

Rafay Platform

Installation of Rafay Controller
in Air Gapped Environments



Version History

Version	Date Modified	Changes
0.1	Oct 17, 2024	Initial version based on Rafay docs
1.0	Nov 15, 2024	Standalone version

TABLE OF CONTENTS

Prerequisites	4
Infrastructure Requirements	4
Logo [Optional]	7
X509 Certificates [optional]	7
SSL Offloading in Load Balancer	7
DNS settings for using External SSL Offload.	8
Installation Steps	9
Preparation	9
Install radm services	9
Configuration	10
Access the Controller Web Console	14
Uploading Cluster Dependencies	16
Appendix	17
Multiple Interface Support	17
Hosted DNS Support	17

Prerequisites

Ensure that you have the download URL for the controller software package and the checksum information to validate once you download the package. The download location is generally in the following format.

`https://rafay-airgap-controller.s3.us-west-2.amazonaws.com/x.y/rafay-airgap-controller-x.y-z.tar.gz`

Infrastructure Requirements

The following table captures the requirements for the infrastructure associated with the installation of the Rafay Controller in an air gapped environment.

Operating System	<ul style="list-style-type: none">- Ubuntu 22.04 LTS- RHEL 8 or 9 <p>INFO <i>RHEL installations need Internet connectivity to the default repository servers. Licensing does not permit bundling into the package.</i></p>
Number of Nodes	Minimal, single node install (1 node) Highly Available, Multi Node (3 master, 1 worker node)
Node Capacity	16 CPU, 64 GB Memory (Minimum)
Root Disk	250 GB (Minimum)
Temp Directory (/tmp)	50 GB Minimum, if not part of the root disk
Data Disk (formatted)	500 GB (attached as /data volume). Note that size is dependent on actual storage requirements
Network	Allow inbound 443/tcp to all instances. Ensure that all localhost ports are reachable.
Security	Ensure SELinux and firewall is disabled in all the nodes
DNS	For non-DNS environments, ensure that port 300,053/UDP is reachable.

Listed below are Linux commands that administrators can use to configure their nodes.

Disable Firewall

```
$ sudo systemctl stop firewalld
$ sudo systemctl disable firewalld
```

Disable SELinux

```
$ sudo sed -c -i "s/SELINUX=.*/SELINUX=disabled/" /etc/sysconfig/selinux
$ sudo reboot
```

Disable cgroup v2

For nodes based on Ubuntu 22.04 LTS or higher, please make sure to disable “cgroup v2”

```
$ sudo sed -i
's/^GRUB_CMDLINE_LINUX=""$/GRUB_CMDLINE_LINUX="systemd.unified_cgroup_hie
rarchy=0"/ /etc/default/grub

$ sudo update-grub
$ sudo reboot

Login into the node again.
$ sudo iptables -F && sudo iptables -t nat -F && sudo iptables -t mangle -F && sudo
iptables -X
```

DNS Record Creation

The installation of the Rafay controller software requires wildcard records as mentioned below. In the below examples, replace *rafay.example.com* with the desired domain.

`*.rafay.example.com`

In case wildcard DNS is not available, ensure that individual records are created as listed below.

1. `api.<rafay.example.com>`
2. `console.<rafay.example.com>`
3. `fluentd-aggr.<rafay.example.com>`
4. `grafana.<rafay.example.com>`
5. `kibana.<rafay.example.com>`
6. `nexus.<rafay.example.com>`
7. `ops-console.<rafay.example.com>`
8. `rcr.<rafay.example.com>`
9. `regauth.<rafay.example.com>`
10. `repo.<rafay.example.com>`
11. `*.cdrelay.<rafay.example.com>`
12. `*.core-connector.<rafay.example.com>`
13. `*.core.<rafay.example.com>`
14. `*.connector.infrarelay.<rafay.example.com>`
15. `*.user.infrarelay.<rafay.example.com>`
16. `*.kubeapi-proxy.<rafay.example.com>`
17. `*.user.<rafay.example.com>`

DNS records for the wildcard FQDN should point to the controller nodes' IP addresses. If external SSL offloading is used, please refer to [section 1.5](#)

Logo [Optional]

Provide a logo of your product/company so that your controller deployment can be white labeled. Provide a logo of size less than 600 KB in png format for white labeling and branding purposes.

