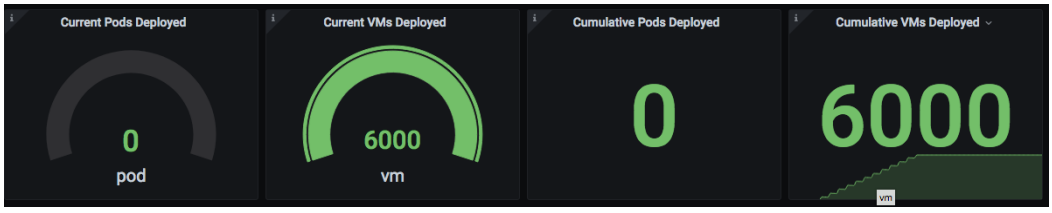


MANO Metrics

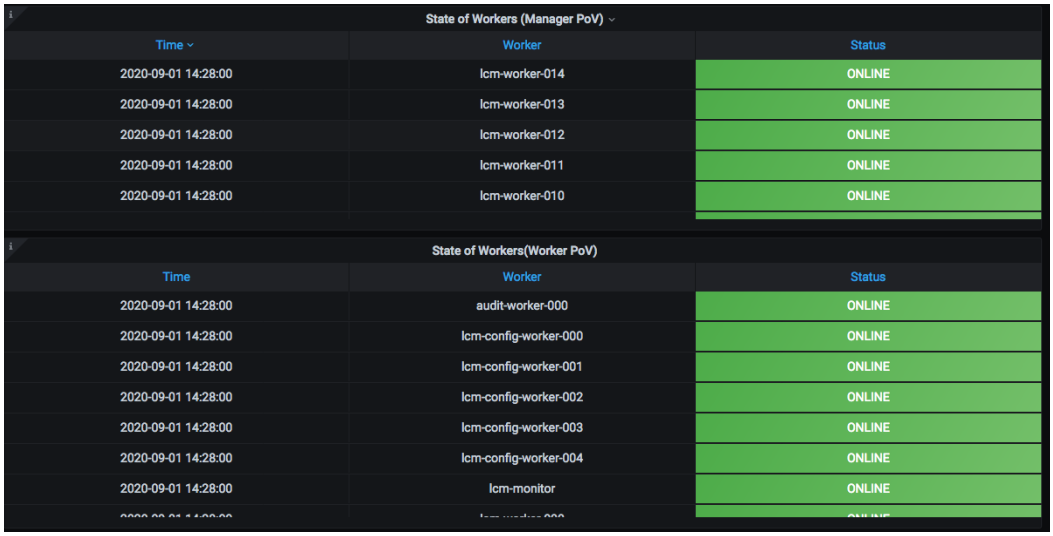
The dashboard displays system variables. Some examples of the variables that you can view in this panel are:

- the number of deployed VMs/Pods. This includes the running and cumulative count.
- the number of NSes that are managed by the orchestration layer. This includes the running and cumulative count.
- the state of the LCM workers in the backend. This section shows if the workers are online or offline for processing requests.
- the backend queue status along with the processing rates.

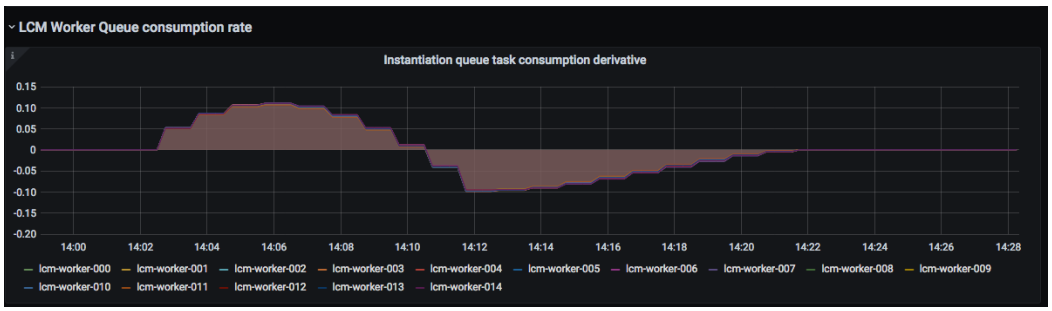
VM/Pods deployed example



Worker states example



Task completion rate example

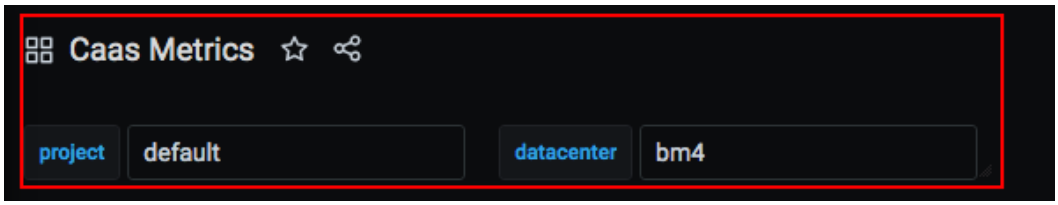


Caas Metrics

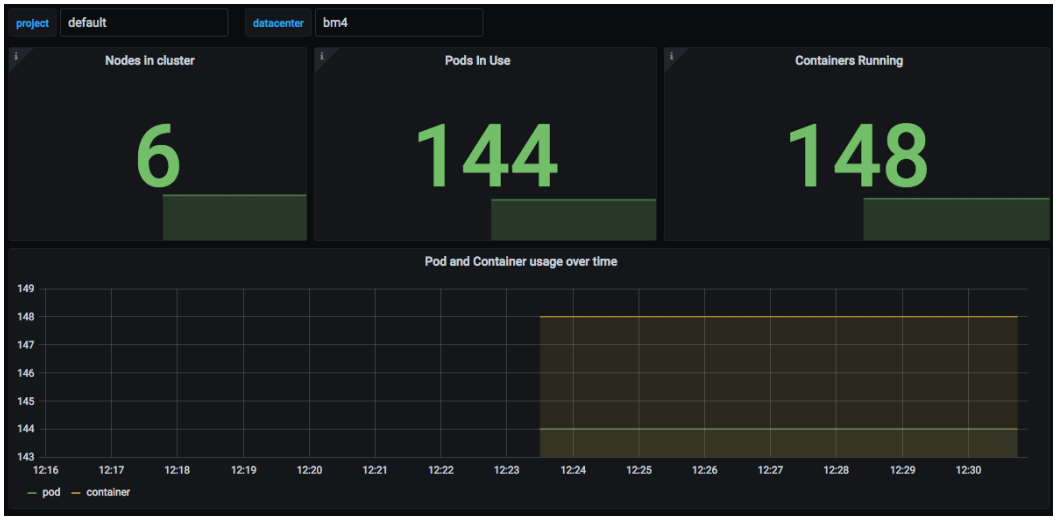
This dashboard shows statistics for Kubernetes based cloud accounts and datacenters and Amazon Elastic Kubernetes accounts and datacenters.

