**Installation Guides**

This document describes how to install the RIC components deployed by scripts and Helm charts under the it/dep repository, including the dependencies and required system resources.

* [Version history](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#version-history)
* [Installing Near-realtime RIC](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#installing-near-realtime-ric)
  + [Prerequisites](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#prerequisites)
    - [VirtualBox VMs as Installation Hosts](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#virtualbox-vms-as-installation-hosts)
    - [One-Node Kubernetes Cluster](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#one-node-kubernetes-cluster)
  + [RIC Platform](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#ric-platform)
  + [AUX Functionalities (Optional)](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#aux-functionalities-optional)
  + [RIC Applications](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#ric-applications)
* [Installing Non-realtime RIC](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#installing-non-realtime-ric)
  + [Non-realtime RIC Platform](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#non-realtime-ric-platform)

[**Version history**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id1)

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Ver.** | **Author** | **Comment** |
| 2019-11-25 | 0.1.0 | Lusheng Ji | Amber |
| 2020-01-23 | 0.2.0 | Zhe Huang | Bronze RC |
| 2020-01-23 | 0.3.0 | Zhe Huang | Cherry |

[**Installing Near-realtime RIC**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id2)

The installation of Near Realtime RAN Intelligent Controller is spread onto two separate Kubernetes clusters. The first cluster is used for deploying the Near Realtime RIC (platform and applications), and the other is for deploying other auxiliary functions. They are referred to as RIC cluster and AUX cluster respectively.

The following diagram depicts the installation architecture.

[Graphical user interface, treemap chart

Description automatically generated](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/_images/nrtric-amber.png)

Within the RIC cluster, Kubernetes resources are deployed using three name spaces: ricinfra, ricplt, and ricxapp by default. Similarly, within the AUX cluster, Kubernetes resources are deployed using two name spaces: ricinfra, and ricaux.

For each cluster, there is a Kong ingress controller that proxies incoming API calls into the cluster. With Kong, service APIs provided by Kubernetes resources can be accessed at the cluster node IP and port via a URL path. For cross-cluster communication, in addition to Kong, each Kubernetes namespace has a special Kubernetes service defined with an endpoint pointing to the other cluster’s Kong. This way any pod can access services exposed at the other cluster via the internal service hostname and port of this special service. The figure below illustrates the details of how Kong and external services work together to realize cross-cluster communication.

[Graphical user interface, application

Description automatically generated](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/_images/kong-extservice.png)

[**Prerequisites**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id3)

Both RIC and AUX clusters need to fulfill the following prerequisites.

* Kubernetes v.1.16.0 or above
* helm v2.12.3/v3.5.x or above
* Read-write access to directory /mnt

The following two sections show two example methods to create an environment for installing RIC.

[**VirtualBox VMs as Installation Hosts**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id4)

The deployment of Near Realtime RIC can be done on a wide range of hosts, including bare metal servers, OpenStack VMs, and VirtualBox VMs. This section provides detailed instructions for setting up Oracle VirtualBox VMs to be used as installation hosts.

**Networking**

The set up requires two VMs connected by a private network. With VirtualBox, this can be done by going under its “Preferences” menu and setting up a private NAT network.

1. Pick “Preferences”, then select the “Network” tab;
2. Click on the “+” icon to create a new NAT network. A new entry will appear in the NAT networks list
3. Double click on the new network to edit its details; give it a name such as “RICNetwork”
4. In the dialog, make sure to check the “Enable Network” box, uncheck the “Supports DHCP” box, and make a note of the “Network CIDR” (for this example, it is 10.0.2.0/24);
5. Click on the “Port Forwarding” button then in the table create the following rules:
   1. “ssh to ric”, TCP, 127.0.0.1, 22222, 10.0.2.100, 22;
   2. “ssh to aux”, TCP, 127.0.0.1, 22223, 10.0.2.101, 22;
   3. “entry to ric”, TCP, 127.0.0.1, 22224, 10.0.2.100, 32080;
   4. “entry to aux”, TCP, 127.0.0.1, 22225, 10.0.2.101, 32080.
6. Click “Ok” all the way back to create the network.

**Creating VMs**

Create a VirtualBox VM:

1. “New”, then enter the following in the pop-up: Name it for example **myric**, of “Linux” type, and at least 6G RAM and 20G disk;
2. “Create” to create the VM. It will appear in the list of VMs.
3. Highlight the new VM entry, right click on it, select “Settings”.
   1. Under the “System” tab, then “Processor” tab, make sure to give the VM at least 2 vCPUs.
   2. Under the “Storage” tab, point the CD to a Ubuntu 18.04 server ISO file;
   3. Under the “Network” tab, then “Adapter 1” tab, make sure to:
      1. Check “Enable Network Adapter”;
      2. Attached to “NAT Network”;
      3. Select the Network that was created in the previous section: “RICNetwork”.