X509 Certificates [optional]

The controller uses TLS for secure communication. As a result, x509 certificates are required to secure all endpoints. Customers are expected to provide a trusted CA signed wildcard certificate for the target DNS (e.g. *.rafay.example.com).

INFO

For non-prod/internal scenarios, If trusted CA signed certificates are not available then the controller can generate self-signed certificates automatically. This can be achieved by setting the “generate-self-signed-certs” key to “True” in the config.yaml during installation.

SSL Offloading in Load Balancer

Rafay controller supports SSL offload at load balancer level using ACM/certificates. This would need two load balancers, one for UI FQDNs which requires SSL offload and another for backed FQDNs which requires SSL passthrough. To enable external SSL offloading, the below override-config has to be enabled in config.yaml.

```
override-config.global.external_lb: true
```

DNS settings for using External SSL Offload.

For extended security, all Rafay backend endpoints use mTLS and they do not support offloading SSL except the frontend UI endpoints.

Below frontend FQDNs should be pointed to the Classic LB for SSL offloading:

1. *api.<rafay.example.com>*
2. *console.<rafay.example.com>*
3. *fluentd-aggr.<rafay.example.com>*
4. *ops-console.<rafay.example.com>*
5. *grafana.<rafay.example.com>*
6. *repo.<rafay.example.com>*

FQDNs pointing to the NLB.[mTLS]

1. *rcr.<rafay.example.com>*
2. *regauth.<rafay.example.com>*
3. *registry.<rafay.example.com>*
4. **.core-connector.<rafay.example.com>*
5. **.core.<rafay.example.com>*
6. **.kubeapi-proxy.<rafay.example.com>*
7. **.user.<rafay.example.com>*
8. **.cdrelay.<rafay.example.com>*
9. **.infrarelay.<rafay.example.com>*
10. **.connector.infrarelay.<rafay.example.com>*
11. **.user.infrarelay.<rafay.example.com>*

Installation Steps

Preparation

- Create instance with the specifications described in [section 1.1](#) above
- Create wildcard DNS entries for the Controller domains mentioned in [section 1.2](#), and point their A record to node/LB IP addresses.

Optional Step

Generate a wildcard certificate for the FQDN and get it signed by a trusted CA. Alternatively, configure the controller to use self-signed certificates.

Install radm services

Download the controller installation package from the following URL to ALL controller nodes. You should have the <download URL> and checksum value from Rafay for the package.

```
$ wget <downloadurl>
```

Verify the downloaded package, check if the checksum matches as given to you.

```
$ md5sum <downloaded-package>
```

Note: make sure the name of the tar package is correct

From your controller instance home directory, untar the package using the command below

```
$ tar -xf rafay-controller-airgap-*.tar.gz
```

Configuration

Copy and edit the config.yaml file.

```
$ sudo mv ./radm /usr/bin/  
$ cp -rp config.yaml-airgap-tmpl config.yaml  
$ vi config.yaml
```

Customize config.yaml

```
spec.deployment.blueprintVersion: v2 # Use v2 for v2 controller  
spec.deployment.ha: set to true if its HA controller.  
spec.repo.*.path: Path of the tar location  
spec.app-config.generate-self-signed-certs: if true then generates  
and uses self signed certs for incoming core traffic.  
spec.app-config.partner.star-domain: Wildcard FQDN (*.example.com)  
spec.app-config.logo: Display logo in UI.  
Spec.app-config.super-user.user : Provide an email id which uses as an  
admin for managing ops-console  
Spec.app-config.super-user.password : Provide a password for the  
ops-console admin  
spec.override-config.global.tsdb_backup.enabled: false  
spec.override-config.global.secrets.tsdb.gke.storage_account_k  
ey: "" # base64 encoded value of Service account Json  
spec.override-config.global.enable_hosted_dns_server: set true if  
DNS is not available.  
spec.override-config.global.external_lb: set true to SSL offload in LB.  
spec.override-config.global.use_instance_role: set true to provision  
EKS clusters using controller instance IAM role for EC2 controller. Refer section 6.1.  
If the above instance role is not used, we can use the below parameter for adding cross account  
ID and credentials.  
spec.override-config.global.secrets.aws_account_id  
spec.override-config.global.secrets.aws_access_key_id  
spec.override-config.global.secrets.aws_secret_access_key
```

If your Airgap controller is a HA installation, then copy the updated `config.yaml` to every controller node/instance. Keep the same folder path in all controller instances to store the extracted controller packages.