- **Kubernetes based cloud accounts:** The statistics are fetched from the API and metric server.
- **Amazon Elastic Kubernetes accounts**: The metrics are fetched using the CloudWatch container insights APIs.

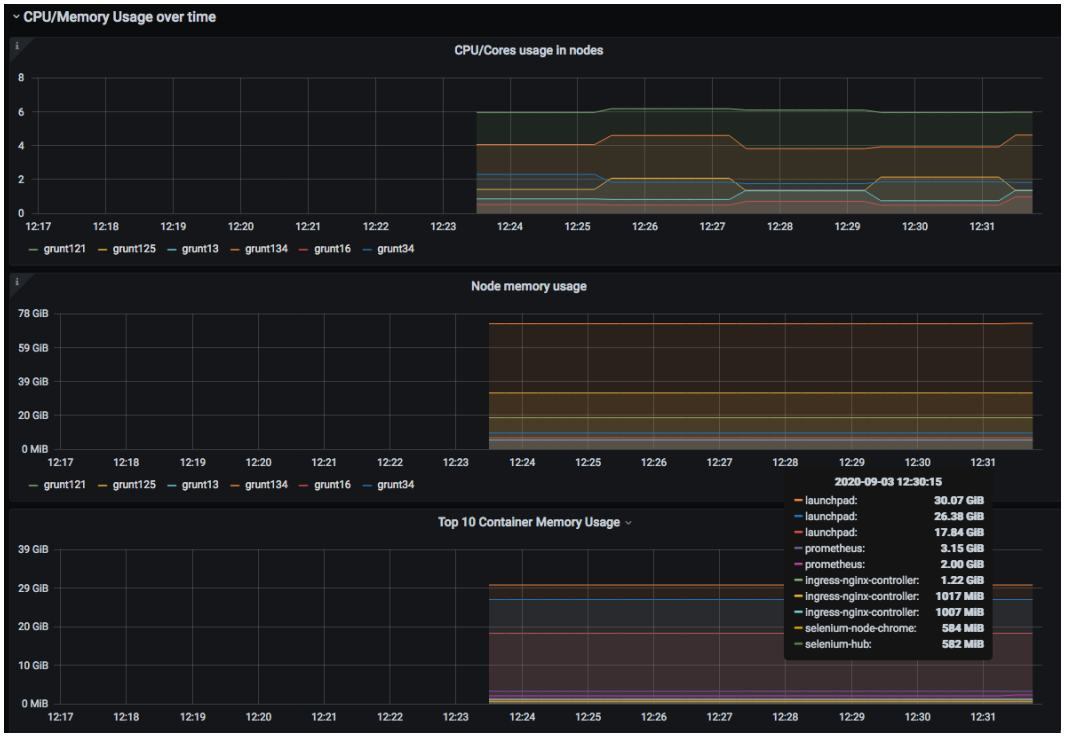
Project and datacenter as template variable values example