Repeat the process and create the second VM named **myaux**.

**Booting VM and OS Installation**

Follow the OS installation steps to install OS to the VM virtual disk media. During the setup you must configure static IP addresses as discussed next. And make sure to install openssh server.

**VM Network Configuration**

Depending on the version of the OS, the networking may be configured during the OS installation or after. The network interface is configured with a static IP address:

* IP Address: 10.0.2.100 for myric or 10.0.2.101 for myaux;
* Subnet 10.0.2.0/24, or network mask 255.255.255.0
* Default gateway: 10.0.2.1
* Name server: 8.8.8.8; if access to that is is blocked, configure a local DNS server

**Accessing the VMs**

Because of the port forwarding configurations, the VMs are accessible from the VirtualBox host via ssh.

* To access **myric**: ssh [{{USERNAME}}@127.0.0.1](mailto:%7B%7BUSERNAME%7D%7D%40127.0.0.1) -p 22222
* To access **myaux**: ssh [{{USERNAME}}@127.0.0.1](mailto:%7B%7BUSERNAME%7D%7D%40127.0.0.1) -p 22223

[**One-Node Kubernetes Cluster**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id5)

This section describes how to set up a one-node Kubernetes cluster onto a VM installation host.

**Script for Setting Up 1-node Kubernetes Cluster**

The it/dep repo can be used for generating a simple script that can help setting up a one-node Kubernetes cluster for dev and testing purposes. Related files are under the **tools/k8s/bin** directory. Clone the repository on the target VM:

% git clone https://gerrit.o-ran-sc.org/r/it/dep

**Configurations**

The generation of the script reads in the parameters from the following files:

* etc/env.rc: Normally no change needed for this file. If where the Kubernetes cluster runs has special requirements, such as running private Docker registry with self-signed certificates, or hostnames that can be only resolved via private /etc/hosts entries, such parameters are entered into this file.
* etc/infra.rc: This file specifies the docker host, Kubernetes, and Kubernetes CNI versions. If a version is left empty, the installation will use the default version that the OS package management software would install.
* etc/openstack.rc: If the Kubernetes cluster is deployed on Open Stack VMs, this file specifies parameters for accessing the APIs of the Open Stack installation. This is not supported in Amber release yet.

**Generating Set-up Script**

After the configurations are updated, the following steps will create a script file that can be used for setting up a one-node Kubernetes cluster. You must run this command on a Linux machine with the ‘envsubst’ command installed.

Nicher oita korar age: envsubst>>sudo apt-get install gettext-base

Do> **ls -al** (see the result)

Do>> **cd dep** then Do>> **ls -al**

Now follow under command

% cd tools/k8s/bin

% ./gen-cloud-init.sh

A file named **k8s-1node-cloud-init.sh** would now appear under the bin directory.

**Setting up Kubernetes Cluster**

The new **k8s-1node-cloud-init.sh** file is now ready for setting up the Kubernetes cluster.

It can be run from a root shell of an existing Ubuntu 16.04 or 18.04 VM. Running this script will replace any existing installation of Docker host, Kubernetes, and Helm on the VM. The script will reboot the machine upon successful completion. Run the script like this:

% sudo -i

*# ./k8s-1node-cloud-init.sh*

This script can also be used as the user-data (a.k.a. cloud-init script) supplied to Open Stack when launching a new Ubuntu 16.04 or 18.04 VM.

Upon successful execution of the script and reboot of the machine, when queried in a root shell with the kubectl command the VM should display information similar to below:

*# kubectl get pods --all-namespaces*

NAMESPACE NAME READY STATUS RESTARTS AGE

kube-system coredns-5644d7b6d9-4gjp5 1/1 Running 0 103m

kube-system coredns-5644d7b6d9-pvsj8 1/1 Running 0 103m

kube-system etcd-ljitest 1/1 Running 0 102m

kube-system kube-apiserver-ljitest 1/1 Running 0 103m

kube-system kube-controller-manager-ljitest 1/1 Running 0 102m

kube-system kube-flannel-ds-amd64-nvjmq 1/1 Running 0 103m

kube-system kube-proxy-867v5 1/1 Running 0 103m

kube-system kube-scheduler-ljitest 1/1 Running 0 102m

kube-system tiller-deploy-68bf6dff8f-6pwvc 1/1 Running 0 102m

**Onetime setup for Influxdb**

Once Kubernetes setup is done, we have to create PersistentVolume through the storage class for the influxdb database. The following one time process should be followed before deploying the influxdb in ricplt namespace.