Initialize the Controller from the controller node using the command shown below. If the controller is HA, then run the following command from any of the controller instances.

```
$ sudo radm init --config config.yaml
```

Once initialization is complete, copy the admin config file to the home directory to access the kube controller API from CLI of the particular controller instance. This is the same step for both single node and HA controllers.

```
$ mkdir -p $HOME/.kube  
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
$ sudo chown $(id -u):$(id -g) -R $HOME/.kube
```

If this is an Airgap HA controller, then you will see output similar to the following.

```
You can now join any number of the control-plane node running the  
following command on each as root:
```

```
radm join 10.1.0.26:6443 --token 0bqke6.w2wo2hc35kjwnbna \  
--discovery-token-ca-cert-hash  
sha256:24df9ec7ca2af55f8644afd36fb151c2588d0cb2f99f67e837c3a3e9690d5d0d  
\  
--control-plane --certificate-key  
210573eef5a9663c15a5c63ac45275aa0b40b6b13ca9249b80f915e5c278bd1f  
--config config.yaml
```

```
Please note that the certificate-key gives access to cluster sensitive  
data, keep it secret!  
As a safeguard, uploaded-certs will be deleted in two hours; If  
necessary, you can use  
"radm init phase upload-certs --upload-certs" to reload certs afterward.
```

```
Then you can join any number of worker nodes by running the following on  
each as root:
```

```
radm join 10.1.0.26:6443 --token 0bqke6.w2wo2hc35kjwnbna \  

```

```
--discovery-token-ca-cert-hash  
sha256:24df9ec7ca2af55f8644afd36fb151c2588d0cb2f99f67e837c3a3e9690d5d0d  
--config config.yaml
```

(Only for HA controller) From the above output, run the “radm join” command with sudo permission which has “--control-plane” in the command, in all other controller master nodes. Ensure to run only on master nodes.

Sample command:

```
radm join 10.1.0.26:6443 --token 0bqke6.w2wo2hc35kjwnbna \  
--discovery-token-ca-cert-hash  
sha256:24df9ec7ca2af55f8644afd36fb151c2588d0cb2f99f67e837c3a3e9690d5d0d \  
--control-plane --certificate-key  
210573eef5a9663c15a5c63ac45275aa0b40b6b13ca9249b80f915e5c278bd1f --config config.yaml
```

(Only for HA controller) From the above output, run the “radm join” command using sudo permission which does not have “control-plane, in the command, in all other controller worker nodes. Ensure to run only on worker nodes.

Sample command:

```
radm join 10.1.0.26:6443 --token 0bqke6.w2wo2hc35kjwnbna \  
--discovery-token-ca-cert-hash  
sha256:24df9ec7ca2af55f8644afd36fb151c2588d0cb2f99f67e837c3a3e9690d5d0d --config  
config.yaml
```

Now, all these nodes are joined in HA mode. Now, it is time to deploy the Rafay Controller software application on top of this substrate.

If this is an HA controller, then run the following commands from any of the master nodes where you keep all the controller packages.

NOTE

- These commands should run only once on any of the master.
- You don't have to run the same command on other master nodes.
- If it is a Single node Airgap controller, then run the following commands from the corresponding controller instance.

Run the below command to initialize the dependencies in the controller.

Note: In case of a HA controller, run the following commands only once on any master.

```
$ sudo radm dependency --config config.yaml
```

Install the rafay application.

```
$ sudo radm application --config config.yaml
```

NOTE

*This will bring up all rafay services. Note that this will take approximately **20-30 mins** for all pods to be up and ready.*

Before proceeding further, confirm that all pods are in running state using kubectl.