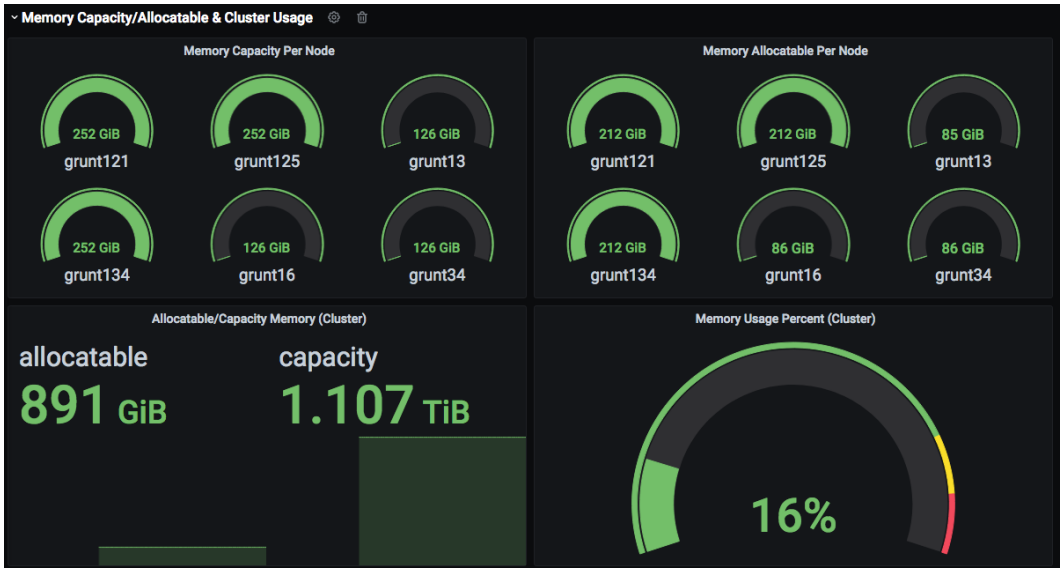
General usage stats example



Node and container level resource usage example

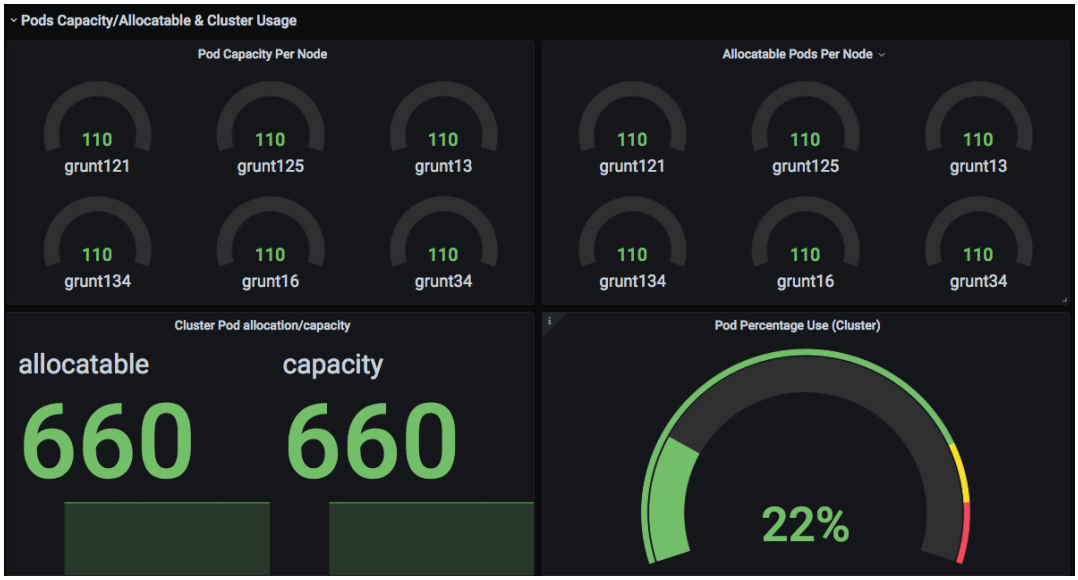


Memory capacity example



CPU capacity example

Pod capacity example



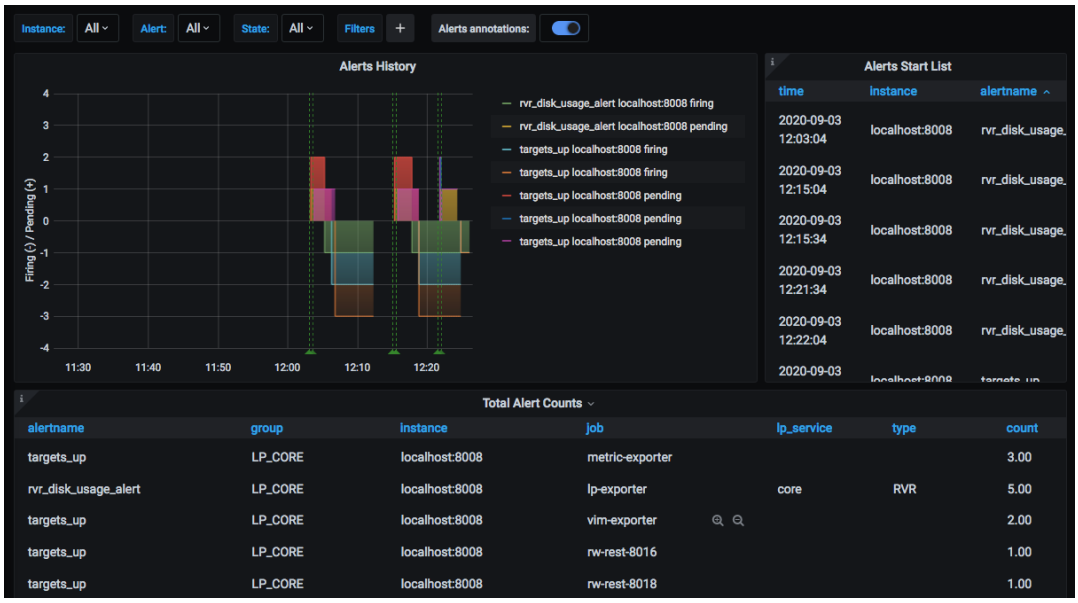
Pod resource utilization example (Amazon Elastic Kubernetes Only)

LP Alerts

This dashboard displays all the alerts that have been triggered by RIFT.ware. The possible alerts are system resource usage alerts, targets up/down alerts, MANO policy invoked alerts or NS auto scaling alerts. You can customize this panel to displays specific alerts. Some of the alerts that you can pick for this panel are:

- the length of time that an alert is in the FIRING state.
- the number of times that a particular alert has occurred.
- the length of time it takes for an alert to be in the resolved state.

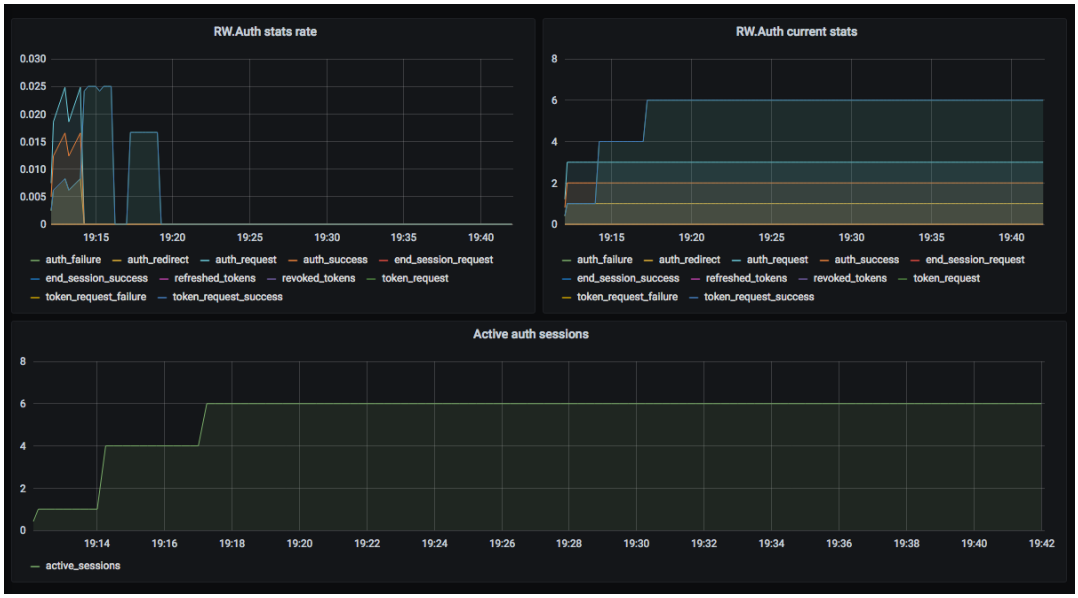
LP Alert example



RW.Auth Metrics

This dashboard shows the authentication statistics. For example, the panel shows the historical view of openidc provider stats.

RW.Auth example



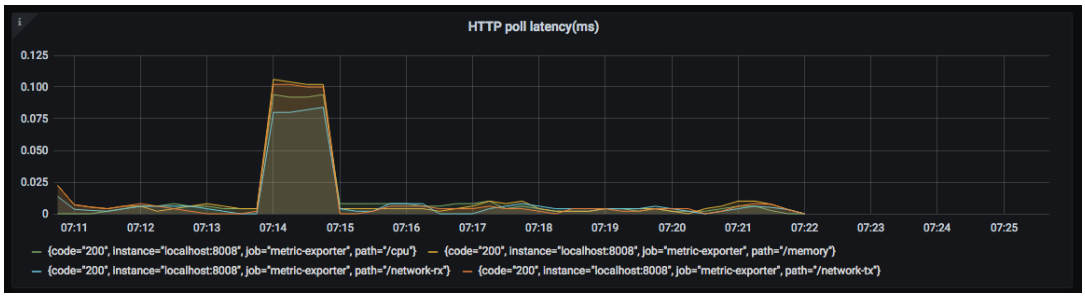
Metric Exporter

This dashboard displays the scalable monitoring param exporter statistics. Some of the possible reports are:

- the number of endpoints that are being polled.
- the latency of the polled requests.
- the failed polled requests.