*# User has to check the following namespace exist or not using*

% kubectl get ns ricinfra

*# If the namespace doesn’t exist, then create it using:*

% kubectl create ns ricinfra

% helm install stable/nfs-server-provisioner --namespace ricinfra --name nfs-release-1

% kubectl patch storageclass nfs -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'

% sudo apt install nfs-common

[**RIC Platform**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id6)

After the Kubernetes cluster is installed, the next step is to install the (Near Realtime) RIC Platform.

See instructions in ric-plt/ric-dep: <https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-ric-dep/en/latest/installation-guides.html>

[**AUX Functionalities (Optional)**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id7)

**Resource Requirements**

To run the RIC-AUX cluster in a dev testing setting, the minimum requirement for resources is a VM with 4 vCPUs, 16G RAM and at least 40G of disk space.

**Getting and Preparing Deployment Scripts**

Run the following commands in a root shell:

git clone https://gerrit.o-ran-sc.org/r/it/dep

cd dep

git submodule update --init --recursive --remote

**Modify the deployment recipe**

Edit the recipe file ./RECIPE\_EXAMPLE/AUX/example\_recipe.yaml.

* Specify the IP addresses used by the RIC and AUX cluster ingress controller (e.g., the main interface IP) in the following section. If you are only testing the AUX cluster, you can put down any private IPs (e.g., 10.0.2.1 and 10.0.2.2).

extsvcplt:

ricip: ""

auxip: ""

* To specify which version of the RIC platform components will be deployed, update the RIC platform component container tags in their corresponding section.
* You can specify which docker registry will be used for each component. If the docker registry requires login credential, you can add the credential in the following section. Note that the installation script has already included credentials for O-RAN Linux Foundation docker registries. Please do not create duplicate entries.

docker-credential:

enabled: true

credential:

SOME\_KEY\_NAME:

registry: ""

credential:

user: ""

password: ""

email: ""

For more advanced recipe configuration options, refer to the recipe configuration guideline.

**Deploying the Aux Group**

After the recipes are edited, the AUX group is ready to be deployed.

cd dep/bin

./deploy-ric-aux ../RECIPE\_EXAMPLE/AUX/example\_recipe.yaml

**Checking the Deployment Status**

Now check the deployment status and results similar to the below indicate a complete and successful deployment.

*# helm list -A*

NAME REVISION UPDATED STATUS CHART APP VERSION NAMESPACE

r3-aaf 1 Mon Jan 27 13:24:59 2020 DEPLOYED aaf-5.0.0 onap

r3-dashboard 1 Mon Jan 27 13:22:52 2020 DEPLOYED dashboard-1.2.2 1.0 ricaux

r3-infrastructure 1 Mon Jan 27 13:22:44 2020 DEPLOYED infrastructure-3.0.0 1.0 ricaux

r3-mc-stack 1 Mon Jan 27 13:23:37 2020 DEPLOYED mc-stack-0.0.1 1 ricaux

r3-message-router 1 Mon Jan 27 13:23:09 2020 DEPLOYED message-router-1.1.0 ricaux