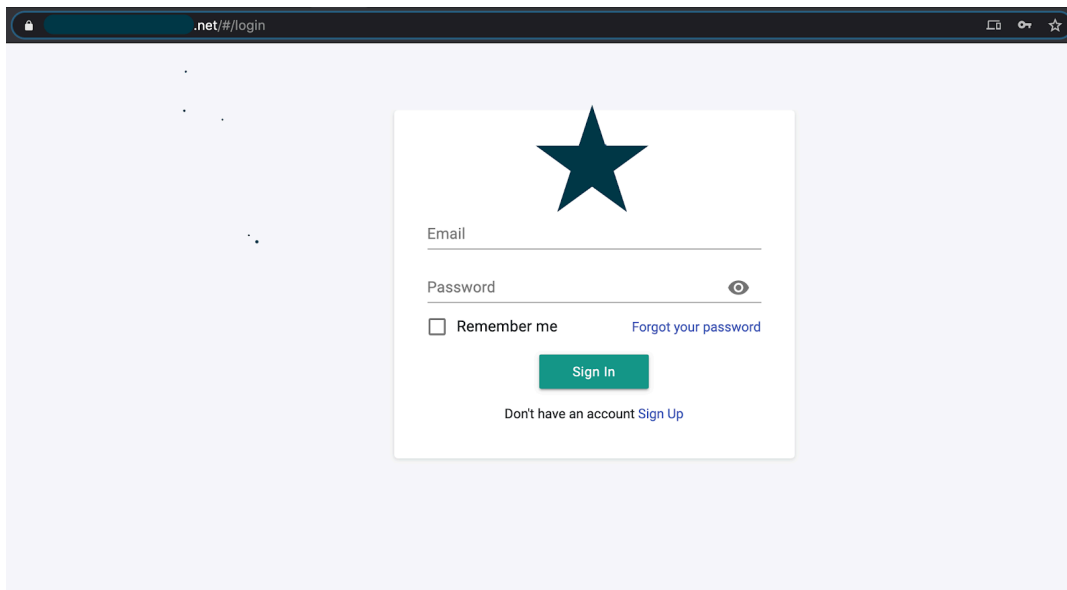
```
$ kubectl get pods -A
```

Access the Controller Web Console

Try accessing the Rafay air-gapped controller's web console to verify that the installation was successful. Open the following URL in a web browser (ideally Chrome)

<https://console.<rafay.example.com>>

You should see a screen similar to the image below when you access the UI.



Click on the “Sign Up” link to create the first Organization of the Rafay air-gapped controller.

Register a new account for the organization as below screenshot

console:// #/signup

Create an account

First Name

Last Name


Organization Name

Email

Password

Passwords must be atleast 8 characters long

Confirm Password



Enter letters displayed above

Then try to login to this Organization with the newly registered account on the login screen.

Uploading Cluster Dependencies

Run the below command on any of the master nodes only once to enable support for Kubernetes cluster provisioning from the Rafay air-gapped controller and upload dependencies for Kubernetes cluster provisioning to the controller.

```
$ sudo radm cluster --config config.yaml
```

NOTE

- The radm cluster command has a provision to pass the **--num-threads** if you'd want more threads to execute the task.
- It is set to **1** by default, however it can be increased based on the size of the node you're provisioning this.
- Setting this to a higher value could have a direct impact on the nexus pod, hence it advisable to set it to a value between **1 to max CPU count of the node**

Appendix

Multiple Interface Support

The Rafay controller software supports multiple interfaces and it can be set in the config.yaml file during the initialization. The selected interface is used for all connections related to Rafay apps and Kubernetes. In default, the primary interface is used.

```
spec:
  networking:
    interface: ens3
```

In cases where complete interface isolation is needed, few pods which use host networks like the monitoring/metrics pods, do not adhere to the interface selection on k8s layer and still use the default interface. If complete traffic isolation on the interface is needed, then we recommend to add the below routing rules on your controller and clusters.

```
$ ip route add 10.96.0.0/12 dev <secondary-interface>
$ ip route add 10.224.0.0/16 dev <secondary-interface>
```

Hosted DNS Support

In the absence of DNS servers in your environment, the managed clusters will not have a way to communicate to the controller. In this case, the Rafay Controller can host its own DNS server and propagate the records to the cluster.

Hosted DNS can be enabled on the config.yaml using the below flag in the controller.

```
override-config:
  global.enable_hosted_dns_server: true
```

For accessing the controller UI in your local machine, add a /etc/hosts entry pointing the console FQDN to your controller IP.

```
123.456.789.012 console.<rafay.example.com>
```

While provisioning clusters, add the “-d < controller-ip >” to the conjurer command.

```
$ tar -xjf conjurer-linux-amd64.tar.bz2 && sudo ./conjurer -edge-name="test" \
-passphrase-file="passphrase.txt" -creds-file="credentials.pem" -dns-server
< Controller-IP >
```