Success poll rate and connected HTTP endpoints

Poll latency



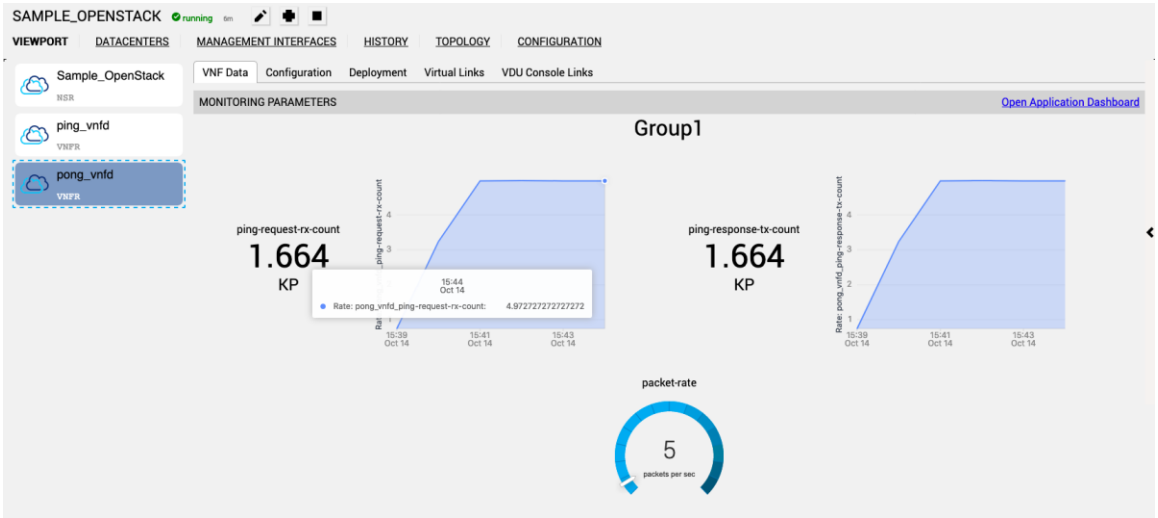
Scalable Monitoring Parameters Support

In release 8.2, the RIFT.ware orchestrator is more scalable and it supports monitoring a variety of components using difference schemes and protocols. RIFT.ware can now monitor a Network Service using HTTP polling, Prometheus scraping, Kubernetes VIM monitoring and Kubernetes resource watching. In addition, RIFT.ware now stores the fetched monitoring parameters from the Network Service in a time series database (Prometheus). An operator can view dynamic statistics for all VIM accounts on the updated QUOTA tab in the UI. See QUOTA Tab Enhancements for more information on the updated display. This feature also includes an updated display for historical monitoring params for individual Network Function (NF). The new view is not visible at the NS level but you can see the updated display when you switch to an individual NF on the Viewport screen.

Operators and the RIFT support team can also view historical data on the Grafana Dashboards. This data will include KPI metrics and NS and VNF metrics. See the RIFT.ware Observability Enhancements section for more details about Grafana.

Monitoring Params UI Enhancements

1. On the **Launchpad** menu, click **SERVICES** > **FULL LIST** to open the list of NS instances.
2. Click  next to the NS that you want to see in more detail. The **VIEWPORT** screen opens.
3. Select a record on the left to view a graphical representation of its monitoring parameters.



Support for Enhanced Datacenter Accounts and Informational Display Updates

This feature introduces an updated display on the **QUOTA** tab in the **DATACENTERS** section. The **QUOTA** tab now displays dynamic statistics for all VIM accounts. The information now represents the actual state of resources that are being consumed at a given time.

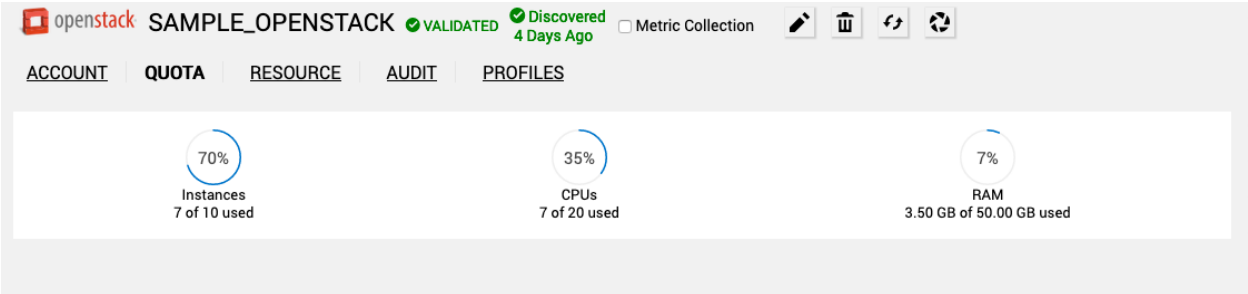
Some datacenters such as RedHat OCP now support using an external API to gather more information and allow for linking out to the actual datacenter UI. There are model changes to support a new **Misc** connector account to provide the information to be used for the external API and an updated datacenter account model for this Miscellaneous account to be linked with a datacenter.

QUOTA Tab Enhancements

Previously, the information provided in the **QUOTA** tab was static and it only reflected the resource utilization as of the most recent datacenter discovery operation. In release 8.2, this tab now displays the actual state of resources that are being consumed at any time. The display varies slightly depending on your account type.