r3-mrsub 1 Mon Jan 27 13:23:24 2020 DEPLOYED mrsub-0.1.0 1.0 ricaux

r3-portal 1 Mon Jan 27 13:24:12 2020 DEPLOYED portal-5.0.0 ricaux

r3-ves 1 Mon Jan 27 13:23:01 2020 DEPLOYED ves-1.1.1 1.0 ricaux

*# kubectl get pods -n ricaux*

NAME READY STATUS RESTARTS AGE

deployment-ricaux-dashboard-f78d7b556-m5nbw 1/1 Running 0 6m30s

deployment-ricaux-ves-69db8c797-v9457 1/1 Running 0 6m24s

elasticsearch-master-0 1/1 Running 0 5m36s

r3-infrastructure-kong-7697bccc78-nsln7 2/2 Running 3 6m40s

r3-mc-stack-kibana-78f648bdc8-nfw48 1/1 Running 0 5m37s

r3-mc-stack-logstash-0 1/1 Running 0 5m36s

r3-message-router-message-router-0 1/1 Running 3 6m11s

r3-message-router-message-router-kafka-0 1/1 Running 1 6m11s

r3-message-router-message-router-kafka-1 1/1 Running 2 6m11s

r3-message-router-message-router-kafka-2 1/1 Running 1 6m11s

r3-message-router-message-router-zookeeper-0 1/1 Running 0 6m11s

r3-message-router-message-router-zookeeper-1 1/1 Running 0 6m11s

r3-message-router-message-router-zookeeper-2 1/1 Running 0 6m11s

r3-mrsub-5c94f5b8dd-wxcw5 1/1 Running 0 5m58s

r3-portal-portal-app-8445f7f457-dj4z8 2/2 Running 0 4m53s

r3-portal-portal-cassandra-79cf998f69-xhpqg 1/1 Running 0 4m53s

r3-portal-portal-db-755b7dc667-kjg5p 1/1 Running 0 4m53s

r3-portal-portal-db-config-bfjnc 2/2 Running 0 4m53s

r3-portal-portal-zookeeper-5f8f77cfcc-t6z7w 1/1 Running 0 4m53s

[**RIC Applications**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id8)

See instructions in ric-plt/ric-dep: <https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-ric-dep/en/latest/installation-guides.html>

[**Installing Non-realtime RIC**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id9)

Non-realtime RAN Intelligent Controller is deployed on a kubernetes cluster with helm being the package manager. Please refer to the near-realtime RIC installation about preparing for a kubernetes cluster. A recipe file is used to configure the non-realtime RIC instance.

[**Non-realtime RIC Platform**](https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/installation-guides.html#id10)

After the Kubernetes cluster is ready, the next step is to install the Non-realtime RIC Platform.

**Getting and Preparing Deployment Scripts**

Clone the it/dep git repository that has deployment scripts and support files on the target VM. (You might have already done this in a previous step.)

% git clone https://gerrit.o-ran-sc.org/r/it/dep

Check out the appropriate branch of the repository with the release you want to deploy. For example:

git clone https://gerrit.o-ran-sc.org/r/it/dep

cd dep

git submodule update --init --recursive --remote

**Modify the deployment recipe**

Edit the recipe files ./RECIPE\_EXAMPLE/NONRTRIC/example\_recipe.yaml.

* To specify which version of the non-realtime RIC platform components will be deployed, update the platform component container tags in their corresponding section.
* You can specify which docker registry will be used for each component. If the docker registry requires login credential, you can add the credential in the following section. Please note that the installation suite has already included credentials for O-RAN Linux Foundation docker registries. Please do not create duplicate entries.

docker-credential:

enabled: true

credential:

SOME\_KEY\_NAME:

registry: ""

credential:

user: ""

password: ""

email: ""

**Deploying Platform**

After the recipes are edited, the Near Realtime RIC platform is ready to be deployed.

cd dep/bin

./deploy-nonrtric ../RECIPE\_EXAMPLE/PLATFORM/example\_recipe.yaml

**Checking the Deployment Status**

Now check the deployment status after a short wait. Results similar to the output shown below indicate a complete and successful deployment. Check the STATUS column from both kubectl outputs to ensure that all are either “Completed” or “Running”, and that none are “Error” or “ImagePullBackOff”.

*# helm list*

NAME REVISION UPDATED STATUS CHART APP VERSION NAMESPACE

r2-dev-nonrtric 1 Sat Dec 12 02:55:52 2020 DEPLOYED nonrtric-2.0.0 nonrtric

*# kubectl get pods -n ricplt*

NAME READY STATUS RESTARTS AGE

a1-sim-osc-0 1/1 Running 0 2m8s

a1-sim-std-0 1/1 Running 0 2m8s

a1controller-ff7c8979f-l7skk 1/1 Running 0 2m8s

controlpanel-859d8c6c6f-hmzrj 1/1 Running 0 2m8s

db-6b94f965dc-dsch6 1/1 Running 0 2m8s

enrichmentservice-587d7d8984-vllst 1/1 Running 0 2m8s

policymanagementservice-d648f7c9b-54mwc 1/1 Running 0 2m8s

rappcatalogueservice-7cbbc99b8d-rl2nz 1/1 Running 0 2m8s

**Undeploying Platform**

To undeploy all the containers, perform the following steps in a root shell within the it-dep repository.

*# cd bin*

*# ./undeploy-nonrtric*

Results similar to below indicate a complete and successful cleanup.

*# ./undeploy-ric-platform*

Undeploying NONRTRIC components [controlpanel a1controller a1simulator policymanagementservice enrichmentservice rappcatalogueservice]

release "r2-dev-nonrtric" deleted

configmap "nonrtric-recipe" deleted

namespace "nonrtric" deleted