VIM Accounts

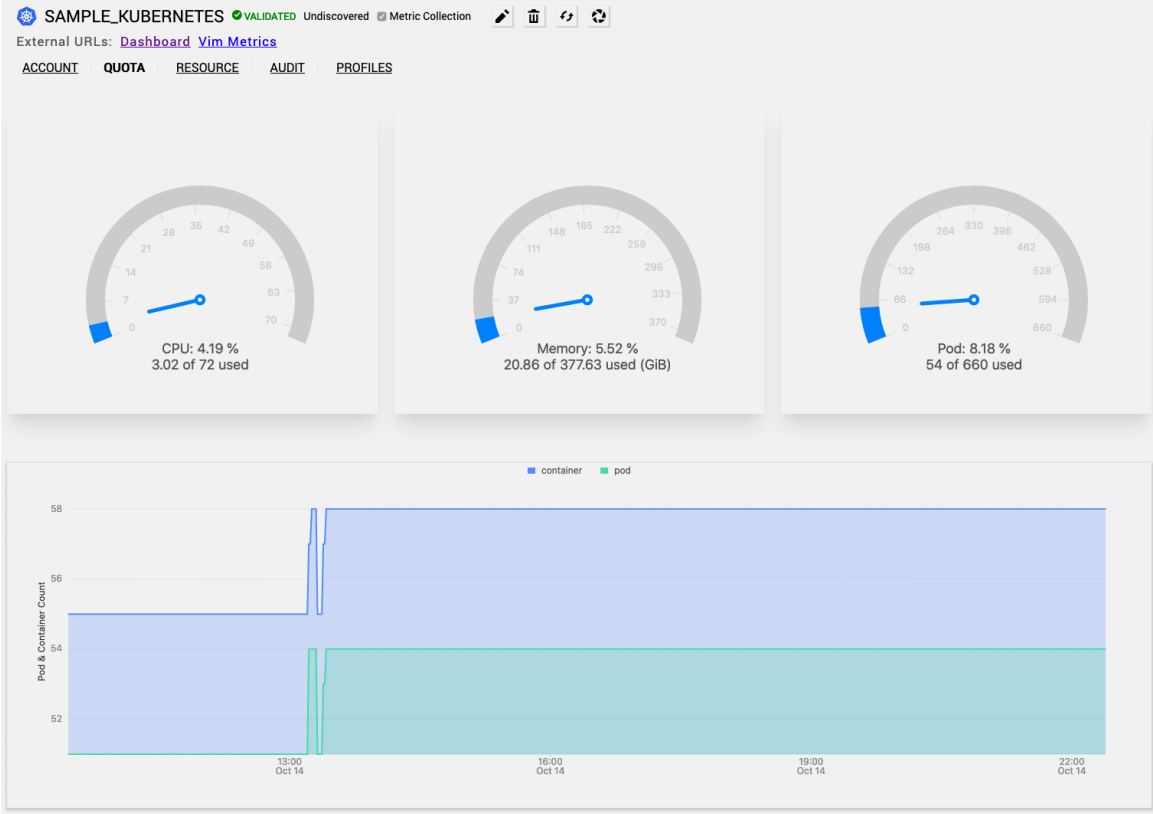
The quota tab displays dynamic Instances, CPUs and RAM used.



Kubernetes Accounts

The quota tab displays dynamic CPU, memory, Pod used. This section also includes the following statistics:

- Pods & Container Count
- CPU/Core Usage in Nodes (%)
- Node Memory usage (GIB)



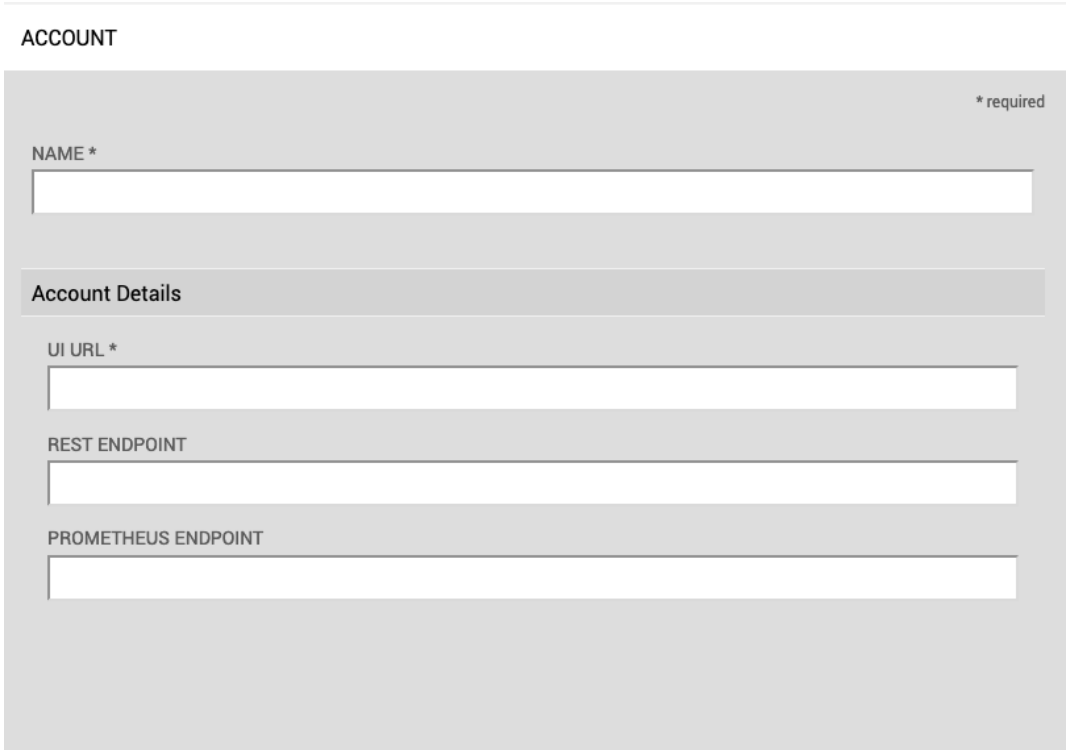
For additional statistics, reference the Grafana Dashboards. See RIFT.ware Observability Enhancements for more details about this tool.

Account Enhancements in the UI

New Misc Connector

There is a new Misc connector in the UI for Miscellaneous accounts.

1. On the Launchpad menu and click **CONNECTORS**.
2. Under **ACCOUNT TYPE**, choose **Misc** and click **CREATE ACCOUNT**.
3. Add the unique **Misc** account **Name*** for the existing Misc account you are linking to.
4. Under Account Details, provide the following information (fields with * are required):
   - **UI URL***: URL to the UI
   - **REST ENDPOINT***: URL for the REST API endpoint
   - **PROMETHEUS ENDPOINT***: URL to the Prometheus endpoint
5. Click **SAVE**.



New Datacenter RedHat OCP and OSP VIM Account

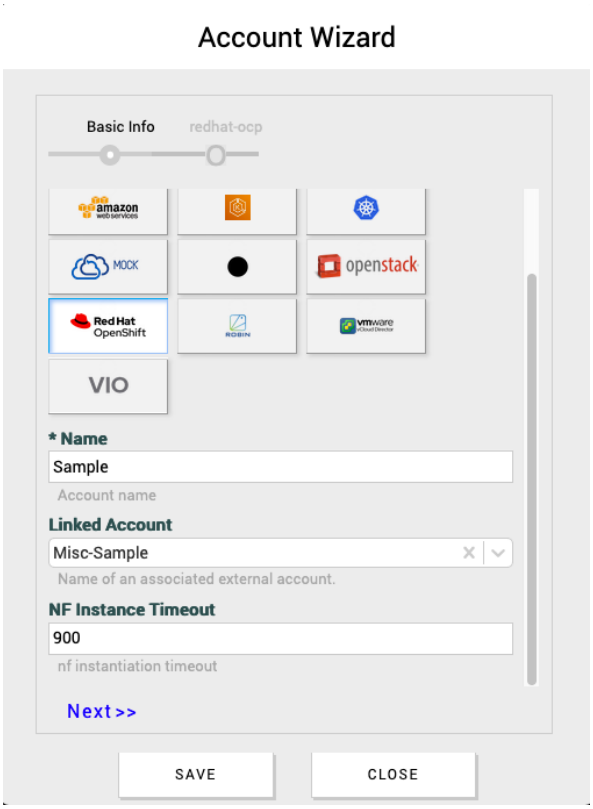
RIFT.ware now supports orchestration of CNFs or VNFs on RedHat OCP (v4.3 minimum) and OSP 13.

- RedHat OCP (CNFs)
- RedHat OSP 13 (VNFs)

Redhat OCP (CNFs)

The Datacenter section in the UI displays a new RedHat OpenShift account type.

1. On the **Launchpad** menu, click **DATACENTERS**.
2. Click + to add a new datacenter. An account wizard appears.
3. Choose Red Hat OpenShift in the **Account Type** section.
4. Add a unique account name in the ***Name** field.
5. Add a link to an associated external account in the **Linked Account** field.
6. Add the **Nf Instance Timeout** (SECONDS) information. This is the maximum time allocated for resource instantiation in the RedHat OpenShift account. Default is 900.



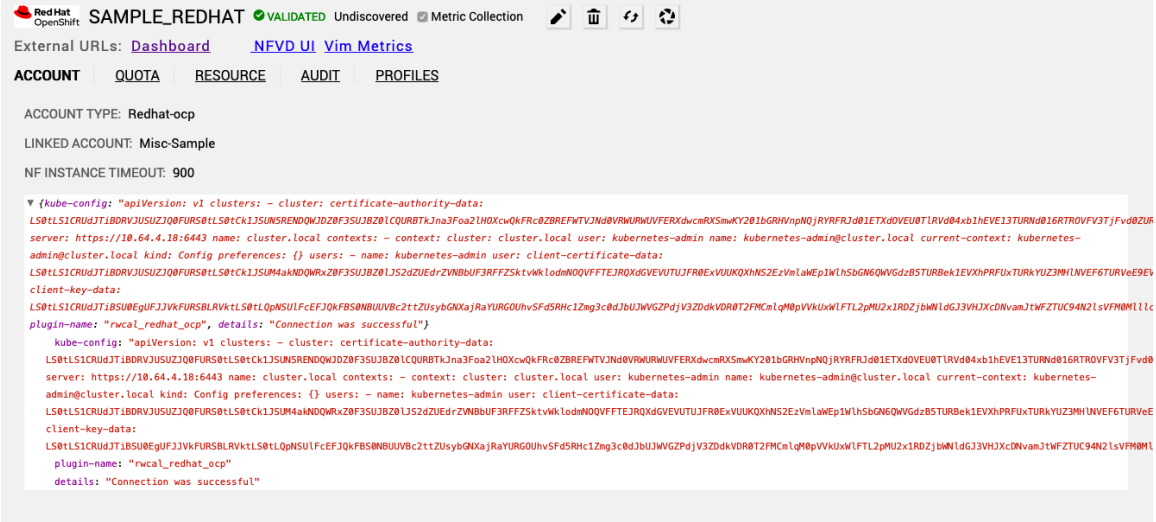
7. Click **Next >>** and provide the following OpenShift Container Platform as a cloud account details:
   - **KUBE CONFIG***: Click Browse to provide the Kubernetes configuration file in YAML format or type the Kube Config in the Contents of the Kube config file in YAML format field.
     Note: This configuration file must be provided by the operator.
     Organizing Cluster Access using Kubeconfig Files
   - **Monitoring Server**: Select ip-based or server-based.
     - **ip-based**: If you choose an ip-based monitoring server, then add:
       - **Monitoring Server IP address**: IP of the metric server.
     - **server-based**: If you choose a server-based monitoring server, then add:
       - **Service Name**: Service name of the monitoring server.
       - **Service Namespace**: Namespace of service of metric server.
8. Carefully check your entries and click **SAVE**.

RedHat OSP 13 (VNFs)

RedHat OSP 13 is equivalent to Openstack. An operator can use an OpenStack Account type to instantiate a RedHat OSP service.

Linked Accounts

After adding the Misc Connector account and the RED RedHat OCP (v4.3 minimum) or OSP 13 Account, the external url; **ACCOUNT TYPE** and the **LINKED ACCOUNT** appear on the **DATACENTERS** → **RedHat OpenShift** Account page.



Model Changes

Misc Connector

This feature introduces a new model to add a Misc connector account in RIFT.ware.

```
augment "/rw-project:project" {
    container misc-connector {
      list account {
        key "name";
        leaf name {
          description "Name of the connector account.";
          type string {
              length 1..256;
          }
        }

        leaf account-type {
          description "Type of connector.";
          type connector-type;
        }

        choice connector-type {
          container nfvd {
            leaf ui-url {
              type inet:uri;
              description "Base URL to the NFVD UI";
              mandatory true;
            }

            leaf rest-endpoint {
              type inet:uri;
              description "Base URL to the NFVD REST API endpoint";
            }
```

```
            leaf prometheus-endpoint {
              type inet:uri;
              description "Base URL to the NFVD Prometheus endpoint";
            }
          }
        }
      }
    }
```

Cloud Account Updates

The following changes have been made to the model:

- Added a new `redhat-ocp` field to `rw-cloud:account` in the rw-cloud:cloud Data Model. This field includes two new variables: `kube-config` and `plugin-name`.
- Added a new `linked-account` field to the datacenter (cloud account). This new fields links the misc connector to the Datacenter.

```
augment "/rw-project:project" {
    container cloud {
      list account {
        description "Configure Cloud Accounts";

        max-elements 2048;
        key "name";

        leaf name {
          mandatory true;
          type string {
            length "1..255";
          }
        }

        leaf sdn-account {
          description "SDN account associated with this cloud account";
          type leafref {
            path "../../../rw-sdn:sdn/rw-sdn:account/rw-sdn:name";
          }
        }

        leaf linked-account {
          description "Other linked accounts related to this";
          type leafref {
            path "../../../rw-misc-connector:misc-connector/rw-misc-
connector:account/rw-misc-connector:name";
          }
        }

        uses rwcloudcommon:provider-auth;
        uses rwcloudcommon:nf-instance-timeout;
      }
    }
  }
```

Fixed Issues in RIFT.ware 8.2.0.2

Support Tickets	Title	Description
RIFT-28519	Config agent account broken	An operator went to the connector plugin in the UI and created a config agent account with random credentials. After 5 to 10 minutes, the account did not have a connection status or a message. Also, the field 'secret' from the UI was not present in the API response when a user queries config-agent. Lastly, the oper-state was missing. This issue is resolved.
RIFT-29355	NS Instantiation stuck at INIT stage while using instantiation variable file	An operator added a datacenter and then uploaded ping pong descriptors. In the Create NS wizard, a user attached the instantiation variable file and clicked YES on the instantiate radio button. The caused the NS to be stuck in the INIT stage. This issue is resolved.
RIFT-29396	LP-based CNF orchestration does not work when restricted to non-default namespaces	RIFT.ware expected a full-cluster-access Kube-config and used this config to create a net-attach-def CRD in the default namespace used for networking. If the Kube-config provided by the user did not have access to the "default" namespace, then instantiation failed. This issue is resolved.
RIFT-29601	Brownfield Discovery with Scaling descriptors creating fresh new instance on OpenStack.	While attempting discovery, a new instance was created instead of discovering the existing scaling instance in the VIM. This issue is now resolved.
RIFT-29777	Update LP to be launched on OCP	Instantiation failed when deploying RIFT.ware Launchpad using a RedHat OpenShift account. To avoid this issue, pre-create the namespace for deploying RIFT.ware when using RedHat OpenShift. Also, assign the service account used for deploying RIFT.ware with 'anyuid' scc (security context

Support Tickets	Title	Description
		constraint) privileges. The scc can be assigned to the default service account in the current namespace using: `oc adm policy add-scc-to-user anyuid -z default` This issue is resolved.
RIFT-30157	Support for Redhat Openshift for 8.1	When deploying in OpenShift, the namespace has to be pre-created and the default service account should have an anyuid scc policy. It is mentioned in the default values file. If using NodePort, then ensure that Launchpad is running a namespace without any other services in it. Also, make sure that the kubeconfig has a context and the default context has the namespace set to default. This issue is resolved.
RIFT-30678	inotify has too many instances; causes starting trouble for LP	In Ubuntu 20 host OS, the default configuration for /proc/sys/fs/inotify/max_user_instances is low (For example: 128). In this scenario, there could be issues in starting up multiple Launchpad instances on a single k8s node. This could also be an issue with a single Launchpad instance if there are other apps running on the same node making use of inotify based watchers. The resolution is to increase the value of "fs.inotify.max_user_instances" to a sufficiently high value. This issue is now resolved.
RIFT-30750	Unrestricted File Upload	Updated the model to restrict the upload file size to the staging area. The config is /project/<name>/staging-areas-config/maximum-upload-size. Default: 10 MIB An error will appear if you attempt to upload a file that is larger than 10 MIB. This issue is resolved.

Support Tickets	Title	Description
RIFT-30922	RH OSP10 Openstack connection fails with CAL VIM Limits API failed and API version mismatch	RH OSP10 Openstack connection failed from RIFT.ware while using a higher version of the API. Updated RIFT.ware to use the correct version of the API for RH OSP 10. This issue is resolved.

Known Issues in RIFT.ware 8.2.0.2

Support Tickets	Title	Description	Impact	Workaround
RIFT-13715	Confd configuration transaction abort results in inconsistent state.	Confd can abort configuration change transactions due to its own internal reasons. RIFT.ware cannot undo the changes (on an ABORT form Confd) because it is already internally committed.	This could result in a data mismatch between that is there in the configuration database and what is known to the RIFT.ware backend.	This issue mostly occurs when a user makes successive config changes without any idle time in between the modifications. Any test or automation script using RIFT REST APIs to complete configuration changes must have a delay between two successive config change operations.
RIFT-21978	Password is displayed in plain text in event logs.	A password is displayed in plain text in the events log.	A password is displayed in plain text in the events log.	N/A
RIFT-21923	TC_CONCURRENT_NS_TERMINATE: PackageDeleteError	When multiple descriptor (NSD/VNFD) packages are deleted concurrently using an API, the config data in the ConfD might be missing for a brief period for a few packages.	The config data in the ConfD might be missing for a brief period for a few packages.	When using an API to delete multiple package entries, do not send multiple DELETE requests concurrently. Wait for a request to complete before firing another request.
RIFT-23621	Introduce a new field in VNFM account page for RIFT.ware<-->VNFM handshake URL.	This is a request to add a new field in the VNFM accounts page that the Operator so the operator can populate the URL for handshaking. If the URL field is empty,	RIFT.ware uses a GET call to /vnf_instances to validate VNFM Account. If a SVNFM account does not support GET on	N/A

Support Tickets	Title	Description	Impact	Workaround
		handshake is not required.	/vnf_instances, then the VNFM Account validation may Fail. The UI shows the account status as failed. If the SVNFM account supports notification, then RIFT.ware sends a subscription request next as part of the validation. This is successfully and the VNFM account status is successful. Therefore, a SVNFM account which doesn't support GET call on /vnf_instances, then the VNFM Account status will appear to fail. This will not block any further operations such as instantiation and termination.	
RIFT-23760	Terminating one service with the same name on any Launchpad interrupts existing services on all Launchpads.	This issue occurs because all services on all Launchpads have the same UniqueID which is the InstanceID on ES2. When a user terminates a service, then terminates Instance on ES2 which causes issues for all existing services	If a user terminates any of the services on either of the network-services, this will make any other services orphan.	Use a unique NSR name if an organization is using multiple Launchpads. Ensure that an OpenStack tenant is controlled by a single Launchpad. **Note**: There are no plans to change this behavior.

Support Tickets	Title	Description	Impact	Workaround
		because they have the same UniqueID.		
RIFT-23916	Master: HA- Re-Paring of active-node after failover leaves the nodes in split brain condition.	HA pairing with a node can only be executed once. Upon re-pairing an already paired node, then the system might go into an unstable state.	If an operator attempts to re-pair an already pair node, then the system might go into an unstable state.	Be careful while performing HA pairing. Always wait for a few seconds (30 - 60 seconds) and watch the redundancy state of the peers before performing any commanded action.
RIFT-24872	Updating ipv4 to ipv6 fails in the case of FIXED IP ADDRESS and vice versa.	A user modified the 'FIXED IP ADDRESS' field and then added an IPv4 address in the descriptor details pane for a VDU. Next the user updated the configuration to IPV6 and clicked the 'Update' button. An error message appeared on the UI.	This issue only occurs when there are two IP addresses in interface configurations (IPv4 or IPv6) and the user tries to change one address to one with a different version.	Delete the existing address and add one new address with different a version.
RIFT-24992	Running AWS NS fails after failover to a new LP.	A user created an NS on Launchpad 1. The service is in the running state for tiny descriptors with an AWS VM account. The user set up an HA pairing with another Launchpad of a different version. Failover occurred to Launchpad 2; the NS was that running in the Launchpad 1 failed in instantiate in Launchpad 2.	N/A	Terminate the NS and re-instantiate in the second LP.

Support Tickets	Title	Description	Impact	Workaround
RIFT-25610	AWS - NS stuck indefinitely in VNF-INIT phase when using a scaling-group and constituent VNF's start-by-default = False.	A user tried to instantiate a simple NS with one VNF, a single VDU (for an Ubuntu VM), and a single VLD. The NSD had a scaling group and the constituent VNFD's start-by-default value is False. When the NS is instantiated, it is stuck forever in the VNF-INIT phase. There are no errors seen in the Event Logs or in rift.log.	No VNF is started by default. Since the min-instance-count for scaling group is 1, the VNF is supposed to be scaled out once NS instantiates.	N/A
RIFT-25902	Discovery gets triggered even on failed account due to non-reachable network status.	A user created a new VIM account to OpenStack. If the network is down and the cloud setup is not reachable, then when a user attempts discovery it fails but the process is still triggered.	Misleading status of VIM account on UI.	N/A
RIFT-26409	AWS alarm was failed to create after one of the instance (running) failure in AWS	RIFT supports AWS alarms (SNS notifications). These alarms help in monitoring the state of the VM (VDU). Upon failure of the instance in the AWS, an SNS notification appears.	After the instance failure from the AWS console, the NS should fail in Launchpad. This issue is not occurring correctly.	A user must depend on the monitoring params to check the state of VMs.
RIFT-26498	Creation of custom "User" in the VDUs via	Launchpad provides the option to create custom user accounts	It is not possible to create custom user accounts in VMs	Once the VM is running, user accounts can be created by

Support Tickets	Title	Description	Impact	Workaround
	RIFT.ware isn't working.	in VMs from RIFT.ware using cloud-init. This is not working correctly.	with RIFT.ware. User accounts must be created directly in the VM after logging into Launchpad.	logging into the VM and using the operator interface.
RIFT-26567	Placement groups: transaction failure prints 5 traps riftLPUserSessionEnd	An operator loaded ping pong scaling descriptors and then deleted placement groups from the NSD an VNFDs. Next, the user created and instantiated the NS. An error message and 5 traps appear.	Duplicate traps are generated.	N/A
RIFT-26409	AWS alarm was failed to create after one of the instance (running) failure in AWS	RIFT supports AWS alarms (SNS notifications). These alarms help in monitoring the state of the VM (VDU). Upon failure of the instance in the AWS, an SNS notification appears.	After the instance failure from the AWS console, the NS should fail in Launchpad. This is not occurring correctly.	A user must depend on the monitoring params to check the state of VMs.
RIFT-27186	Service instantiation fails with incorrect timeout error	RIFT.ware was not capturing the real reason why pods failed. Pods only retry in the case of failure, so it was not possible to say that pod creation was in a Failure state.	Instantiation fails in this scenario.	View the cluster event logs in Launchpad.
RIFT-27241	Instantiating ping-pong fails after ping VNF descriptor is modified and it is	If a user choses the management interface type in a VNFD as VDU or Connection Point and the corresponding	If a user chooses the mgmt-interface type but no ID is selected, then RIFT.ware considers	If a user configures the NS properly, then an issue will not occur.

Support Tickets	Title	Description	Impact	Workaround
	impossible to tell why	VDU/CP is not selected in the next input, then no error appears until the user attempts to instantiate. The configuration fails in the instantiation attempt.	both the type and the ID as empty.	
RIFT-28142	Create service progress did not complete but service is created. Cannot instantiate.	An operator attempted to instantiate a Network Service and it failed to initialize. However, the Services tab on the UI lists the service as Created.	This issue only occurs while creating Projects and adding datacenter accounts via the API. The problem occurs when a Datacenter account is created before project creation is complete. This issue will not occur when using the UI.	If you are using APIs, then, create another project and wait for 30 seconds before creating a Datacenter account. Use the following API call to verify that the project creation is complete: https://<lp-ip>:8008/api/operational/project/<default>/project-state
RIFT-28834	Overwrite existing package option is not available	An operator onboarded a package in the UI. An operator might want to onboard the same package again and overwrite the existing package. The option to overwrite the existing package is missing from the UI.	It is not possible to overwrite an existing package in the UI.	If a package is not in use, then it can be deleted from the catalog and then onboarded again. To onboard an updated version of a package, ensure that it has a new unique id. Note: When using Launchpad to copy a package, the new package is automatically given a new ID. The package with the new ID can be onboarded and it will

Support Tickets	Title	Description	Impact	Workaround
				co-exist with the current package.
RIFT-29543	CNFD service instantiation failing with error release monitoring failed: 'str' object has no attribute 'type_yang'	CNFD service instantiation fails when a helm-chart-based Launchpad is setup in a non-default namespace and uses the chart to launch CNFs in other namespaces. The issue is not seen with standalone Launchpads.	NS instantiations fail with the following error message: Release monitoring failed.	Use kube-config with a explicitly defined namespace.
RIFT-30839	Rollback is not supported if segment profile is not created in datacenter while using import-instantiation-variables RPCs	An operator added a datacenter and then uploaded ping pong descriptors. In the Create NS wizard, a user attached the instantiation variable file and clicked YES on the instantiate radio button. This caused the NS to be stuck in Failed state.	This only impacts the import-instantiation-variables rpc. When an operator instantiates an NS using the above steps, then rollback is not supported.	Do not click YES on the instantiate radio button. To avoid this issue, instantiate from the services page.
RIFT-31050	Changing user login password throws "Server Error - 503" message	Error 503 returned when updating a user password.	The user password is still updated and this error can be safely ignored.	N/A
RIFT-31075	Common LP Discovery status is in discovered state even after kubeconfig is modified incorrectly	After configuring a valid Kubernetes account from the datacenter tab, the account is successful and discovery is validated. Next, an operator updated a kubeconfig	Discovery state is not updated when connectivity to the k8s cluster is lost.	N/A

Support Tickets	Title	Description	Impact	Workaround
		by making a slight change. The discovery state did not change to “Undiscovered” but the account status changed to failed.		